Rust at Sentry

Armin @mitsuhiko Ronacher
Hi, I'm Armin.

... and I do Open Source,
lots of Python and SaaS...
... and here is where you can find me

twitter.com/@mitsuhiko

github.com/mitsuhiko

lucumr.pocoo.org/
we show you your crashes
<table>
<thead>
<tr>
<th>Error</th>
<th>Message</th>
<th>DATE</th>
<th>EVENTS</th>
<th>USERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>SalesforceError</td>
<td>salesforce.apply.update</td>
<td>SENTRY-3M</td>
<td>1.8m</td>
<td>0</td>
</tr>
<tr>
<td>ExpiredSignatureError</td>
<td>/plugins/jira-ac/plugin</td>
<td>SENTRY-120</td>
<td>874k</td>
<td>24k</td>
</tr>
<tr>
<td>PluginError</td>
<td>sentry.tasks.commits.fetch.commits</td>
<td>SENTRY-3B</td>
<td>900</td>
<td>0</td>
</tr>
<tr>
<td>ValueException</td>
<td>api/project.jbl/store/</td>
<td>SENTRY-3H</td>
<td>9.4k</td>
<td>12</td>
</tr>
<tr>
<td>PluginError</td>
<td>sentry.tasks.commits.fetch.commits</td>
<td>SENTRY-3V</td>
<td>21</td>
<td>0</td>
</tr>
<tr>
<td>Exception</td>
<td>sentry.tasks.check_auth_identity</td>
<td>SENTRY-S1</td>
<td>3.2k</td>
<td>0</td>
</tr>
</tbody>
</table>
KeyError
'project'

sentry/core.py in insert_data_to_database at line 783
778.
779. def insert_data_to_database(self, data, from_reprocessing=False):
780. # we might be passed LazyData
781. if isinstance(data, LazyData):
782. data = dict(data.items())
783. cache_key = '{}:0'.format(data['project'], data['event_id'])
784. default_cache.set(cache_key, data, timeout=3600)
785. task = from_reprocessing and \
786. preprocess_event_from_reprocessing or preprocess_event
787. task.delay(cache_key=cache_key, start_time=time(),
788. event_id=data['event_id'])

data = {}
from_reprocessing = True

sentry/tasks/reprocessing.py in reprocess_event at line 35
sentry/tasks/base.py in __wrapped at line 54
Called from: celery/app/py in __protected_call_
chapter 0

a bit about us
“sentry”
Open Source Project
“sentry.io”

Cloud Hosted
relatively flexible stack
postgres, mysql, cassandra, riak, …
chapter 1

our general tech stack
Lots of Python
Simple Stack

Python
RabbitMQ
Postgres
Riak
Redis
Conservative Approach to Adding New Systems
prefer Internal Modularity over building More Services
how does rust fit?
why is there Rust?
a bit of a hobby of mine
turned into **sentry-cli**

(command line client to manage organizations / build integration)
later reused components for Python modules
now part of our tech stack
chapter 3

how we started
why we use Rust for sentry-cli
$ otool -L which sentry-cli
/usr/local/bin/sentry-cli:
/System/Library/Frameworks/Security.framework/Versions/A/Security
/usr/lib/libiconv.2.dylib
/System/Library/Frameworks/CoreFoundation.framework/Versions/A/CoreFoundation
/usr/lib/libcurl.4.dylib
/usr/lib/libz.1.dylib
/usr/lib/libSystem.B.dylib
$ curl -sL https://sentry.io/get-cli/ | bash

$ npm install sentry-cli-binary
that and crates.io
rejected alternatives

- go
- JavaScript *(bundled node)*
- C / C++
- Python / Ruby
things that work well

serialization / deserialization
error handling
console UX
file size
chapter 4
from client to server
sourcemaps parsing
reused sentry-cli code for server
processing time

+20 sec $\rightarrow < 500$ms
debug symbols
proguard
chapter 5
marrying python and rust
from setuptools import setup, find_packages

classifiers = [
    "Development Status :: 5 - Production/Stable",
    "Intended Audience :: Developers",
    "License :: OSI Approved :: MIT License",
    "Operating System :: OS Independent",
    "Programming Language :: Python :: 3.6",
    "Programming Language :: Python :: 3.7",
    "Programming Language :: Python :: 3.8",
    "Programming Language :: Python :: 3.9",
    "Topic :: Software Development :: Build Tools",
]

setup(
    name='example',
    version='0.0.1',
    packages=find_packages(),
    install_requires=[
        'snaek',
    ],
    classifiers=
    classifiers,
)
build rust library
expose rust -> cabi
snake to consume
pub struct Point {
    pub x: f32,
    pub y: f32,
}

#[no_mangle]
pub unsafe extern "C" fn example_get_origin() -> Point {
    Point { x: 0.0, y: 0.0 }
}
from example._native import lib

def get_origin():
    point = lib.example_get_origin()
    return (point.x, point.y)
chapter 6

things in rust to love
#[derive(Serialize, Deserialize, Debug, Default)]
pub struct Deploy {
    #[serde(rename="environment")]
    pub env: String,
    pub name: Option<String>,
    pub url: Option<String>,
    #[serde(rename="dateStarted")]
    pub started: Option<DateTime<UTC>>,  
    #[serde(rename="dateFinished")]
    pub finished: Option<DateTime<UTC>>,
}

/// Lists all deploys for a release
pub fn list_deploys(&self, org: &str, version: &str) -> ApiResult<Vec<Deploy>>
{
    self.get(&format!("/organizations/{}/releases/{}/deploys/",
                        PathArg(org), PathArg(version))?)
        .convert()
}
super flexible
(de)serialization layer
chapter 7

crates we use / built
always useful

if_chain

lazy_static
error-chain
for serialization

serde
serde_derive
serde_json
serde_yaml
for pretty console UI

console
indicatif
dialoguer

ours :)
HTTP stuff

rust-curl

openssl-probe
chapter 8

things we don't like
compile times :'( 
“what not to do” guide
incremental compilation
higher level C ABI tools
Questions?