

# Платформа мониторинга Sentry

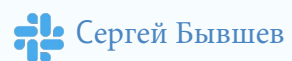


Сергей  
Бывшев



**Сергей Бывшев**

Ведущий инженер автоматизации в "Метр  
квадратный"



---

# План занятия

1. [Введение](#)
2. [Архитектура](#)
3. [Конфигурация](#)
4. [User Interface](#)
5. [Sentry SDK](#)
6. [Итоги](#)
7. [Домашнее задание](#)



# Введение

# Введение



Платформа мониторинга приложений Sentry **помогает диагностировать, исправлять и оптимизировать производительность** своего кода.

- Механизм работы Sentry основывается на real-time мониторинге событий в ПО, их сбору, анализу и нотификации.
- Каждое событие в системе Sentry имеет развёрнутое описание, содержащее метаданные ПО (версия ПО, модуль, stacktrace и т.д.).
- События можно визуализировать, группируя их. Например, по типу или по гео-привязке сервисов.
- Чаще всего отслеживаемыми ошибками являются ошибки приложения.

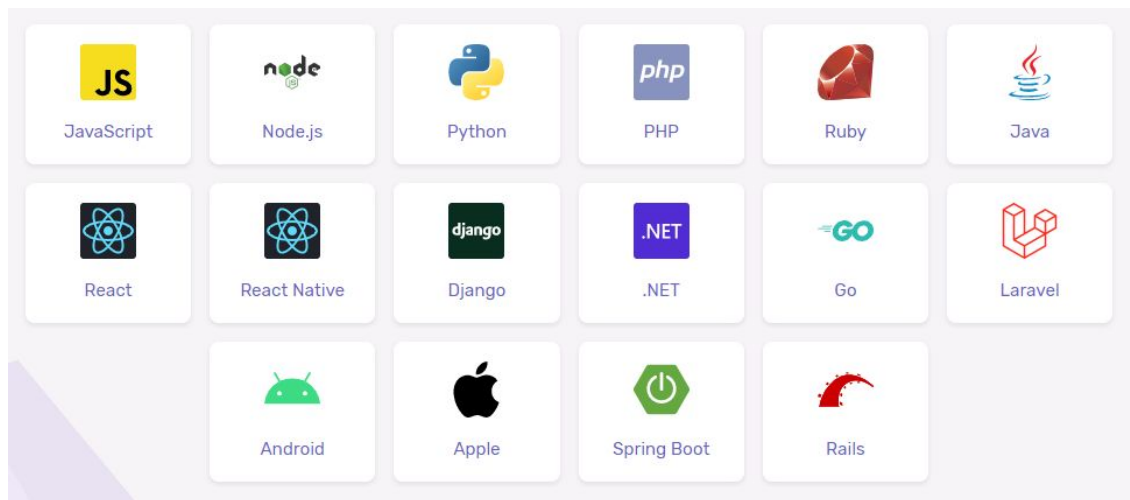
# Введение



Для интеграции Sentry в ПО требуется добавить в код специальный обработчик.

Обработчики поставляются в виде SDK, которые легко добавляются в различные ЯП или фреймворки.

Всего Sentry поддерживает до 93 различных платформ.



Взято с сайта: [sentry.io](https://sentry.io)

# Введение



**Sentry позволяет провести интеграцию с различными системами ведения задач.**

Например, Atlassian Jira.

Можно прямо из события Sentry:

- создать задачу,
- привязать исполнителя,
- автоматически заполнить описание данными события,
- сменить статус задачи

Jira Issue

Create Link

Jira Project

API

Title \*

SyntaxError: Unexpected identifier

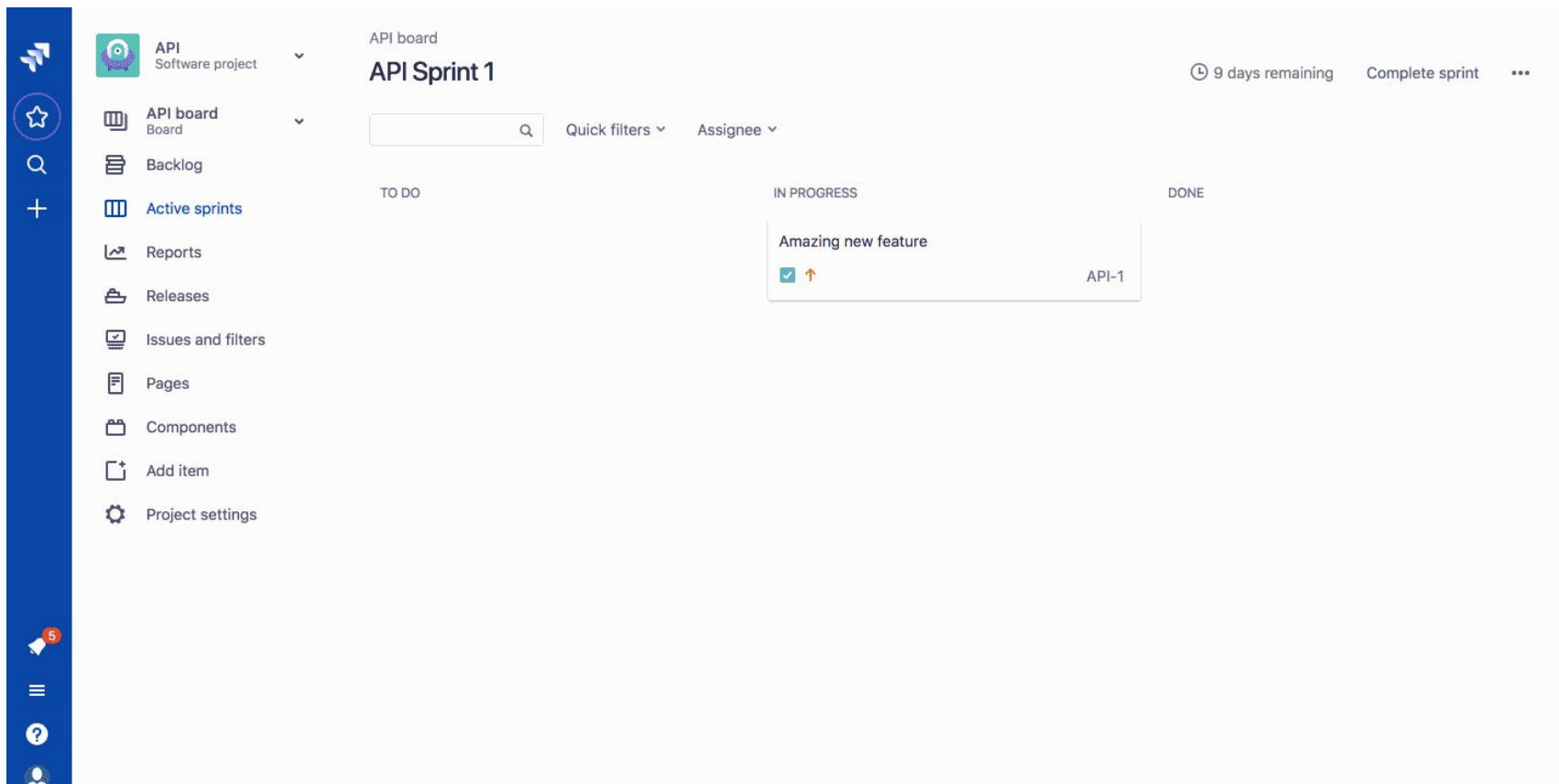
Description

<https://sentry.io/simpsons/backend/issues/630769594/>  

```
{code}
SyntaxError: Unexpected identifier
  at syntaxError (components/Errors.js:47:9)
  at Array.forEach (<anonymous>)
...
(12 additional frame(s) were not displayed)
{code}
```

Взято с сайта: [sentry.io](https://sentry.io)

# Введение



Взято с сайта: [sentry.io](https://sentry.io)



---

# Введение



Одним из главных преимуществ Sentry является **нотификация о событиях**.

Sentry поддерживает множество платформ для нотификации, а также имеет механизм вебхуков для ваших собственных или не обслуживаемых систем.

Нотификации можно настраивать гибким образом, выбирая их **чувствительность, частоту повтора, каналы оповещения**.

# Введение



**ALERT CONDITIONS**

>

WHEN

an event is captured by Sentry and

all

of the following happens

Add optional trigger...

>

IF

all

of these filters match

Add optional filter...

>

THEN

perform these actions

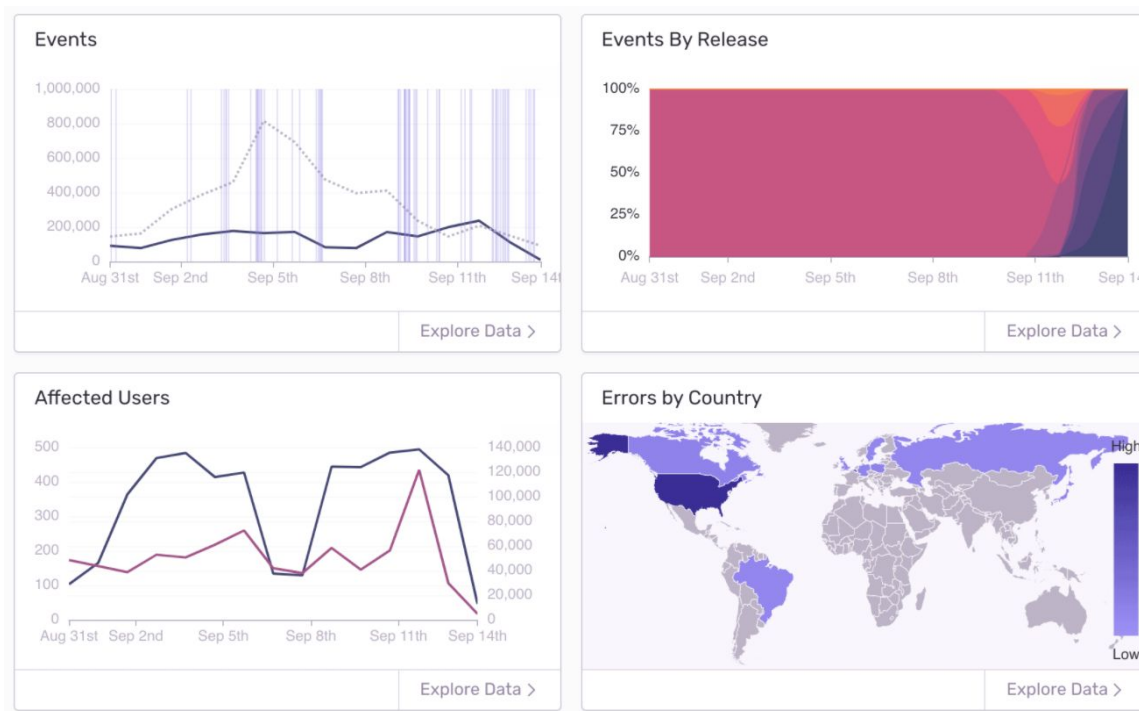
Add action...

Взято с сайта: [sentry.io](https://sentry.io)

# Введение



Таким образом, система Sentry **может заменить собой логирование приложения**, ведь вы всегда будете информированы о событиях, знать их частоту, связанные реплику и версии приложения и т.д.



Взято с сайта: [sentry.io](https://sentry.io)



# Архитектура

---

# Архитектура

**Sentry** представляет из себя веб-приложение, которое поставляется в двух видах:

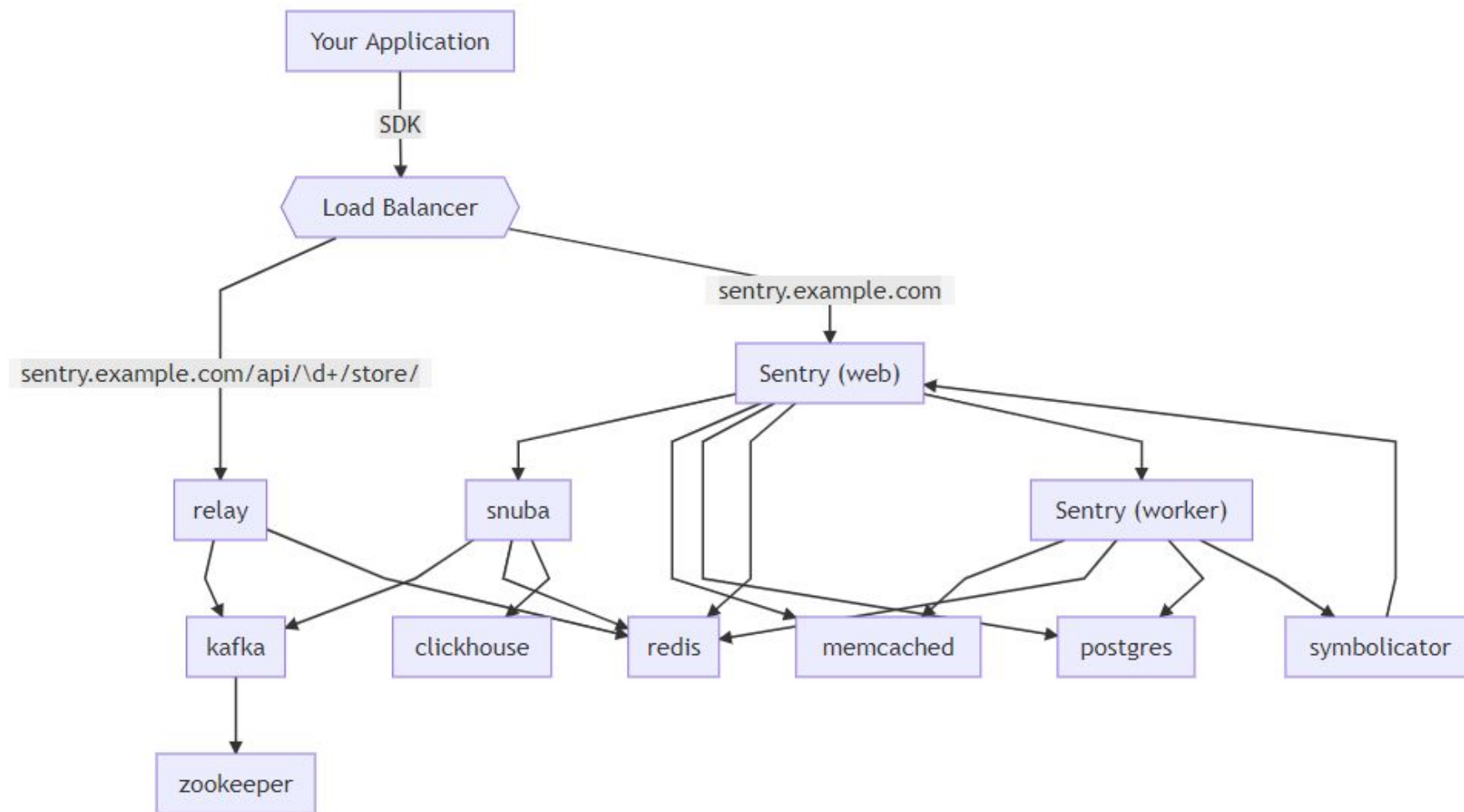
- cloud
- self-hosted

**Cloud-архитектура** размещается на серверах sentry.io и представляет из себя многокомпонентную систему, содержащую:

- Sentry app
- PostgreSQL
- Redis
- Memcached
- Kafka + Zookeeper

Аналогичную Cloud-архитектуру возможно использовать и в self-hosted системе, но это влечет большие затраты по обслуживанию. В то же время, это бесплатное предоставление всех возможностей системы для использования, в соответствии с прикрепленной лицензией.

# Архитектура





# Конфигурация

# Конфигурация

**Конфигурация Sentry** происходит в файлах:

- `config.yml` - настройка основных параметров работы системы
- `sentry.conf.py` - более расширенная настройка системы

**Основную настройку** рекомендуется производить в `config.yml`.

**Если необходимости расширить** настройки или функционал, можно производить настройку в `sentry.conf.py` на ЯП python v3.

Часть настроек также можно производить из `environment` переменных среды.

Данные конфигурационные файлы имеют хорошее внутреннее описание и простую структуру.



# Конфигурация

Пример конфигурации smtp из config.yaml mail сервера для отправки уведомлений:

```
#####  
# Mail Server #  
#####  
  
# mail.backend: 'smtp' # Use dummy if you want to disable email entirely  
mail.host: 'smtp'  
# mail.port: 25  
# mail.username: ''  
# mail.password: ''  
# mail.use-tls: false  
# The email address to send on behalf of  
# mail.from: 'root@localhost'
```

# Конфигурация

Пример конфигурации sentry.conf.py для подключения postgresQL:

```
DATABASES = {  
    "default": {  
        "ENGINE": "sentry.db.postgres",  
        "NAME": "postgres",  
        "USER": "postgres",  
        "PASSWORD": "",  
        "HOST": "postgres",  
        "PORT": "",  
    }  
}
```

# Конфигурация

Пример конфигурации sentry.conf.py для подключения Redis:

```
SENTRY_OPTIONS["redis.clusters"] = {  
    "default": {  
        "hosts": {0: {"host": "redis", "password": "", "port": "6379", "db": "0"}}  
    }  
}
```

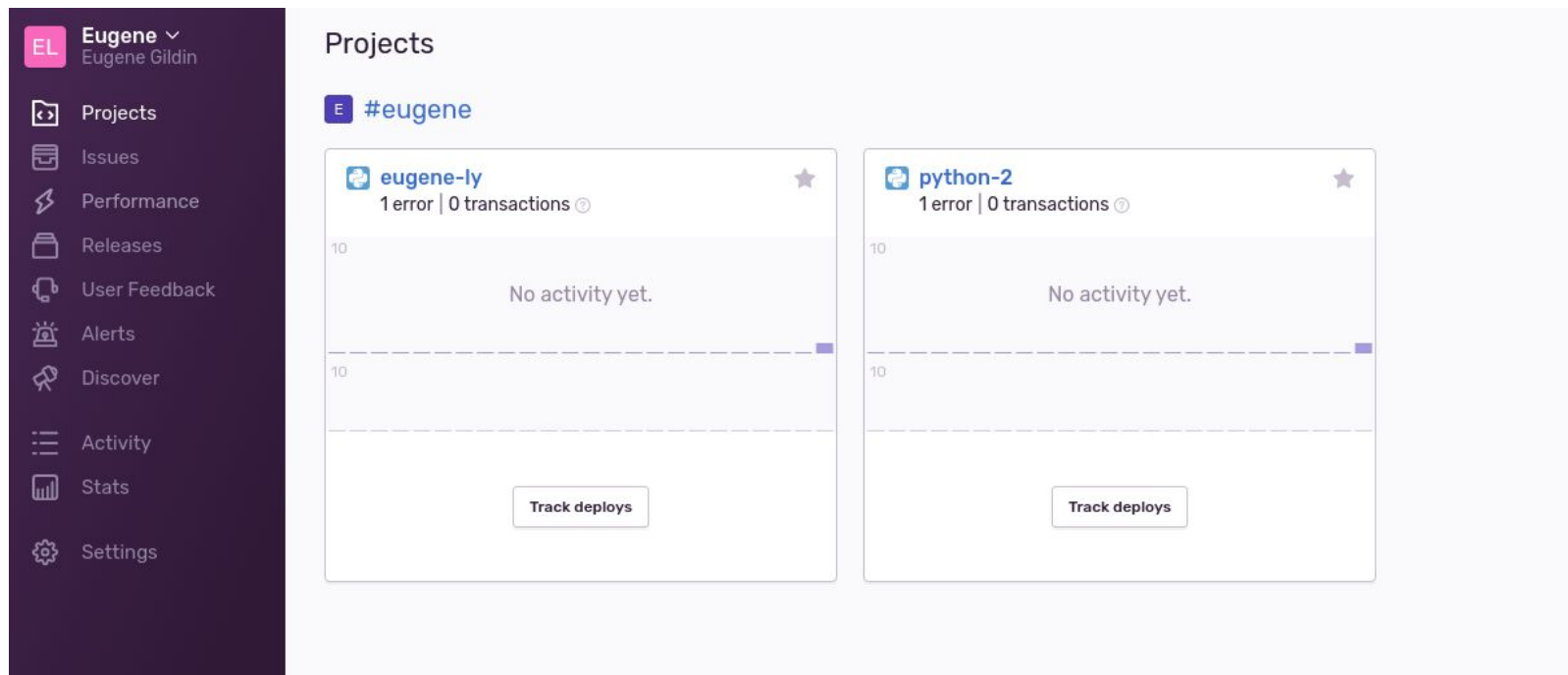


# User interface

# User interface

UI предоставляет огромный набор возможностей для гибкой настройки обработки и мониторинга входящих событий. Мы рассмотрим лишь основные.

Основной dashboard выглядит следующим образом:



# User interface

Переходя в проект, мы видим непосредственно зарегистрированные события и некоторую статистику:

The screenshot shows the User Interface for the 'eugene-ly' project. The interface is divided into a sidebar on the left and a main content area on the right.

**Sidebar:**

- EL Eugene Gildin
- Projects
- Issues
- Performance
- Releases
- User Feedback
- Alerts
- Discover
- Activity
- Stats
- Settings

**Main Content Area:**

Top bar: eugene-ly | All Environments | Last 14 days

Issues (1) | Sort by: Last Seen | Unresolved Issues | Search: is:unresolved

Issue details:

- ☐ Resolve | ☐ Ignore | Merge | ... | Play
- ☐ This is an example Python exception  
raven.scripts.runner in main
- ☒ EUGENE-LY-1 | 20 minutes ago — 20 minutes old

Graph: 24h 14d | EVENTS: 1 | USERS: 1 | ASSIGNEE: [User Icon]

# User interface

Переходя в событие мы видим множество данных, связанных с ним и возможности интерактивного взаимодействия, как привязка к исполнителю для решения или проставления признака, что событие разрешено:

This is an example Python exception

raven.scripts.runner in main

Resolve

Ignore

Share

Details

Activity 0

User Feedback 0

Attachments

Tags

Events

Merged Issues

Similar Issues

ISSUE #

EUGENE-LY-1

EVENTS

1

USERS

1

ASSIGNEE

Event 457897c95dba4538b08ca1e977aa953d

Feb 19, 2021 7:20:01 AM UTC | JSON (7.7 KB)

Older

Newer

S

sentry@example.com

ID: 1

Chrome

Version: 28.0.1500

Windows

Version: 8

TAGS

browser

Chrome 28.0.1500

browser.name

Chrome

client\_os

Windows 8

client\_os.name

Windows

environment

prod

level

error

server\_name

web01.example.org

uri

http://example.com/foo

user

id:1

MESSAGE

This is an example Python exception

STACK TRACE (most recent call first)

Full

Raw

raven/scripts/runner.py in main at line 112

raven/scripts/runner.py in send\_test\_message at line 77

raven/base.py in captureMessage at line 577

raven/base.py in capture at line 459

raven/base.py in build\_msg at line 303

Ownership Rules

Create Ownership Rule

All Environments

LAST 24 HOURS

LAST 30 DAYS

LAST SEEN 23 minutes ago

FIRST SEEN 23 minutes ago

RELEASES NOT CONFIGURED

Setup Releases to make issues easier to fix.

Linked Issues

Set up Issue Tracking

Tags

browser

Chrome 28.0.1500

100%

23

# User interface

Следующая важная сущность - это алёрты. Dashboard для алёртов выглядит следующим образом:

The screenshot shows a web application interface for managing alerts. On the left is a dark purple sidebar with a user profile 'Eugene Gildin' and a list of navigation items: Projects, Issues, Performance, Releases, User Feedback, Alerts (highlighted), Discover, Activity, Stats, and Settings. The main content area has a header with 'python-2' and 'All Environments'. Below this is the 'Alerts' section with a 'Create Alert Rule' button. A table titled 'Alert Rules' contains one entry: 'Test alert' of type 'ISSUE' for the 'python-2' project, created by 'Eugene Gildin' on 'Feb 19, 2021'. The table has columns for Type, Alert Name, Project, Created By, Created, and Actions. The Actions column contains icons for deleting and configuring the alert rule.

TYPE	ALERT NAME	PROJECT	CREATED BY	CREATED ↓	ACTIONS
ISSUE	<a href="#">Test alert</a>	python-2	Eugene Gildin	Feb 19, 2021	



# User interface

Если мы перейдем непосредственно в алёрт - мы увидим меню его настройки, где сможем всегда его отредактировать.

### Edit Alert Rule: Test alert

ALERT SETUP

Environment

Choose an environment for these conditions to apply to

All Environments

Alert name \*

Add a name for this alert

Test alert

ALERT CONDITIONS

1

WHEN

an event is captured by Sentry and

all

of the following happens

Add optional trigger...

2

IF

all

of these filters match

Add optional filter...

3

THEN

perform these actions

Add action...

RATE LIMIT

Action Interval \*

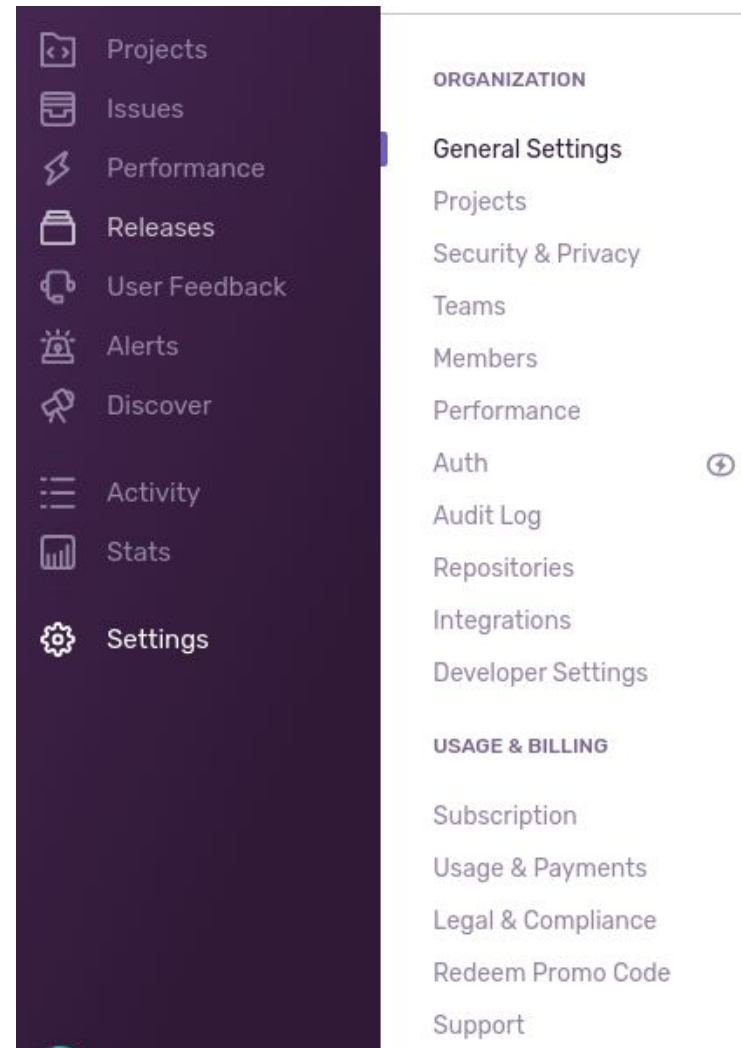
Perform these actions once this often for an issue

5 minutes

# User interface

Все настройки, как например интеграции или ACL - мы можем произвести в боковом меню Settings.

Если у вас есть свой pet-project в виде веб сервиса или приложения - вы можете попробовать подключить его к Sentry, используя SDK и поэкспериментировать с отображаемыми в Sentry данными.





# Sentry SDK

# Sentry SDK

Рассмотрим на примере настройку python приложения для отправки событий в Sentry.

- установим необходимую библиотеку с SDK:

```
pip install sentry-sdk
```

- во входной точке ПО импортируем sentry SDK, свяжем с библиотекой логирования logging и произведем настройку отлова событий:

```
import logging
import sentry_sdk
from sentry_sdk.integrations.logging import LoggingIntegration

# All of this is already happening by default!
sentry_logging = LoggingIntegration(
    level=logging.INFO,          # Capture info and above as breadcrumbs
    event_level=logging.ERROR    # Send errors as events
)
```

# Sentry SDK

Таким образом, мы установили отлов событий типа ERROR и отправку их в Sentry. Также в дополнительной информации к событиям ERROR, в Sentry будет отправлена краткая информация о событиях уровня INFO для диагностики.

Осталось только запустить процесс обработки событий и отправки их в Sentry.

Для этого во входной точке приложения добавляем следующий блок:

```
sentry_sdk.init(  
    dsn="https://examplePublicKey@o0.ingest.sentry.io/0",  
    integrations=[sentry_logging]  
)
```

Параметр **dsn** - предоставляется в UI системы Sentry при создании проекта.

Параметры **integrations** - это настройки SDK, которые мы произвели ранее.

# Sentry SDK

Рассмотрим, что теперь произойдет при следующей очередности событий:

```
import logging
logging.debug("I am ignored")
logging.info("I am a breadcrumb")
logging.error("I am an event", extra=dict(bar=43))
logging.exception("An exception happened")
```

- Зарегистрируется событие error “I am an event”
- Событие INFO “I am a breadcrumb” добавится к событию ERROR как диагностическое
- bar=43 добавится в событие как параметр
- Событие EXCEPTION “An exception happend” зарегистрируется как Stacktrace с полным списком системных вызовов
- Событие DEBUG “I am ignored” будет проигнорировано и никак не обработается



# Итоги

---

# Итоги

В данной лекции мы узнали:

- Что такое Sentry
- Возможности данного инструмента
- Как выглядит self-hosted и cloud архитектура
- Рассмотрели какими путями можно сконфигурировать Sentry и изучили некоторые настройки
- Ознакомились с основными элементами интерфейса Sentry
- Увидели как интегрировать ПО с системой Sentry



---

## Домашнее задание

Давайте посмотрим ваше [домашнее задание](#).

- Вопросы по домашней работе задавайте **в чате** мессенджера Slack.
- Задачи можно сдавать **по частям**.
- Зачёт по домашней работе проставляется после того, как **приняты все задачи**.

**Задавайте вопросы и  
пишите отзыв о лекции!**

**Сергей Бывшев**



Сергей Бывшев