

# Alertmanager

Alertmanager — это компонент экосистемы Prometheus, который управляет срабатыванием и обработкой оповещений. Он играет ключевую роль в системе мониторинга на базе Prometheus, организуя оповещения о различных событиях и аномалиях в работе кластера.

## 1. Основные функции Alertmanager

---

- **Маршрутизация оповещений.** Вы можете настроить маршруты для оповещений в зависимости от их источников, приоритетов и других метаданных. Это позволяет направлять сообщения в разные команды или сервера в зависимости от типа проблемы.
- **Группировка оповещений.** Alertmanager агрегирует схожие оповещения в единую группу, что позволяет уменьшить избыточность и снизить уровень информационного шума. Такая агрегация упрощает обработку инцидентов и повышает эффективность управления системой оповещений.
- **Уведомления.** Alertmanager поддерживает интеграцию с различными системами оповещения, такими как электронная почта, Slack, PagerDuty, Telegram и многими другими, что делает его гибким для использования в различных рабочих процессах.
- **Подавление оповещений.** Alertmanager позволяет временно скрывать (или подавлять) определенные оповещения, которые находятся в процессе обработки.
- **Управление тишиной (Silence):** Функция позволяет временно "заглушать" определенные оповещения, чтобы избежать избыточных уведомлений в случае плановых работ или ожидаемых сбоев.

## 2. Особенности Alertmanager в Nova Container Platform

---

### 2.1. Оповещения в веб-консоли Nova Container Platform

В Nova Container Platform оповещения настроены на странице **Observer > Система оповещений**.

- В блоке **Оповещения** видны все текущие проблемы в кластере.
- В блоке **Управление оповещениями** видны все правила остановки оповещения. Также можно создать правило для заглушения оповещений.

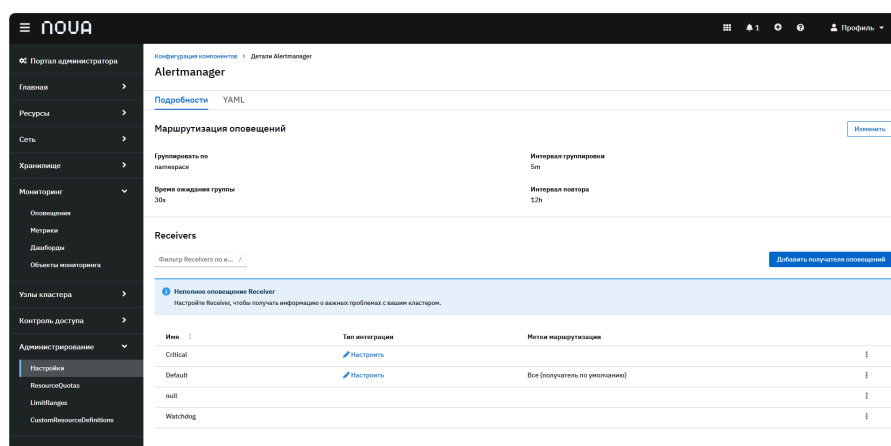
- В блоке **Правила оповещения** видны все настроенные правила для оповещений, включая как предустановленные, так и созданные пользователем.

## 3. Примеры

### 3.1. Пример создания правила для оповещения через web-интерфейс

В данном примере будет настроено оповещение в Telegram через web-интерфейс.

1. В веб-консоли Nova Container Platform перейдите на страницу **Администрирование > Настройки > Конфигурация компонентов > Alertmanager**.



2. Нажмите на [ **Добавить получателя оповещений** ].

#### Новый получатель оповещений

**Имя \***

**Тип \***

Выбрать типа получателя...

- PagerDuty
- Webhook
- Email
- Slack
- Telegram

3. Введите имя для получателя оповещений в поле **"Имя"**.
4. Из выпадающего списка **"Тип"** выберите **"Telegram"**.

## Новый Telegram получатель оповещений

**Имя \***  
New1

**Тип \***  
Telegram

**Токен бота \***  
bot\_id  
Токен бота Telegram для аутентификации.

**Идентификатор чата \***  
1234567890  
Числовой идентификатор чата, куда будут отправляться сообщения.

▼ Скрыть расширенную конфигурацию

**Режим парсинга \***  
HTML  
Строка, определяющая режим парсинга сообщений бота.

**Шаблон сообщения \***  
{{ template "telegram.default.message" . }}  
Строка, содержащая шаблон контента для сообщений бота.

☒ Отправлять уведомления об исправленных оповещениях этому получателю?

☒ Следует ли выполнять переадресацию?

**Метки маршрутизации \***  
Активные оповещения с метками, которые соответствуют всем этим селекторам, будут отправлены на этот приемник. Значения меток могут соответствовать точно или с помощью регулярных выражений.

Имя	Значение
Rule1	new1

☐ Регулярное выражение

● Добавить метку

[Создать](#) [Отмена](#)

### 5. Заполните данные для получателя оповещений:

- Токен бота — имя секрета, содержащего пароль ID Telegram-бота.
- Идентификатор чата (ID Telegram-чата).
- Нажмите на [ **Показать расширенную конфигурацию** ], чтобы отобразить расширенные параметры.
  - В поле **"Режим парсинга"** впишите режим парсинга для сообщений бота. Например режим парсинга — **HTML**.
  - Заполните поле **Шаблон сообщения**.

### 6. В блоке **"Метки маршрутизации"** заполните поля **Имя** и **Значение**. Метки назначаются оповещениям и позволяют управлять их маршрутизацией в Alertmanager.

### 7. Нажмите [ **Создать** ].

Receivers

Вывести Receivers по №... /

[Добавить нового получателя оповещений](#)

ⓘ **Настоящее состояние Receivers**  
Настоящее состояние Receivers, чтобы получить информацию о статусе проблем с вашим кластером.

Имя	Тип интеграции	Метки маршрутизации	
Critical	<a href="#">Настроить</a>		1
Default	<a href="#">Настроить</a>	Все (получатель по умолчанию)	1
New1	telegram	Rule1=new1	1
null			1
Watchdog			1

На странице **Alertmanager** в блоке Receivers появится созданный получатель оповещений с настроенным шаблоном оповещений.



# Мониторинг сертификатов платформы

## 1. Обзор

Под `x509-certificate-exporter` в namespace `nova-monitoring` отвечает за мониторинг сертификатов платформы.

Под проверяет все секреты с типом `--secret-type=kubernetes.io/tls:tls.crt` и собирает информацию о дате окончания действия сертификата.

## 2. Grafana

Метрики с пода `x509-certificate-exporter` выведены в виде отдельного дашборда в Grafana с именем `Certificates Expiration`.

## 3. Оповещения

В Nova Container Platform Platform предварительно настроены оповещения для мониторинга сертификатов.

Весь список оповещений можно найти в разделе **Мониторинг > Правила оповещения**.

Например, оповещения по окончании срока действия сертификатов начнут появляться на главной странице веб-консоли платформы за 14 дней до окончания срока действия сертификата.

Если настроить [AlertManager](#), то оповещения будут приходить выбранным методом. Например, на электронную почту.

## 4. nova-ctl

В `nova-ctl` можно проверить срок действия сертификатов:

```
nova-ctl certs check-expiration
```

BASH | 

Обновить сертификаты:

```
nova-ctl certs renew
```

BASH | 



# Grafana

Grafana — это платформа с открытым исходным кодом для визуализации и анализа данных, которая позволяет создавать динамические и интерактивные дашборды. Она поддерживает интеграцию с различными источниками данных, такими как базы данных, системы мониторинга и облачные сервисы. Grafana позволяет пользователям строить графики, диаграммы и настраивать предупреждения в реальном времени, что делает её популярным инструментом для мониторинга и анализа системных метрик, бизнес-аналитики и других видов данных.

## 1. Основные особенности Grafana

---

- **Интерактивные дашборды:** Возможность создавать настраиваемые панели управления с широким спектром визуальных элементов.
- **Поддержка множества источников данных:** Платформа может подключаться к различным источникам данных, включая Prometheus, Graphite, InfluxDB, Elasticsearch и многим другим.
- **Оповещения и уведомления:** Настройка триггеров для отправки уведомлений при достижении определённых условий.
- **Гибкость и расширяемость:** Поддержка плагинов, которые расширяют функциональность Grafana, добавляя новые типы визуализаций и поддержки дополнительных источников данных.
- **Совместная работа:** Возможность совместного использования дашбордов с коллегами или в больших командах.

## 2. Основные сведения

---

Grafana устанавливается в виде ресурса *Deployment* в Namespace `nova-monitoring`. Для доступа к Grafana из внешней сети установлен ресурс *Ingress*.

В Grafana предустановлен источник Prometheus и дашборды. Данные из этих дашбордов отображаются на странице **Observe > Панели мониторинга** в веб-консоли Nova Container Platform. Данные с других дашбордов не будут выведены.

Рекомендуется добавлять новые дашборды и источники данных через ConfigMap. Grafana автоматически начнёт с ними работать.

## 3. Добавление нового плагина

---

В настоящее время установка плагинов на постоянной основе невозможна. После перезапуска pod'a настройки возвращаются к исходному состоянию. Данный функционал будет реализован в одной из будущих версий платформы.

## 4. Добавление нового источника данных [Datasource]

---

Для добавления нового источника данных создайте ресурс *ConfigMap* в Namespace `nova-monitoring` с именем `nova-grafana-datasources-<имя_источника>` и меткой `nova-platform.io/cluster-dashboard-datasource: 'true'`, например:

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: nova-grafana-datasources-test ①
  namespace: nova-monitoring ②
  labels:
    nova-platform.io/cluster-dashboard-datasource: 'true' ③
data: ④
  datasources_prometheus_new.yaml: |-
    apiVersion: 1
    datasources:
    - name: nova-prometheus-new
      type: prometheus
      url: http://prometheus-operated.nova-monitoring.svc:9090
      isDefault: false
```

YAML | 

1. Имя ресурса: должно быть `nova-grafana-datasources-<имя_источника>`.
2. Имя Namespace: не должно быть изменено.
3. Метка, по которой ресурс автоматически добавляется в Grafana.
4. Пример файла конфигурации. Имя файла должно быть уникально.

## 5. Добавление нового дашборда

---

Для добавления нового дашборда создайте ресурс *ConfigMap* в Namespace `nova-monitoring` с именем `nova-grafana-dashboard-<имя_источника>` и меткой `grafana_dashboard: '1'`, например:

```
kind: ConfigMap
apiVersion: v1
```

YAML | 



```
metadata:
  name: nova-grafana-dashboard-alertmanager-overview ①
  namespace: nova-monitoring ②
  labels:
    grafana_dashboard: '1' ③
data: ④
  alertmanager-overview.json: |-
    {
      "annotations": {
        "list": [
          {
            "builtIn": 1,
            "datasource": {
              "type": "grafana",
              "uid": "-- Grafana --"
            },
            ...
          ]
        }
      }
```

1. Имя ресурса: должно быть `nova-grafana-dashboard-<имя_дашборда>` .
2. Имя Namespace: не должно быть изменено.
3. Метка, по которой ресурс автоматически добавляется в Grafana.
4. Пример файла конфигурации. Имя файла должно быть уникально. Можно заранее настроить дашборд в Grafana и взять его конфигурацию из `JSON Model` .