нетология

Платформа мониторинга Sentry

Сергей
Бывшев

Сергей Бывшев

Ведущий инженер автоматизации в "Метр квадратный"

Сергей Бывшев

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.

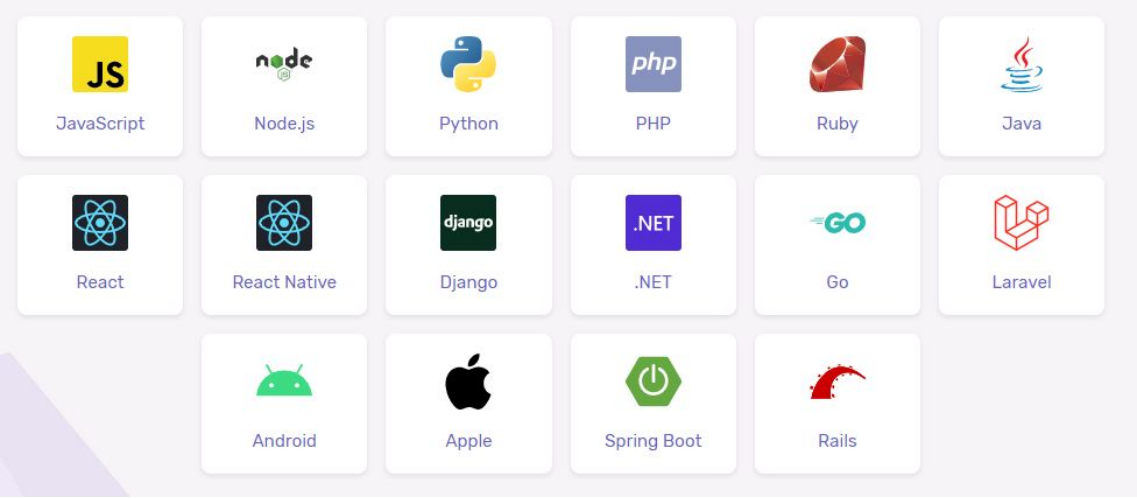
- Механизм работы Sentry основывается на real-time мониторинге событий в ПО, их сбору, анализу и нотификации.
- Каждое событие в системе Sentry имеет развёрнутое описание, содержащее метаданные ПО (версия ПО, модуль, stacktrace и т.д.).
- События можно визуализировать, группируя их. Например, по типу 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

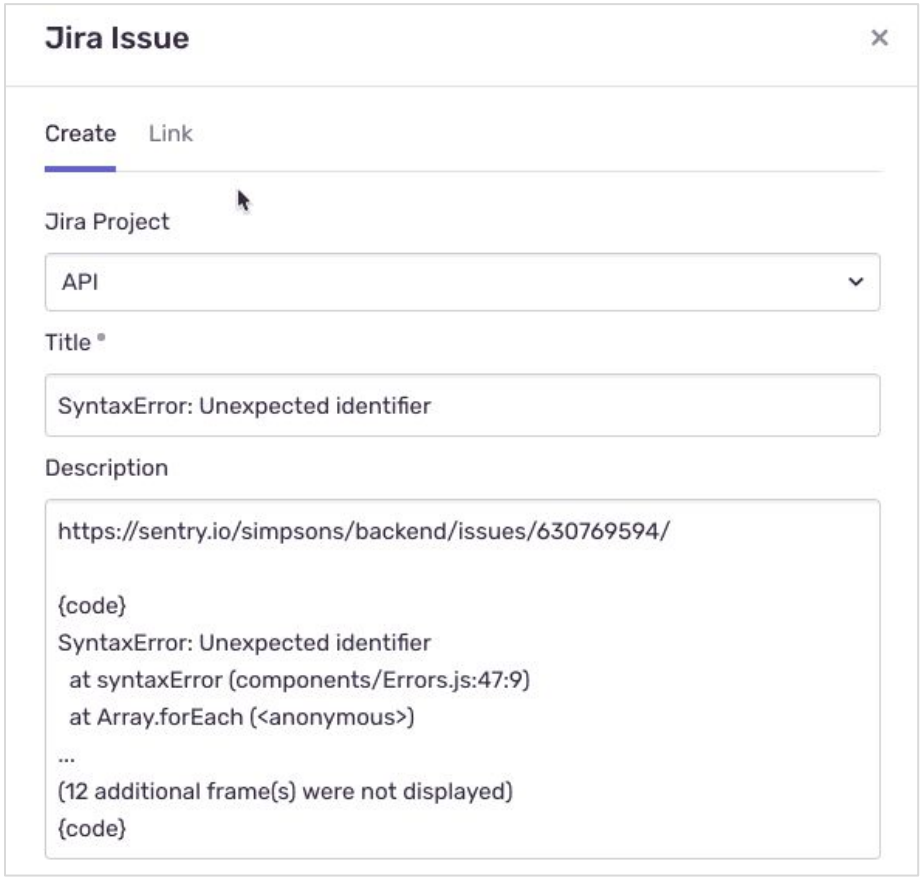
Введение

Sentry позволяет провести интеграцию с различными системами ведения задач.

Например, Atlassian Jira.

Можно прямо из события Sentry:

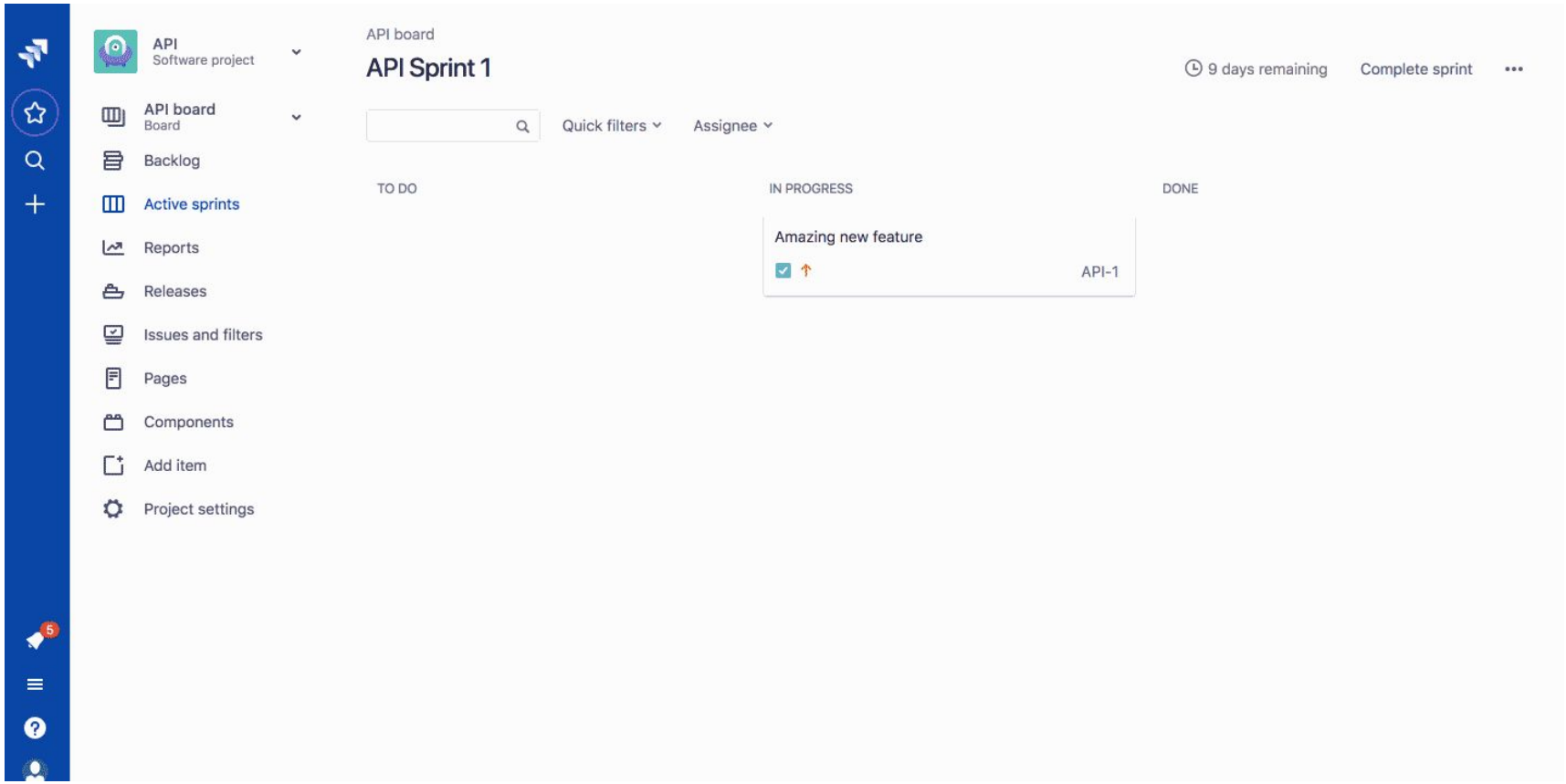
- создать задачу,
- привязать исполнителя,
- автоматически заполнить описание данными события,
- сменить статус задачи



Взято с сайта: sentry.io

Введение

SENTRY



Взято с сайта: sentry.io

Введение

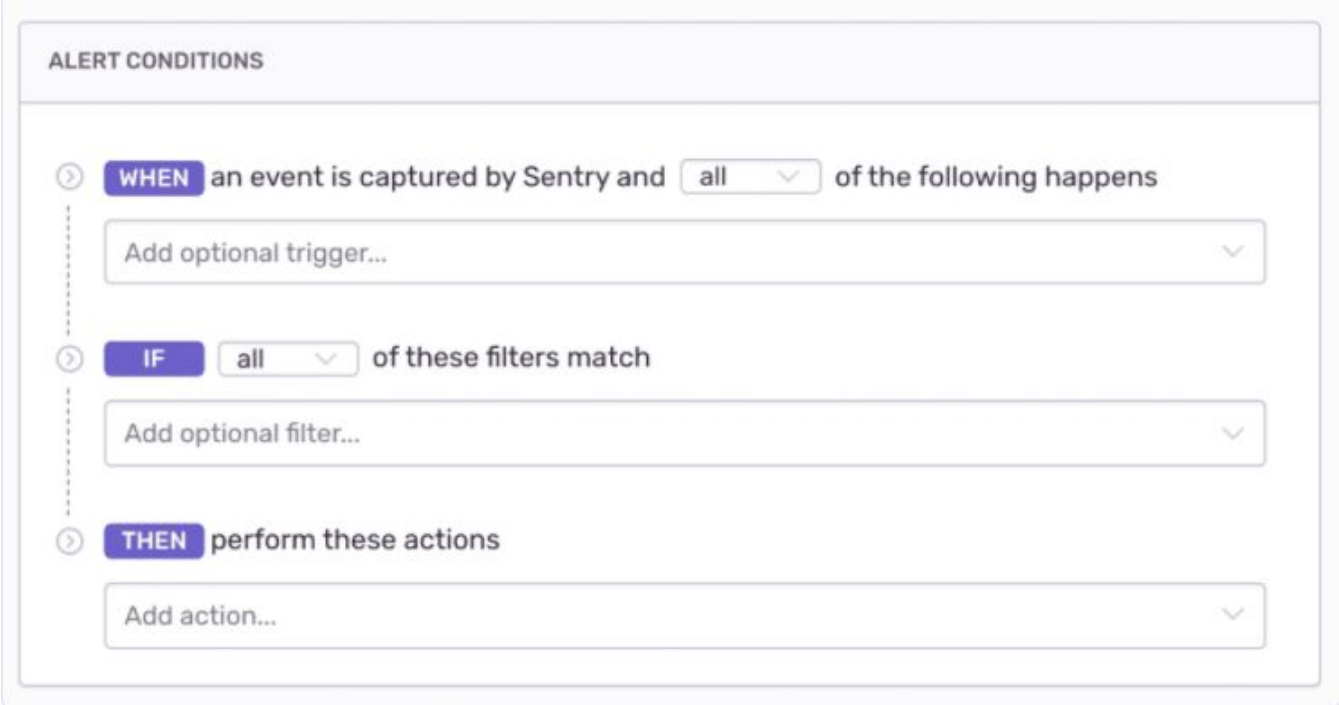
Одним из главных преимуществ Sentry является **нотификация о событиях**.

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

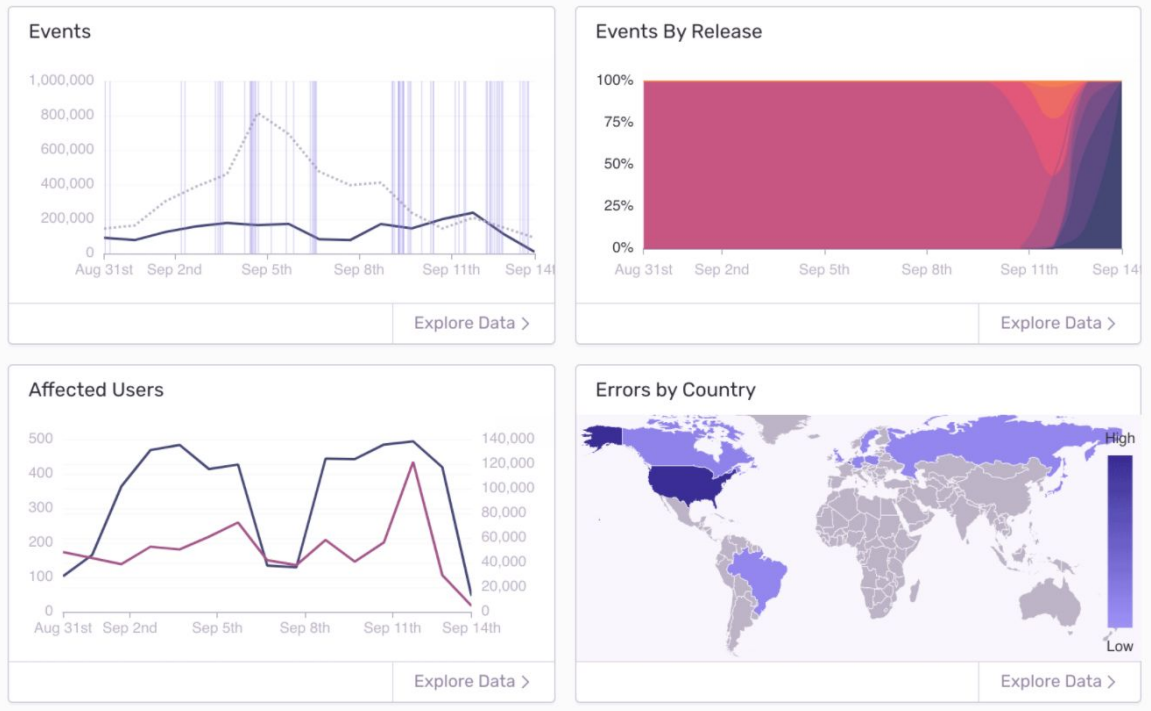
SENTRY



Взято с сайта: 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

Архитектура

Архитектура

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

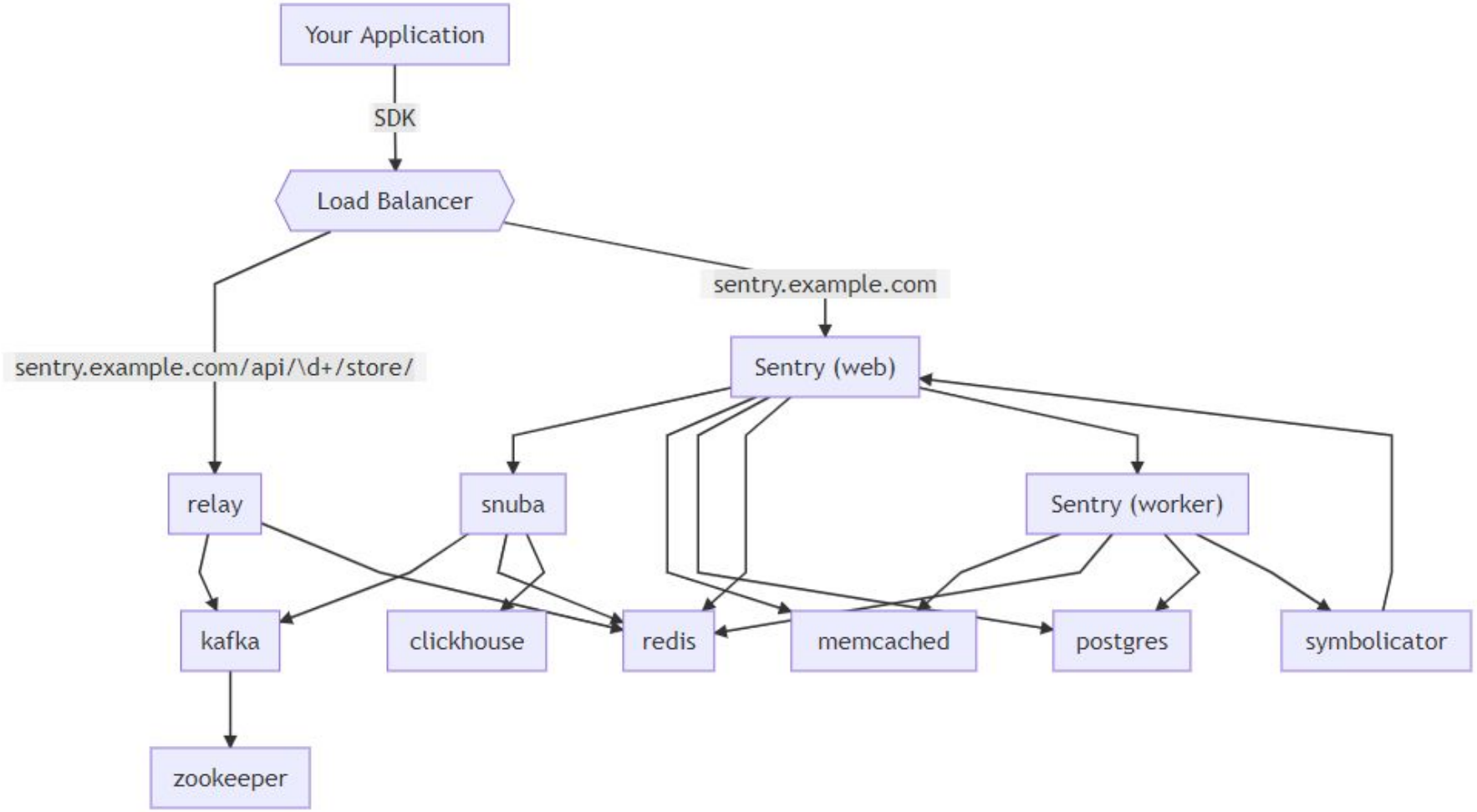
- cloud
- self-hosted

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

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

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



Взято с сайта: sentry.io

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 нотификаций:

```
###############
# 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:

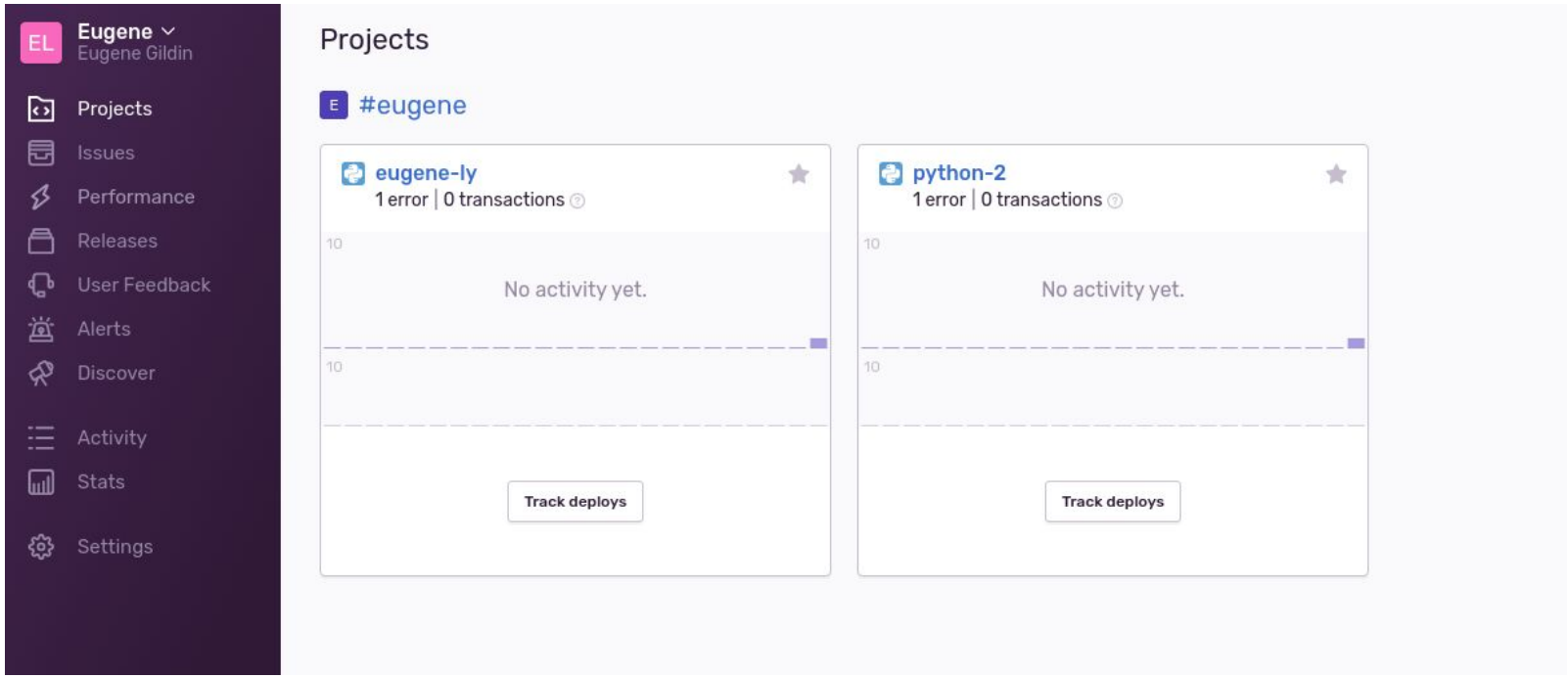
```
SENTRY_OPTIONS["redis.clusters"] = {
    "default": {
        "hosts": {0: {"host": "redis", "password": "", "port": "6379", "db": "0"}}
    }
}
```

User interface

User interface

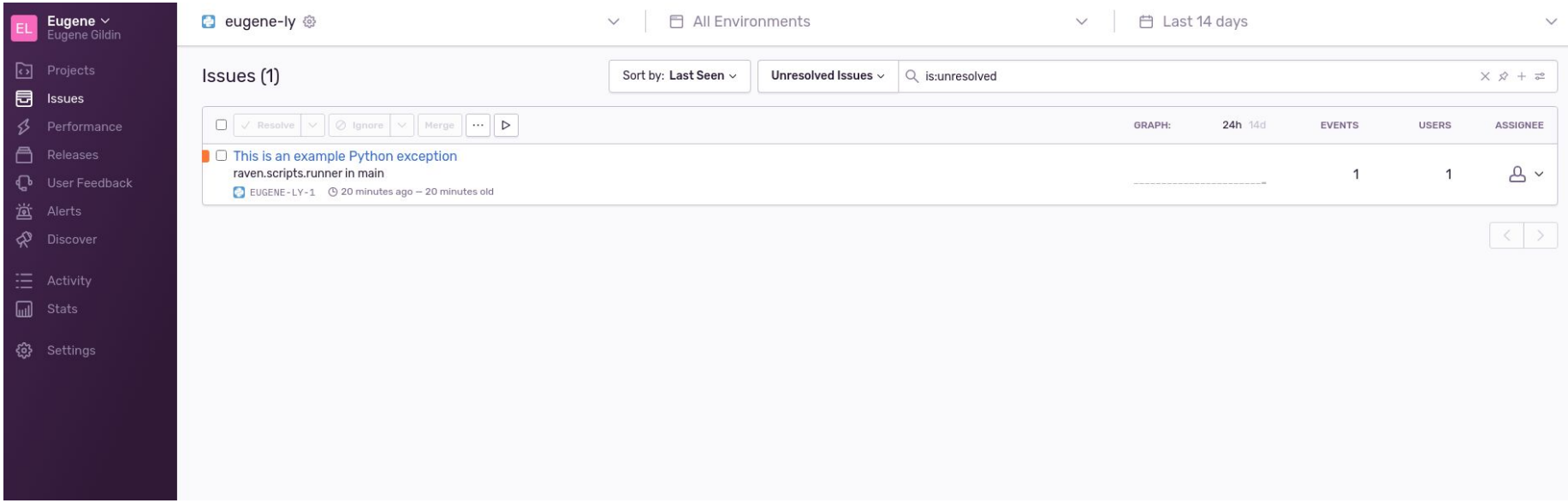
UI предоставляет огромный набор возможностей для гибкой настройки обработки и мониторинга входящих событий. Мы рассмотрим лишь основные.

Основной dashboard выглядит следующим образом:



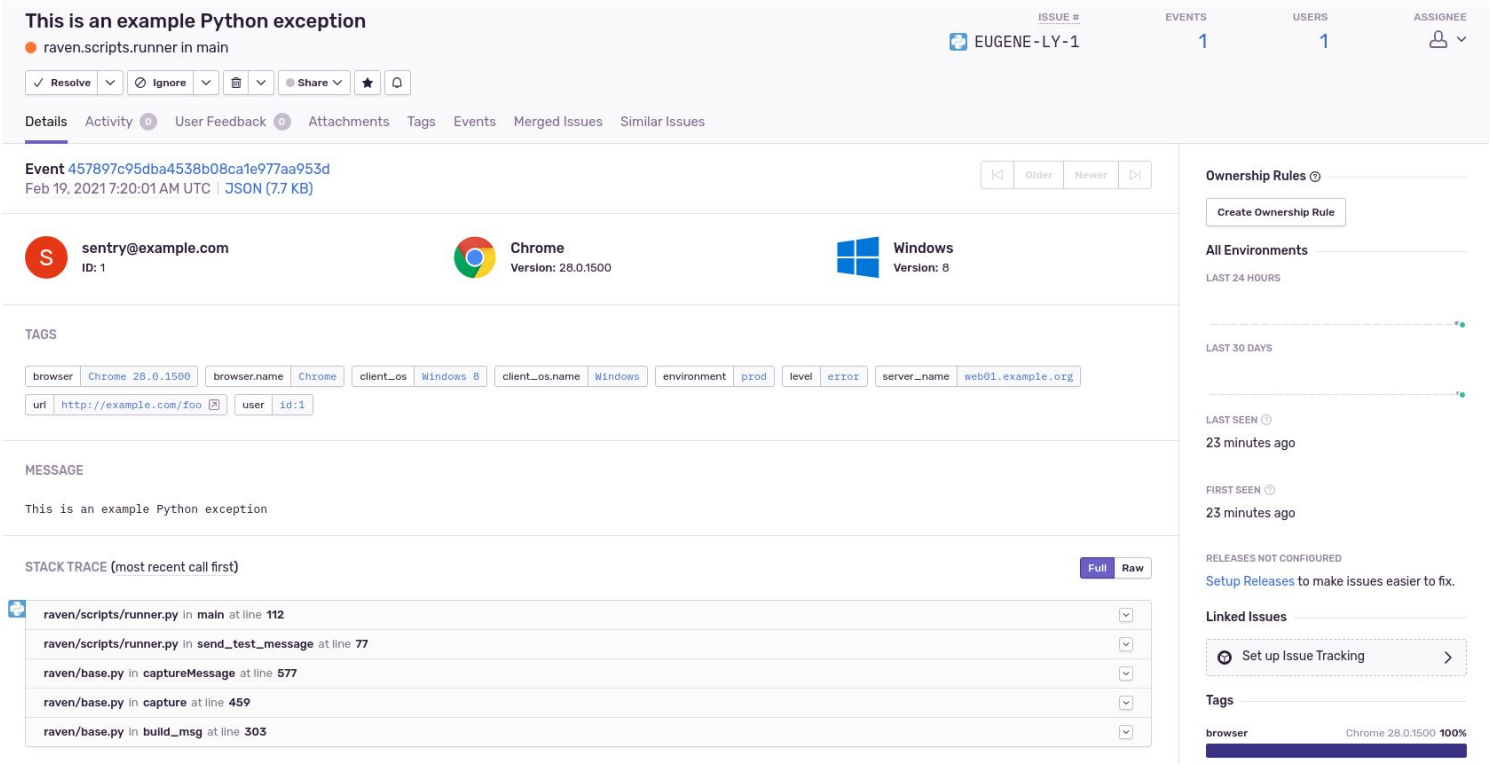
User interface

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



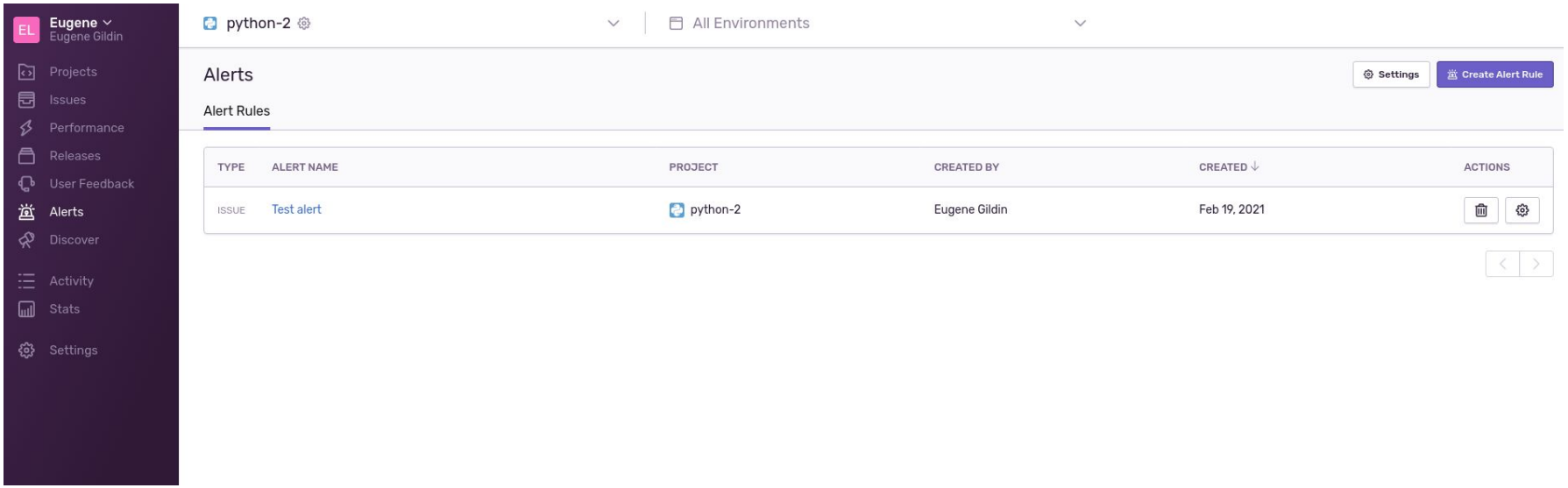
User interface

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:



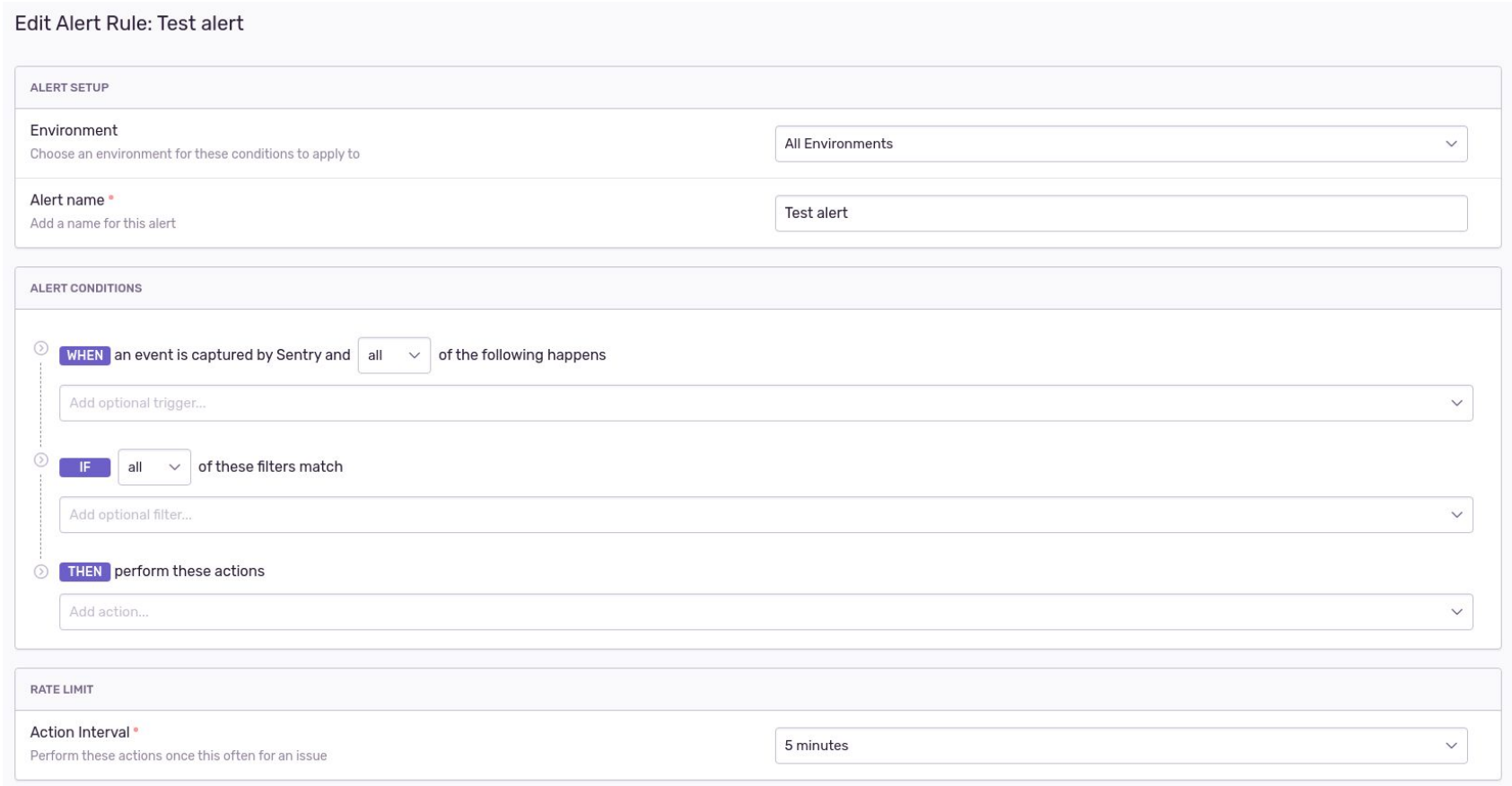
User interface

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



User interface

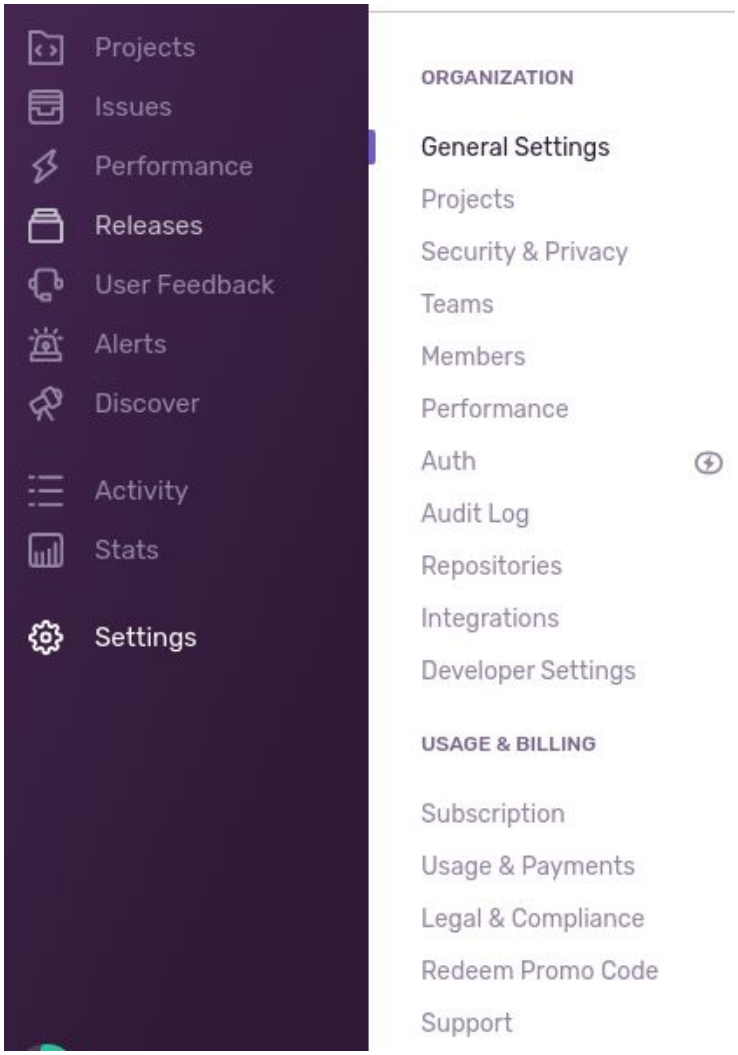
Если мы перейдем непосредственно в алёрт - мы увидим меню его настройки, где сможем всегда его отредактировать.



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 b 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**.

нетология

Задавайте вопросы и пишите отзыв о лекции!

Сергей Бывшев

Сергей Бывшев