1. Обзор

Logging operator решает задачи логирования в среде Kubernetes, автоматизируя развертывание и настройку конвейера логирования.

- Logging operator развертывает и конфигурирует на каждом узле сборщик логов (Fluent Bit DaemonSet) для сбора логов контейнеров и приложений из файловой системы узла.
- Fluent Bit отправляет запрос на API Kubernetes, добавляет метаданные о pod'ах к логам и передает их в экземпляр форвардера логов.
- Экземпляр форвардера логов получает, фильтрует и преобразует входящие логи и передает их на одну или несколько точек выхода. Logging operator поддерживает Fluentd и syslog-ng (через AxoSyslog syslog-ng distribution) в качестве форвардеров логов.

2. Основные характеристики

- Изоляция Namespace.
- Встроенные селекторы меток Kubernetes.
- Защищенная связь (TLS).
- Проверка конфигурации.
- Поддержка нескольких потоков (несколько логов для разных преобразований).
- Поддержка нескольких выходов (хранение одних и тех же логов в нескольких хранилищах: S3, GCS, ES, Loki и других).
- Поддержка нескольких систем логирования (развертывание нескольких экземпляров Fluentd, Fluent Bit в одном кластере).
- Поддержка syslog-ng и Fluentd в качестве центрального компонента маршрутизации логов.

3. Архитектура

Logging operator управляет сборщиками и форвардерами логов в инфраструктуре логирования, а также правилами маршрутизации, которые определяют адресатов различных логов.

Сборщики логов — это агенты конечных устройств, которые собирают логи с узлов Kubernetes и отправляют их форвардерам логов. Logging operator использует Fluent Bit в качестве агентов сборщика логов.

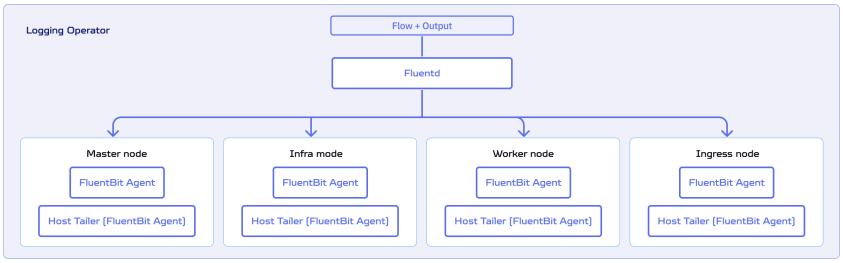
Экземпляр форвардера логов (он же агрегатор логов) получает, фильтрует и преобразует входящие логи и передает их на одну или несколько точек выхода. Logging operator использует Fluentd и syslog-ng в качестве форвардеров логов. Выбор форвардера зависит от ваших требований к логированию.

Фильтровать и обрабатывать входящие логи можно с помощью пользовательского ресурса flow (поток) форвардера логов и направлять их на соответствующий output (выход). Выходы — это пункты назначения для логов, например, OpenSearch или сервер Syslog. Можно настроить выходы и потоки на уровне кластера, например, использовать общий выход для всех пользователей в Namespace, но ограничить права на его изменение. У каждого типа форвардера (Fluentd или syslog-ng) свои потоки и выходы.

Для настройки Logging operator используются следующие пользовательские определения ресурсов (CRD / Custom Resource Definitions):

- **Logging**: ресурс logging определяет инфраструктуру логирования (сборщики и форвардеры логов) в кластере Kubernetes. В нем также содержатся конфигурации для Fluent Bit, Fluentd и syslog-ng.
- **CRDs для Fluentd**:
  - **Output**: Определяет выход для потока логов, куда Fluentd отправляет логи. Это ресурс в Namespace.
  - **Flow**: Фильтрует и обрабатывает поток логов Fluentd. Потоки задают маршруты для выбранных логов к указанным выходам. Это ресурс Namespace.
  - **ClusterOutput**: определяет выход Fluentd, доступный для всех потоков кластера. Logging operator работает с этим выходом кластера в Namespace, заданном в `controlNamespace`, только если в параметре `allowClusterResourcesFromAllNamespaces` не установлено значение `true`.
  - **ClusterFlow**: определяет поток логов Fluentd, который собирает логи из всех Namespace. Logging operator работает с этим потоком в Namespace, заданном в `controlNamespace`, только если в параметре `allowClusterResourcesFromAllNamespaces` не установлено значение `true`.
- **CRDs для Syslog-ng** (эти ресурсы аналогичны соответствующим ресурсам Fluentd, но адаптированы к функциям, доступным через syslog-ng):
  - **SyslogNGOutput**: Определяет выход syslog-ng для потока логов, куда логи отправляются с помощью Fluentd Это ресурс в Namespace.
  - **SyslogNGFlow**: Фильтрует и обрабатывает поток логов syslog-ng. Потоки задают маршруты для выбранных логов к указанным выходам. Это ресурс Namespace.
  - **SyslogNGClusterOutput**: определяет выход syslog-ng, доступный для всех потоков кластера. Logging operator работает с этим выходом кластера в Namespace, заданном в `controlNamespace`, только если в параметре `allowClusterResourcesFromAllNamespaces` не установлено значение `true`.

- **SyslogNGClusterFlow**: определяет поток логов syslog-ng, который по умолчанию собирает логи из всех Namespace. Logging operator работает с этим потоком в Namespace, заданном в `controlNamespace`, только если в параметре `allowClusterResourcesFromAllNamespaces` не установлено значение `true`.



4. Особенности работы

При установке форвардера логов (Fluentd или Syslog-ng) автоматически создаются Fluent Bit агенты, которые начинают собирать и отправлять логи в форвардер. Однако, поскольку выход (Output) и поток (Flow) не настроены, логи не передаются дальше.

Можно настроить выход (Output) и поток (Flow) сразу при создании форвардера логов, чтобы логи отправлялись в указанное место по умолчанию.

При создании Output задаётся точка выхода, т.е.место, куда будут отправляться логи.

При создании Flow задаётся фильтр логов, что позволяет отправлять в Output только те события, которые соответствуют заданным параметрам.

Output и Flow не применяются сразу. Сначала выполняется проверка на соответствие структуры кода. За это отвечает pod с именем `configcheck`, который запускается каждый раз при изменении настроек. Если этот pod не запускается или возникает ошибка, это указывает на наличие проблемы с настройкой Output или Flow.

Далее конфигурация передаётся форвардеру логов (Fluentd или Syslog-ng). Изменения не применяются сразу, а вступают в силу через некоторое время.

Компоненты Syslog-ng и FluentD должны находиться в разных `Namespace`, поскольку агенты Fluent Bit создаются автоматически при их развертывании. Эти агенты поставляют информацию только тому компоненту, который их создал, и являются уникальными в рамках одного Namespace.

Пример

Устанавливаем Syslog-ng и FluentD в одно `Namespace`. Первым создался Syslog-ng, который автоматически создал агенты. Вторым был развёрнут FluentD, но агентов он не создаёт, т.к. они уже были созданы Syslog-ng. В результате данные будут направляться только к Syslog-ng.

Если после настройки видно, что `configcheck` запускался недавно и форвардер логов работает, а также в консоли логов отсутствуют ошибки, можно зайти в консоль контейнера, чтобы проверить текущие настройки. Возможно, что применился только Output, в результате чего возникла проблема с настройкой Flow.

При удалении компонентов Syslog-ng и FluentD — Fluent Bit Agent не удаляется автоматически. Нужно удалить DaemonSet.

5. Содержание раздела

Обзор

В Nova Container Platfor есть возможность установить сервис OpenSearch, который автоматически начинает собирать логи кластера.

1. Архитектура

1.1. Компоненты модуля

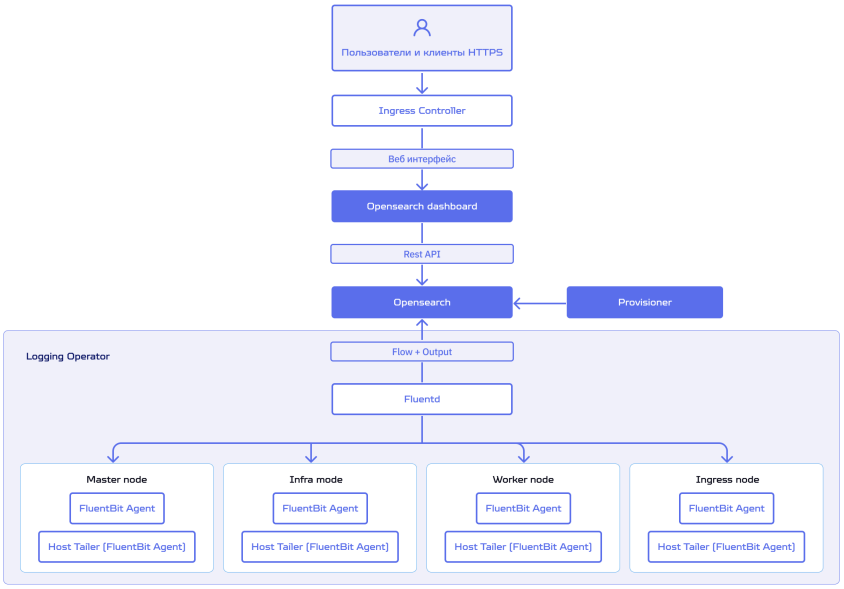
Модуль OpenSearch в кластере Nova Container Platform включает компоненты, представленные в таблице ниже:

Компонент	Категория	Количество
Opensearch	Кластерный компонент	1
Opensearch Dashboard	Кластерный компонент	1
Provisioner	Кластерный компонент	1
FluentbitAgent	Хостовой компонент (агент)	1 на узел
HostTailer	Хостовой компонент (агент)	1 на узел
Fluentd	Кластерный компонент	1

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

- **Opensearch**: основной компонент. В Nova Container Platform контроллер разворачивается в виде сущности StatefulSet. Также контроллер предоставляет REST API для управления сущностями OpenSearch.
- **Opensearch Dashboard**: компонент предоставляет веб-интерфейс для управления сущностями Opensearch.
- **Provisioner**: компонент, запускаемый однократно в виде сущности Job для первоначальной настройки Opensearch.
- **FluentbitAgent**: хостовый компонент (агент), который разворачивается в виде сущности DaemonSet на каждом узле кластера Kubernetes. Предназначен для сбора логов с контейнеров.

- **HostTailer**: аналог FluentbitAgent, но настроенный на сбор логов с узлов кластера.
- **Fluentd**: компонент сервиса сборщика логов. Предназначен для сбора логов с Fluentd в соответствии с заданными правилами (в виде *Custom Resources* `Flow` и `Output`) и отправки их в Opensearch.



1.2. Предварительно настроенные параметры

При установке модуля OpenSearch происходит установка Fluentd, его агентов и правил для сбора логов кластера.

В таблице представлены настройки для сбора и отправки логов в OpenSearch с использованием Logging Operator.

Имя ресурса	**Тип ресурса**	**Название индекса в OpenSearch**
`nova-node-audit-flow`	Flow	`nova-node-audit-<текущая дата>`
`nova-node-audit-output`	Output	`nova-node-audit-<текущая дата>`
`nova-k8s-audit-apiserver-flow`	Flow	`nova-k8s-audit-apiserver-<текущая дата>`
`nova-k8s-audit-apiserver-output`	Output	`nova-k8s-audit-apiserver-<текущая дата>`
`nova-kubelet-flow`	Flow	`nova-kubelet-<текущая дата>`
`nova-kubelet-output`	Output	`nova-kubelet-<текущая дата>`

`nova-starvault-audit-logs-flow`	Flow	`nova-starvault-audit-logs-<текущая дата>`
`nova-starvault-audit-logs-output`	Output	`nova-starvault-audit-logs-<текущая дата>`
`nova-scheduler-flow`	ClusterFlow	`nova-scheduler-<текущая дата>`
`nova-scheduler-output`	ClusterOutput	`nova-scheduler-<текущая дата>`

Также устанавливается дашборд в Grafana в виде *ConfigMap* `nova-grafana-dashboard-opensearch`, который размещается в Namespace `nova-monitoring`, и используется для визуализации и мониторинга основных метрик OpenSearch.

1.3. Предварительно настроенные параметры внутри OpenSearch

После применения Kustomization необходимо дождаться его успешного завершения. В результате установки в кластер OpenSearch будут импортрованы следующие сущности:

- Index pattern
  - nova-k8s-audit-apiserver-* - Аудит лог Kubernetes. Собирается с каждого мастер-узла;
  - nova-node-audit-* - Аудит лог системы. Собирается с каждого узла кластера;
  - nova-kubelet-* - Аудит лог kubelet. Собирается с каждого узла кластера;
  - nova-starvault-audit-logs* - Аудит лог StarVault.
- Dashboards
  - [Nova] Kubernates API audit log
  - [Nova] StarVault audit logs
- Policies
  - nova_policy_1
- Templates
  - nova-base-template

Для доступа ко всем вышеперечисленным ресурсам, необходимо сменить Tenant с Private на Global.

2. Высокая доступность и непрерывность работы

Для обеспечения высокой доступности и непрерывной работы компонентов OpenSearch рекомендуется увеличить количество реплик OpenSearch и Opensearch Dashboard. Это необходимо для предотвращения прерывания работы сервисов в случае недоступности отдельных узлов или при перезапуске сервисов.

Далее рассмотрены особенности работы каждого компонента OpenSearch в режиме высокой доступности.

Opensearch

Для повышения доступности Opensearch рекомендуется устанавливать модуль в рекомендуемой конфигурации.

Opensearch Dashboard

Веб-интерфейс управления сущностями Opensearch взаимодействует с контроллером через REST API и не влияет на работу ключевых сервисов Opensearch. Для повышения доступности рекомендуется устанавливать модуль в рекомендуемой конфигурации. Поддерживается постоянство пользовательских сессий с помощью настройки балансировки в Ingress Controller (`nginx.ingress.kubernetes.io/upstream-hash-by: "$binary_remote_addr"`).

Provisioner

Компонент Provisioner является сущностью типа *Job* в Kubernetes и запускается только один раз при первоначальной установке, поэтому для него высокая доступность и большое количество реплик не требуются.

FluentbitAgent

Разворачивается в виде сущности DaemonSet на каждом узле кластера Kubernetes, поэтому сценарий с увеличением реплик здесь не применим.

HostTailer

Разворачивается в виде сущности DaemonSet на каждом узле кластера Kubernetes, поэтому сценарий с увеличением реплик здесь не применим.

Fluentd

Компонент требует обязательного увеличения количества реплик, если вы обрабатываете большой объем логов.

3. Содержание раздела

- Установка модуля OpenSearch
- Запрет на удаление индексов
- Настройка уведомлений в Opensearch

Примеры использования

Этот раздел описывает все поддерживаемые сценарии использования и примеры конфигураций для управления **LonghornClusterConfig** и **NodeVolumeConfig**.

> Важно соблюдать правильный порядок: перед созданием **LonghornClusterConfig** необходимо создать соответствующий **NodeVolumeConfig**.

1. Последовательность создания и управления

1. Создание **NodeVolumeConfig**

   Для создания пула хранения необходимо сначала определить **NodeVolumeConfig** с необходимыми параметрами:

   ```yaml
   apiVersion: storage.nova-platform.io/v1alpha1
   kind: NodeVolumeConfig
   metadata:
     name: demo
   spec:
     volumeGroup: "my-vg"
     volumes:
       - name: demo-1
         mountPath: /mnt/longhorn-1
         fsType: "xfs"
         size: 27Gi
     nodes:
       - name: worker-135.local
         blockDevices:
           - path: /dev/vdb
   ```

   Описание: Данный **NodeVolumeConfig** создает том `demo-1`, который будет смонтирован в `/mnt/longhorn-1` с файловой системой `xfs` и размером `27Gi`. Блочное устройство `/dev/vdb` подключено на узле `worker-135.local`.

2. Создание **LonghornClusterConfig**

   После определения NodeVolumeConfig можно создать LonghornClusterConfig, используя том из ранее созданного пула хранения:

   ```yaml
   apiVersion: storage.nova-platform.io/v1alpha1
   kind: LonghornClusterConfig
   ```

```yaml
metadata:
  name: longhorn-storage
spec:
  clientNodes:
    - infra-95.local
  storagePool:
    - nodeVolumeConfig:
        name: demo
        volumeName: demo-1
      default: true
```

Описание: Конфигурация **LonghornClusterConfig** создает кластер Longhorn с клиентским узлом `infra-95.local` и использует пул хранения `demo-1` из **NodeVolumeConfig**. Обязательно должен существовать один **StoragePool** с параметром `default: true`. Остальные **StoragePool** по умолчанию получат параметр `default: false`.

После создания ресурса **LonghornClusterConfig** в кластре буду созданы следующие **kustomization**, которые обеспечат установку всех необходимых компонентов для функционирования кластера Longhorn:

- `nova-release-storage-pre`
- `nova-release-storage-main`
- `nova-release-storage-post`

2. Операции с NodeVolumeConfig

2.1. Добавление нового блочного устройства

Чтобы добавить новое блочное устройство в существующий **NodeVolumeConfig**, обновите конфигурацию:

YAML

```yaml
apiVersion: storage.nova-platform.io/v1alpha1
kind: NodeVolumeConfig
metadata:
  name: demo
spec:
  volumeGroup: "my-vg"
  volumes:
    - name: demo-1
      mountPath: /mnt/longhorn-1
      fsType: "xfs"
      size: 27Gi
  nodes:
    - name: worker-135.local
```

```
      blockDevices:
        - path: /dev/vdb
        - path: /dev/vdc ①
```

① Добавлено новое блочное устройство

Описание: Добавление нового блочного устройства `/dev/vdc` на узле `worker-135.local`. В результате объем **volumeGroup** `my-vg` увеличится за счет добавления этого устройства.

2.2. Увеличение размера тома

Для увеличения размера существующего тома обновите параметр `size` в **NodeVolumeConfig**:

YAML

```yaml
apiVersion: storage.nova-platform.io/v1alpha1
kind: NodeVolumeConfig
metadata:
  name: demo
spec:
  volumeGroup: "my-vg"
  volumes:
    - name: demo-1
      mountPath: /mnt/longhorn-1
      fsType: "xfs"
      size: 50Gi ①
  nodes:
    - name: worker-135.local
      blockDevices:
        - path: /dev/vdb
        - path: /dev/vdc
```

① Увеличение размера тома `demo-1` с 27Gi до 50Gi

2.3. Добавление нового Volume

Чтобы добавить новый том в существующий **NodeVolumeConfig**, расширьте список томов:

YAML

```yaml
apiVersion: storage.nova-platform.io/v1alpha1
kind: NodeVolumeConfig
metadata:
  name: demo
spec:
  volumeGroup: "my-vg"
  volumes:
    - name: demo-1
      mountPath: /mnt/longhorn-1
      fsType: "xfs"
      size: 27Gi
```

```
    - name: demo-2 ①
      mountPath: /mnt/longhorn-2
      fsType: "ext4"
      size: 50Gi
  nodes:
    - name: worker-135.local
      blockDevices:
        - path: /dev/vdb
```

① Новый том

Описание: Добавление нового тома `demo-2`, который монтируется в `/mnt/longhorn-2` с файловой системой `ext4`.

2.4. Удаление тома

Для удаления тома из **NodeVolumeConfig** исключите его из списка:

YAML

```yaml
apiVersion: storage.nova-platform.io/v1alpha1
kind: NodeVolumeConfig
metadata:
  name: demo
spec:
  volumeGroup: "my-vg"
  volumes:
    - name: demo-1
      mountPath: /mnt/longhorn-1
      fsType: "xfs"
      size: 27Gi
  nodes:
    - name: worker-135.local
      blockDevices:
        - path: /dev/vdb
```

Описание: Удаление тома `demo-2`, оставляя только том `demo-1`.

2.5. Создание диска на узле Longhorn

Вы можете управлять кластером с использованием стандартных средств Longhorn. При создании диска на узле Longhorn `nova-apps-operator` создаст ресурс **NodeVolumeConfig** с уникальным идентификатором, например `longhorn-volume-f5bba6ab-48cf-4cc3-9c92-4bdde53d0d25`, который будет содержать основную информацию о диске и будет отмечен аннотацией `novaManaged: “false”`.

Пример данного ресурса:

```yaml
apiVersion: storage.nova-platform.io/v1alpha1
kind: NodeVolumeConfig
metadata:
  annotations:
    nova-platform.io/managed-by: "none"
  creationTimestamp: "2024-11-05T11:01:46Z"
  generation: 1
  name: longhorn-volume-f5bba6ab-48cf-4cc3-9c92-4bdde53d0d25
  resourceVersion: "209830"
  uid: 9bfdb382-8029-4d08-8456-eb4c6aac7f6b
spec:
  volumes:
  - mountPath: /mnt/test
    name: disk-1
    size: "0"
```

3. Операции с LonghornClusterConfig

3.1. Удаление LonghornClusterConfig

При удалении **LonghornClusterConfig** также будет удалена соответствующая **kustomization**, а вместе с ней и все созданные ресурсы.

3.2. Изменение LonghornClusterConfig

Можно добавлять или удалять клиентские узлы и **StoragePool**. Однако всегда должен существовать хотя бы один **StoragePool** со значением `default: true`.

```yaml
apiVersion: storage.nova-platform.io/v1alpha1
kind: LonghornClusterConfig
metadata:
  name: longhorn-storage
spec:
  clientNodes:
    - infra-95.local
    - infra-96.local  ①
  storagePool:
    - nodeVolumeConfig:
        name: demo
        volumeName: demo-1
      default: true
    - nodeVolumeConfig:
        name: new-storage
        volumeName: demo-2  ②
      default: false
```

① Добавлен новый клиентский узел `infra-96.local`

② Добавлен новый **StoragePool** `demo-2`.
