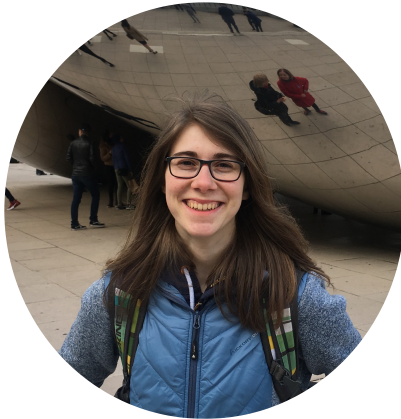


Rust in Mozilla's Data Platform

Vancouver Rust Meetup

Anna Scholtz | 2021-06-16

Hello 🖐️

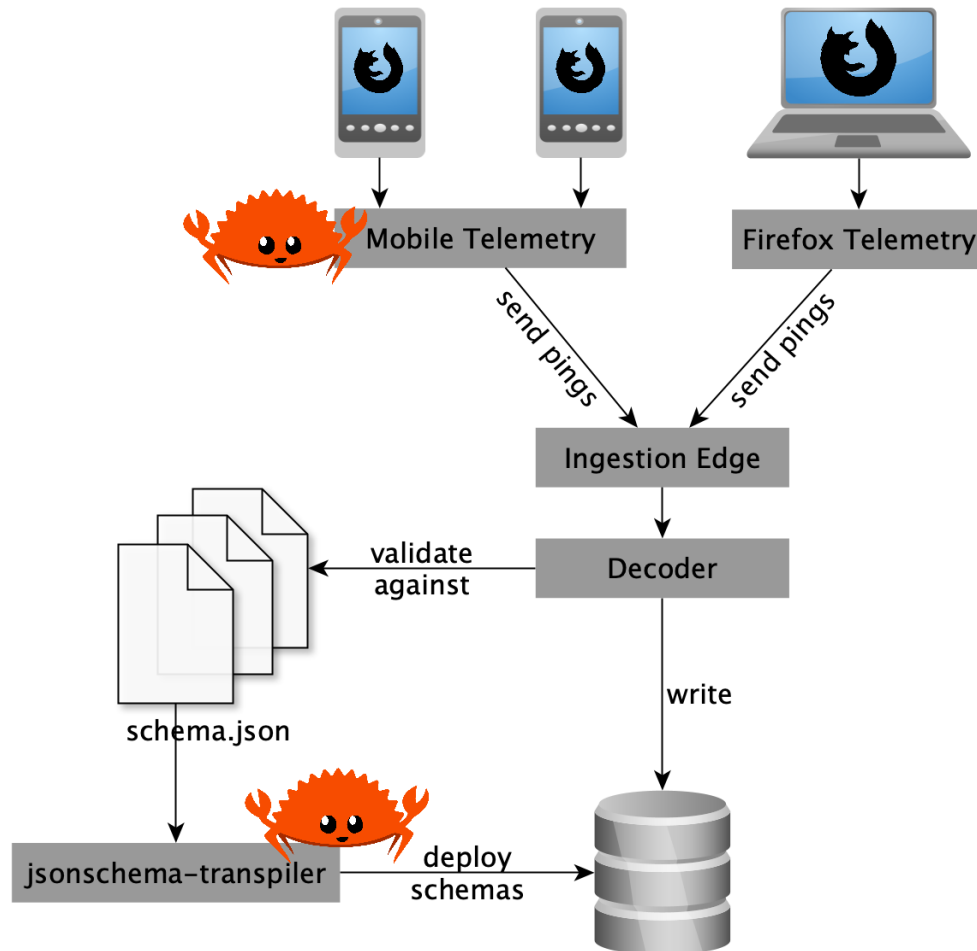


Anna Scholtz

Data Engineer @ Mozilla

@scholtzan

Where is Rust used?



Why is data collected?

Measurement Dashboard

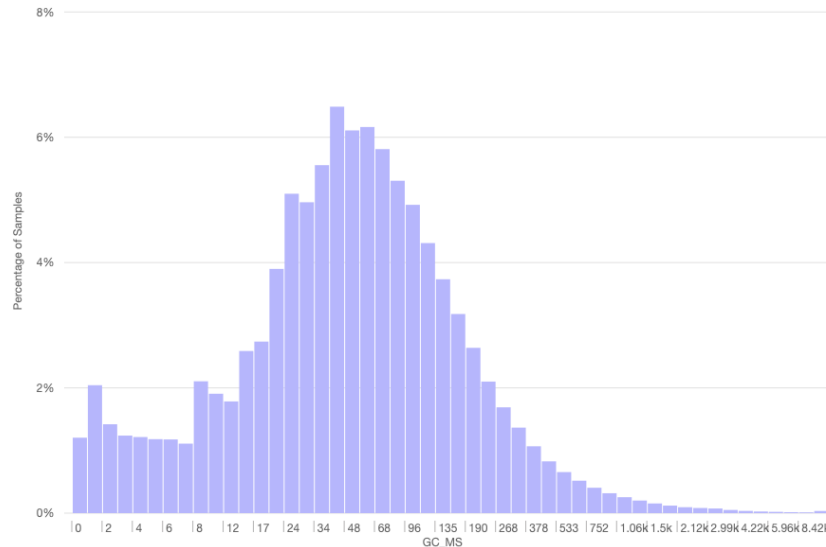
[Get Shortlink](#)[Report bug](#)[Telemetry Portal](#)[Usage Tutorial](#)[Login](#)

Partial outage of release data from 2018-12-17 to present. We are experiencing issues in the underlying dataset that results in the Measurement Dashboards showing no data after December 17 for release data only. We are sorry for the inconvenience. Please watch [Bug 1517018](#) for updates.

[Distribution](#)[Evolution](#)

GC_MS distribution for **Firefox Desktop** **nightly 90**, on **any OS (61)** **any architecture (3)** with **any process** and compare by **none**

Time spent running JS GC (ms) ⓘ More details



Histogram Type	exponential
Ping Count	7.29M
Sample Count	1.96B
Sample Sum	683.75B
Number of dates	30
Selected Dates	2021/04/19 to 2021/05/19

5th Percentile	3.14
25th Percentile	23.14
Median	51.21
75th Percentile	110.07
95th Percentile	376.22

Notice percentiles are estimated based on values in the histogram. Values are only guaranteed to be accurate to the nearest bucket.

[Export CSV](#)[Export JSON](#)

<https://telemetry.mozilla.org/new-pipeline/dist.html>

Lean Data Practices



Stay Lean

Decide if all your data collection delivers value.



Build Security

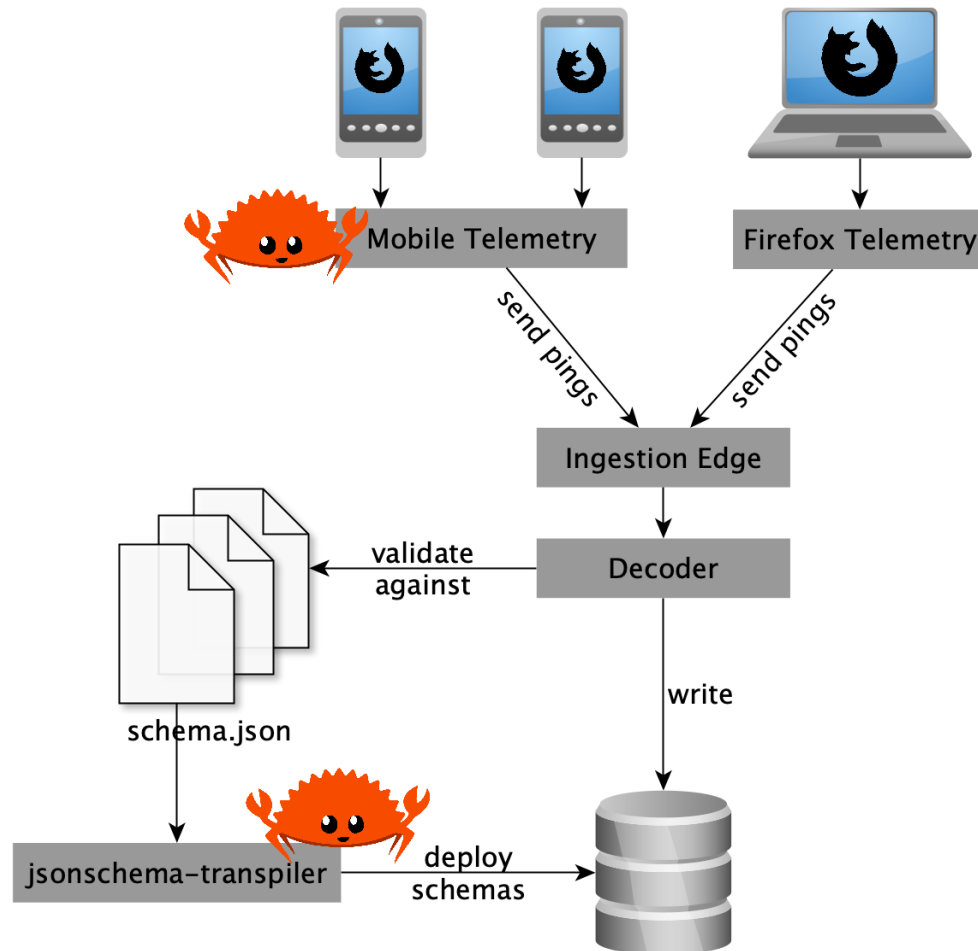
Learn how to protect customer data.



Engage Your Users

Keep customers informed and empowered.

Where is Rust used?





Telemetry library that performs measurements and sends data from Firefox products



Jan-Erik Rediger

Mozilla

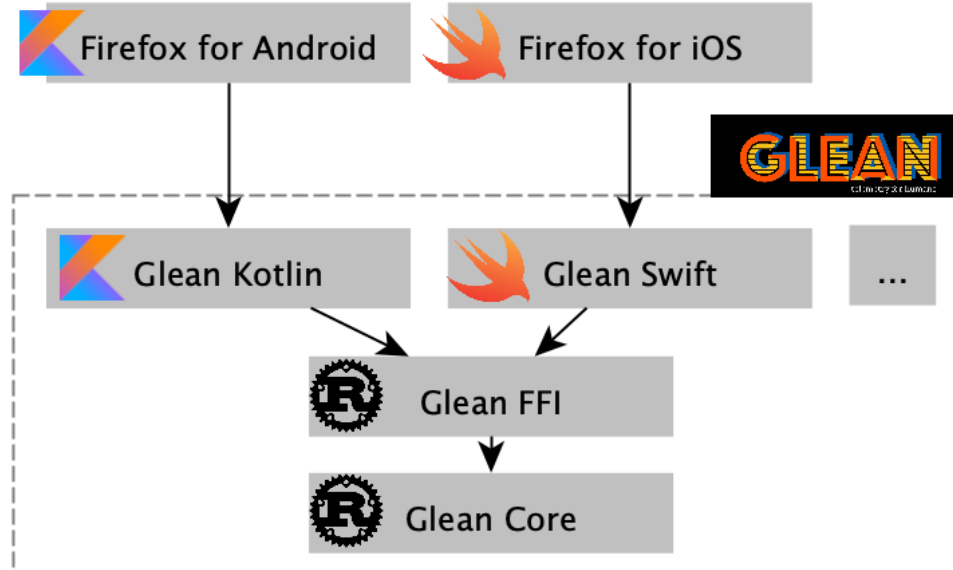
Leveraging Rust to build
cross-platform mobile
libraries

rust-linz.at



<https://www.youtube.com/watch?v=peu-rtN4358>

Glean SDK



Glean Core

```
1 #[derive(Debug)]
2 pub struct Glean {
3     upload_enabled: bool,
4     data_store: Option,
5     event_data_store: EventDatabase,
6     core_metrics: CoreMetrics,
7     additional_metrics: AdditionalMetrics,
8     // ...
9 }
```

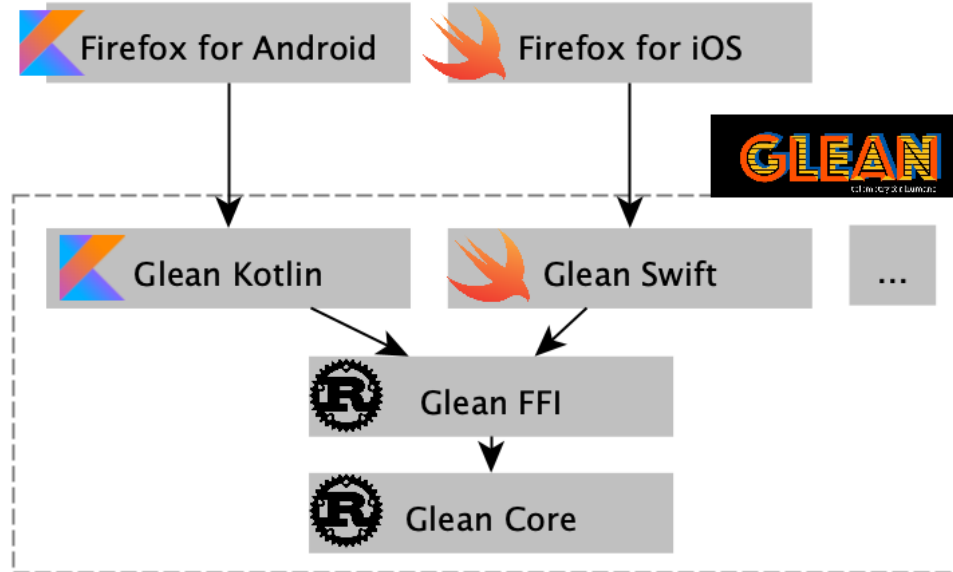
<https://github.com/mozilla/glean/blob/main/glean-core/src/lib.rs>

Metric Types

```
1 pub struct CounterMetric {  
2     meta: CommonMetricData,  
3 }  
4  
5 impl CounterMetric {  
6     pub fn add(&self, glean: &Glean, amount: i32) {  
7         glean  
8             .storage()  
9             .record_with(  
10                 glean,  
11                 &self.meta,  
12                 old_value.add(amount)  
13             )  
14     }  
15 }
```

<https://github.com/mozilla/glean/blob/main/glean-core/src/metrics/counter.rs>

Glean SDK



Glean FFI

FFI - Foreign Function Interface

Connector to lower-level Rust crate

Glean FFI

```
1 #[no_mangle]
2 pub unsafe extern "C" fn glean_initialize(cfg: *const FfiConfiguration) -> u8 {
3     assert!(!cfg.is_null());
4
5     handlemap_ext::handle_result(|| {
6         let glean_cfg = glean_core::Configuration::try_from(&*cfg)?;
7         let glean = Glean::new(glean_cfg)?;
8         glean_core::setup_glean(glean)?;
9         log::info!("Glean initialized");
10        Ok(true)
11    })
12 }
```

<https://github.com/mozilla/glean/blob/main/glean-core/ffi/src/lib.rs>

Glean FFI

```
1 #[no_mangle]
2 pub unsafe extern "C" fn glean_initialize(cfg: *const FfiConfiguration) -> u8 {
3     assert!(!cfg.is_null());
4
5     handlemap_ext::handle_result(|| {
6         let glean_cfg = glean_core::Configuration::try_from(&*cfg)?;
7         let glean = Glean::new(glean_cfg)?;
8         glean_core::setup_glean(glean)?;
9         log::info!("Glean initialized");
10        Ok(true)
11    })
12 }
```

<https://github.com/mozilla/glean/blob/main/glean-core/ffi/src/lib.rs>

Glean FFI

```
1 #[no_mangle]
2 pub unsafe extern "C" fn glean_initialize(cfg: *const FfiConfiguration) -> u8 {
3     assert!(!cfg.is_null());
4
5     handlemap_ext::handle_result(|| {
6         let glean_cfg = glean_core::Configuration::try_from(&*cfg)?;
7         let glean = Glean::new(glean_cfg)?;
8         glean_core::setup_glean(glean)?;
9         log::info!("Glean initialized");
10        Ok(true)
11    })
12 }
```

<https://github.com/mozilla/glean/blob/main/glean-core/ffi/src/lib.rs>

Glean FFI

```
1 #[no_mangle]
2 pub unsafe extern "C" fn glean_initialize(cfg: *const FfiConfiguration) -> u8 {
3     assert!(!cfg.is_null());
4
5     handlemap_ext::handle_result(|| {
6         let glean_cfg = glean_core::Configuration::try_from(&*cfg)?;
7         let glean = Glean::new(glean_cfg)?;
8         glean_core::setup_glean(glean)?;
9         log::info!("Glean initialized");
10        Ok(true)
11    })
12 }
```

<https://github.com/mozilla/glean/blob/main/glean-core/ffi/src/lib.rs>

cbindgen

creates C/C++11 headers for Rust libraries which
expose a public C API

```
1 #include <stdint.h>
2 #include <stdlib.h>
3
4 typedef struct FfiConfiguration {
5     // ...
6 } FfiConfiguration;
7
8 uint8_t glean_initialize(const struct FfiConfiguration *cfg);
```

<https://github.com/eqrion/cbindgen>

<https://github.com/mozilla/glean/blob/main/glean-core/ffi/glean.h>

ffi-support

Library to simplify implementing FFI libraries

Makes it easy to convert Rust types into FFI-compatible types

```
1 unsafe trait IntoFfi: Sized {  
2     type Value;  
3     fn ffi_default() -> Self::Value;  
4     fn into_ffi_value(self) -> Self::Value;  
5 }  
6  
7 unsafe impl IntoFfi for String {  
8     type Value = *mut c_char;  
9     // ...  
10 }
```

https://docs.rs/ffi-support/0.4.2/ffi_support

ffi-support

Library to simplify implementing FFI libraries

Makes it easy to convert Rust types into FFI-compatible types

```
1 unsafe trait IntoFfi: Sized {  
2     type Value;  
3     fn ffi_default() -> Self::Value;  
4     fn into_ffi_value(self) -> Self::Value;  
5 }  
6  
7 unsafe impl IntoFfi for String {  
8     type Value = *mut c_char;  
9     // ...  
10 }
```

https://docs.rs/ffi-support/0.4.2/ffi_support

ffi-support

Library to simplify implementing FFI libraries

Makes it easy to convert Rust types into FFI-compatible types

```
1 unsafe trait IntoFfi: Sized {  
2     type Value;  
3     fn ffi_default() -> Self::Value;  
4     fn into_ffi_value(self) -> Self::Value;  
5 }  
6  
7 unsafe impl IntoFfi for String {  
8     type Value = *mut c_char;  
9     // ...  
10 }
```

https://docs.rs/ffi-support/0.4.2/ffi_support

ffi-support

Library to simplify implementing FFI libraries

Makes it easy to convert Rust types into FFI-compatible types

```
1 unsafe trait IntoFfi: Sized {  
2     type Value;  
3     fn ffi_default() -> Self::Value;  
4     fn into_ffi_value(self) -> Self::Value;  
5 }  
6  
7 unsafe impl IntoFfi for String {  
8     type Value = *mut c_char;  
9     // ...  
10 }
```

https://docs.rs/ffi-support/0.4.2/ffi_support

Compile Targets

```
1 $ rustup target list
2   aarch64-apple-ios
3   aarch64-fuchsia
4   aarch64-linux-android
5   aarch64-pc-windows-msvc
6   ...
```

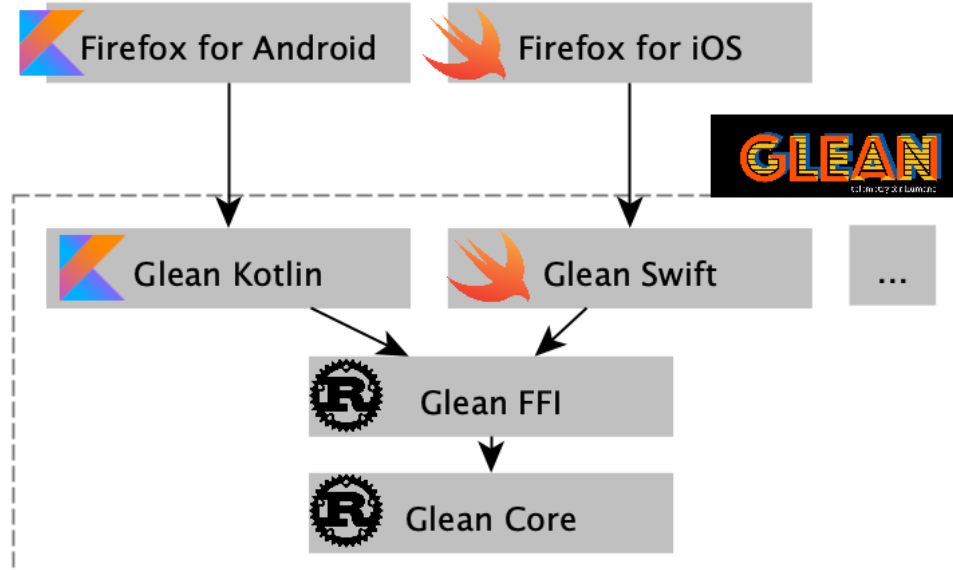
```
1 $ rustup target add aarch64-linux-android
2 $ rustup target add aarch64-apple-ios
3 $ rustup target add x86_64-apple-darwin
4   ...
```

+ SDKs for Android and iOS

<https://rust-lang.github.io/rustup/cross-compilation.html>

<https://blog.mozilla.org/data/2021/04/16/this-week-in-glean-rustc-ios-and-an-m1>

Glean SDK



Glean Implementations

- Swift
- Python
- C++
- JavaScript
- Rust

Future of Glean - uniffi

multi-language bindings generator

```
1 interface TodoList { // Interfaces defined via WebIDL
2     constructor();
3     void add_item(string todo);
4 };
```

```
1 struct TodoList { // Rust implementation
2     items: Vec<String>
3 }
4
5 impl TodoList {
6     fn new() -> Self {
7         TodoList { items: Vec::new() }
8     }
9
10    fn add_item(&mut self, todo: String) {
11        self.items.push(todo);
12    }
13 }
```

<https://mozilla.github.io/uniffi-rs/udl/interfaces.html>

uniffi - Usage

Python

```
1 from todo import *
2
3 todoList = TodoList()
4 todoList.add_item("Write documentation")
```

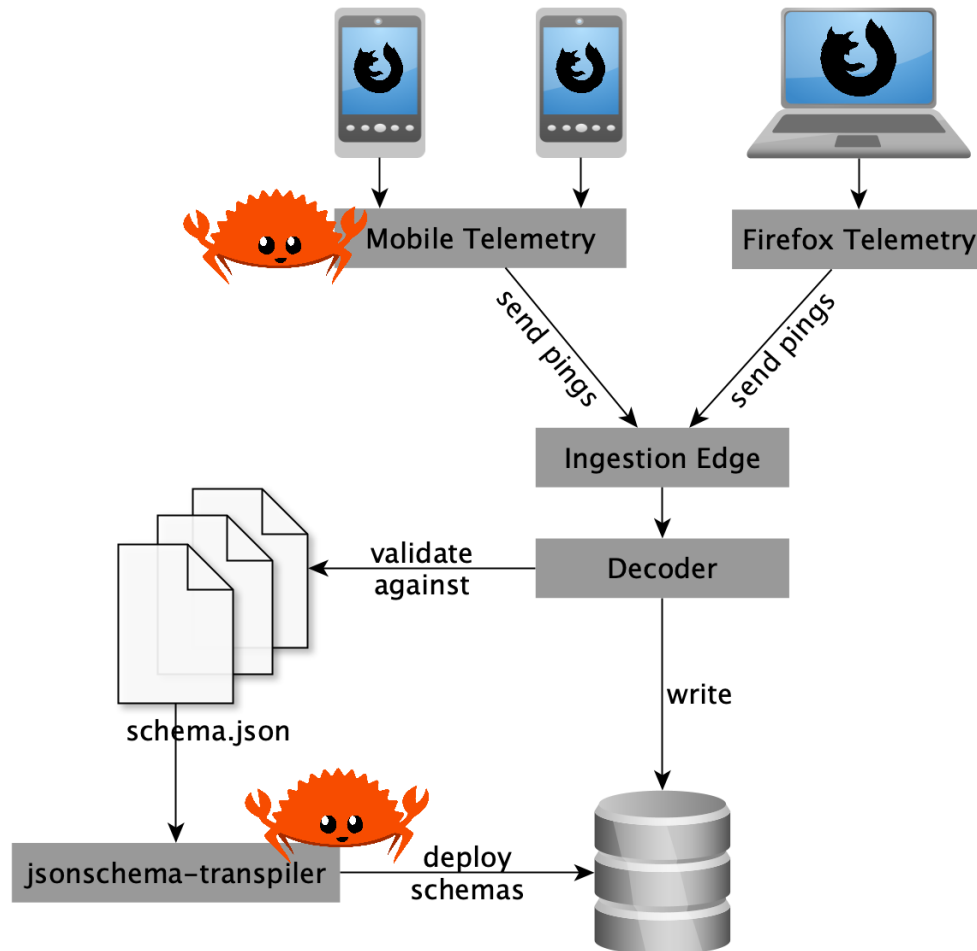
Swift

```
1 import todo
2
3 let todoList = TodoList()
4 todoList.addItem(todo: "Write documentation")
```

Kotlin

...

Where is Rust used?



jsonschema-transpiler

A tool for transpiling JSON Schema into schemas for
Avro and BigQuery.



Cloud data warehouse that Mozilla uses to store telemetry data.

<https://cloud.google.com/bigquery>

Schemas for Mozilla's data ingestion pipeline and data lake outputs.

The screenshot shows the GitHub repository for mozilla-services/mozilla-pipeline-schemas. The repository has 16 watches, 31 stars, and 81 forks. The main branch is selected. The file path is mozilla-pipeline-schemas / schemas / telemetry /. Below the file path, there is a list of files and folders. The 'crash' folder is highlighted, showing a commit by wlach with the message 'Add some missing crash annotations (#677)' 4 days ago. The list of files and folders includes:

File/Folder	Commit Message	Commit Date
addon-install-blocked	uri_scheme -> bq_metadata_format	9 months ago
advancedtelemetry	uri_scheme -> bq_metadata_format	9 months ago
anonymous	uri_scheme -> bq_metadata_format	9 months ago
bhr	[Bug 1709182] Add environment.settings.update.background column in pi...	19 days ago
block-autoplay	uri_scheme -> bq_metadata_format	9 months ago
certificate-checker	uri_scheme -> bq_metadata_format	9 months ago
core	Bug 1637340 - openTabCount in core ping	9 months ago
crash	Add some missing crash annotations (#677)	4 days ago
deletion-request	uri_scheme -> bq_metadata_format	9 months ago
deletion	uri_scheme -> bq_metadata_format	9 months ago
deployment-checker	uri_scheme -> bq_metadata_format	9 months ago
disable-sha1rollout	uri_scheme -> bq_metadata_format	9 months ago

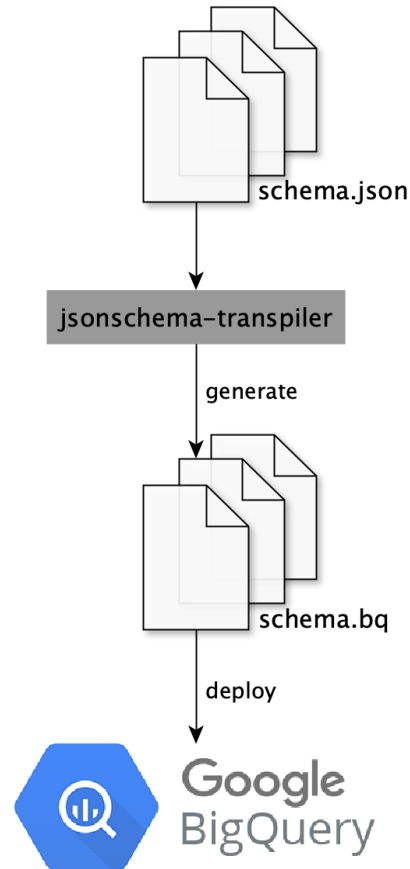
<https://github.com/mozilla-services/mozilla-pipeline-schemas>

JSON Schema

```
1 {
2   "$schema": "http://json-schema.org/draft-04/schema#",
3   "properties": {
4     "clientId": {
5       "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{12}$",
6       "type": "string"
7     },
8     "creationDate": {
9       "pattern": "^[0-9]{4}-[0-9]{2}-[0-9]{2}T[0-9]{2}:[0-9]{2}:[0-9]{2}\\.[0-9]{3}Z$",
10      "type": "string"
11    },
12    "payload": {
13      "properties": {
14        "crashDate": {
15          "pattern": "^[0-9]{4}-[0-9]{2}-[0-9]{2}$",
16          "type": "string"
17        },
18        "crashId": {
19          "description": "Optional, ID of the associated crash. Added to schema in bug 1604666.",
20          "type": "string"
21        },
22        "processType": {
23          "type": "string"
24        },
25        "stackTraces": {
```

<https://github.com/mozilla-services/mozilla-pipeline-schemas/blob/main/schemas/telemetry/crash/crash.4.schema.json>

jsonschema-transpiler Usage



Generating BigQuery Schemas

```
1 // JSON Schema
2 {
3   "type": "object",
4   "properties": {
5     "clientId": {
6       "type": "string"
7     },
8     "payload": {
9       "properties": {
10        "crashDate": {
11          "pattern":
12            "^[0-9]{4}-[0-9]{2}-[0-9]{2}$",
13          "type": "string"
14        }
15      }
16    }
17  }
18 }
```

```
1 // BigQuery Schema
2 // jsonschema-transpiler --type bigquery
3 {
4   "fields": [
5     {
6       "mode": "NULLABLE",
7       "name": "client_id",
8       "type": "STRING"
9     },
10    {
11      "fields": [
12        {
13          "mode": "NULLABLE",
14          "name": "crash_date",
15          "type": "STRING"
16        },
17      ],
18      "mode": "NULLABLE",
19      "name": "payload",
20      "type": "RECORD"
21    }
22  ]
23 }
```

serde_json

Deserializes schemas directly from their JSON counterparts into Rust structs and enums.

https://docs.serde.rs/serde_json/

Types and Tags

Type: set of symbols and rules for producing a sequence of those symbols

```
1 #[derive(Serialize, Deserialize, Debug)]
2 #[serde(rename_all = "kebab-case")]
3 enum Atom {
4     Null,
5     Boolean,
6     Integer,
7     String,
8     // ...
9 }
10
11 enum Type {
12     Atom(Atom),
13     List(Vec<Atom>),
14 }
15
16 let root = Type::List(vec![
17     Type::Null,
18     Type::Atom(Atom::Boolean),
19     Type::List(vec![
20         Type::Null,
21         Type::Atom(Atom::Integer)
22     ])
23 ]);
```

<https://github.com/mozilla/jsonschema-transpiler/blob/main/src/jsonschema.rs#L12-L29>

Types and Tags

Tag: attribute data associated with a type

```
1 #[derive(Serialize, Deserialize, Debug)]
2 #[serde(rename_all = "UPPERCASE")]
3 pub enum Atom {
4     Int64,
5     Float64,
6     Bool,
7     // ...
8 }
9
10 pub struct Record {
11     fields: HashMap<String, Box<Tag>>,
12 }
13
14 pub enum Type {
15     Atom(Atom),
16     Record(Record),
17 }
18
19 pub struct Tag {
20     name: Option<String>,
21     #[serde(flatten, rename = "type")]
22     data_type: Box<Type>,
23     description: Option<String>,
24 }
```

<https://github.com/mozilla/jsonschema-transpiler/blob/main/src/bigquery.rs#L33-L58>

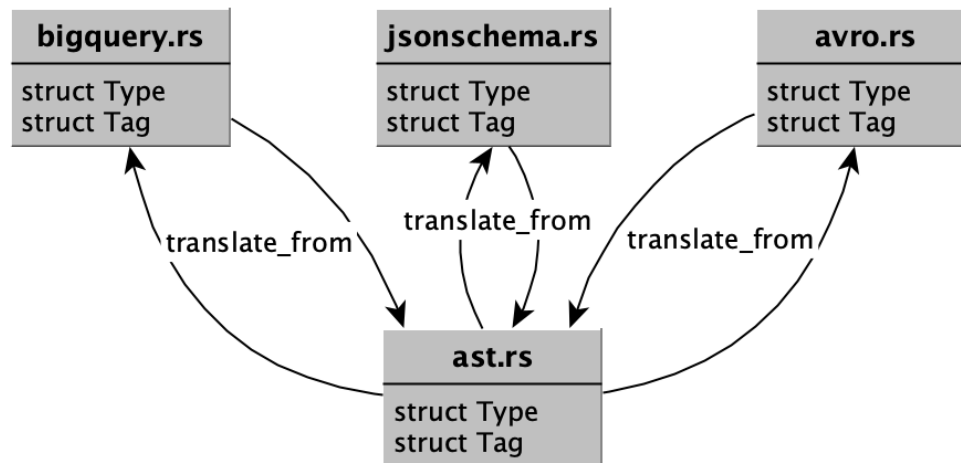
Types and Tags

```
1 let root = Tag {  
2   data_type: Box::new(Type::Atom(Atom::String)),  
3   name: "object".to_string(),  
4   description: "Some cool object".to_string()  
5 };
```

```
1 {  
2   "name": "object",  
3   "type": "string",  
4   "description": "Some cool object"  
5 }
```

<https://github.com/mozilla/jsonschema-transpiler/blob/main/src/bigquery.rs#L33-L58>

Transpiling Tags and Types



TranslateFrom

```
1 impl TranslateFrom for Tag {
2     fn translate_from(tag: ast::Tag, context: Context) -> Result {
3         // ...
4         let data_type = match &tag.data_type {
5             ast::Type::Atom(atom) => Type::Atom(match atom {
6                 ast::Atom::Boolean => Atom::Bool,
7                 ast::Atom::Integer => Atom::Int64,
8                 ast::Atom::String => Atom::String,
9                 // ...
10            }),
11         // ...
12     }
13
14     // ...
15
16     Ok(Tag {
17         name: tag.name.clone(),
18         data_type: Box::new(data_type),
19         description,
20     })
21 }
22 }
```

<https://github.com/mozilla/jsonschema-transpiler/blob/main/src/bigquery.rs#L60>

TranslateFrom

```
1 impl TranslateFrom for Tag {
2     fn translate_from(tag: ast::Tag, context: Context) -> Result {
3         // ...
4         let data_type = match &tag.data_type {
5             ast::Type::Atom(atom) => Type::Atom(match atom {
6                 ast::Atom::Boolean => Atom::Bool,
7                 ast::Atom::Integer => Atom::Int64,
8                 ast::Atom::String => Atom::String,
9                 // ...
10            }),
11         // ...
12     }
13
14     // ...
15
16     Ok(Tag {
17         name: tag.name.clone(),
18         data_type: Box::new(data_type),
19         description,
20     })
21 }
22 }
```

<https://github.com/mozilla/jsonschema-transpiler/blob/main/src/bigquery.rs#L60>

TranslateFrom

```
1 impl TranslateFrom for Tag {
2     fn translate_from(tag: ast::Tag, context: Context) -> Result {
3         // ...
4         let data_type = match &tag.data_type {
5             ast::Type::Atom(atom) => Type::Atom(match atom {
6                 ast::Atom::Boolean => Atom::Bool,
7                 ast::Atom::Integer => Atom::Int64,
8                 ast::Atom::String => Atom::String,
9                 // ...
10            }),
11         // ...
12     }
13
14     // ...
15
16     Ok(Tag {
17         name: tag.name.clone(),
18         data_type: Box::new(data_type),
19         description,
20     })
21 }
22 }
```

<https://github.com/mozilla/jsonschema-transpiler/blob/main/src/bigquery.rs#L60>

Transpile! ✨

```
1 pub fn convert_bigquery(input: &Value, context: Context) -> Value {
2     let bq = bigquery::Tag::translate_from(into_ast(input, context), context).unwrap();
3     json!(bq)
4 }
5
6 fn into_ast(input: &Value, context: Context) -> ast::Tag {
7     let jsonschema: jsonschema::Tag = match serde_json::from_value(json!(input)) {
8         Ok(tag) => tag,
9         Err(e) => panic!(format!("{:#?}", e)),
10    };
11    ast::Tag::translate_from(jsonschema, context).unwrap()
12 }
```

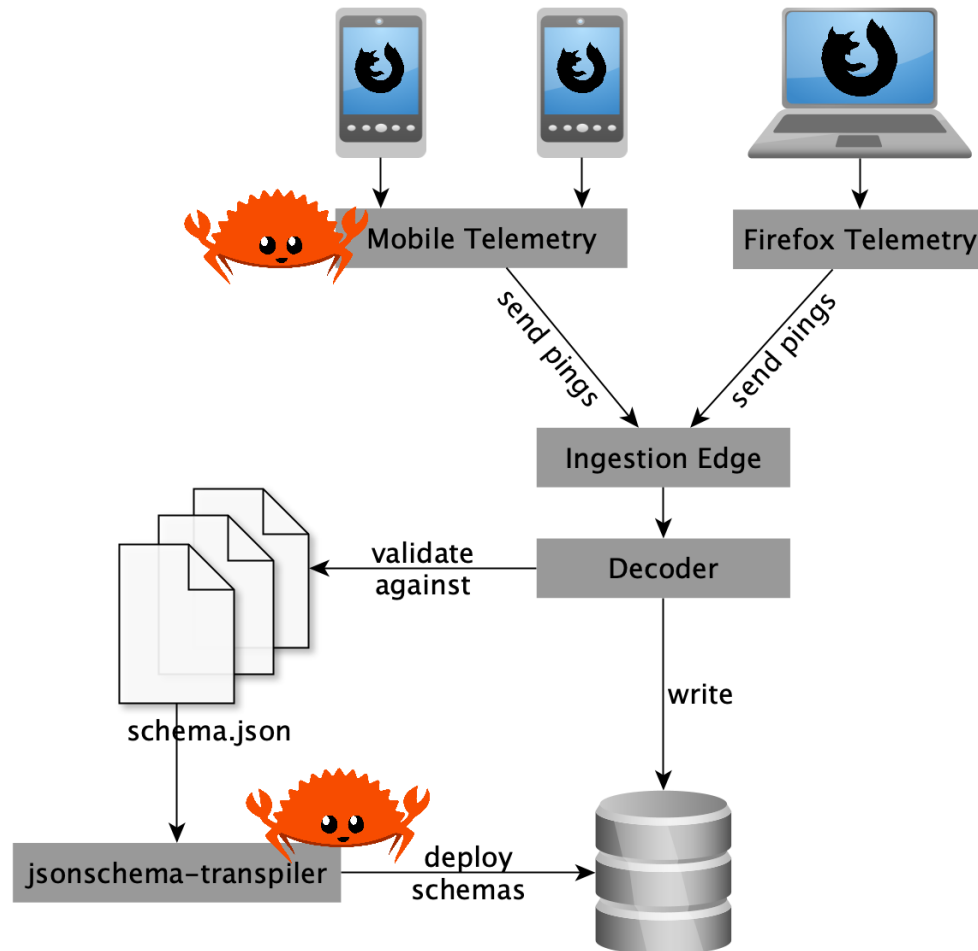
<https://github.com/mozilla/jsonschema-transpiler/blob/main/src/lib.rs#L72-L90>

Transpile! ✨

```
1 pub fn convert_bigquery(input: &Value, context: Context) -> Value {
2     let bq = bigquery::Tag::translate_from(into_ast(input, context), context).unwrap();
3     json!(bq)
4 }
5
6 fn into_ast(input: &Value, context: Context) -> ast::Tag {
7     let jsonschema: jsonschema::Tag = match serde_json::from_value(json!(input)) {
8         Ok(tag) => tag,
9         Err(e) => panic!(format!("{:?}", e)),
10    };
11    ast::Tag::translate_from(jsonschema, context).unwrap()
12 }
```

<https://github.com/mozilla/jsonschema-transpiler/blob/main/src/lib.rs#L72-L90>

Where is Rust used?



Questions?