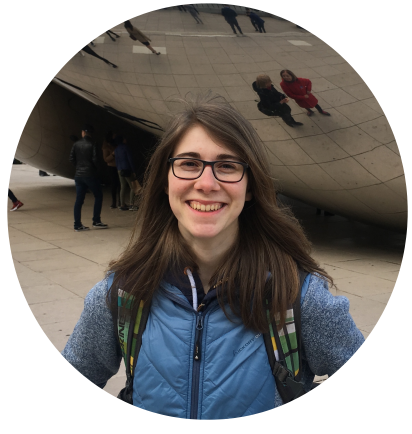
Rust in Mozilla's Data Platform

Vancouver Rust Meetup

Anna Scholtz | 2021-06-16

Hello 👋

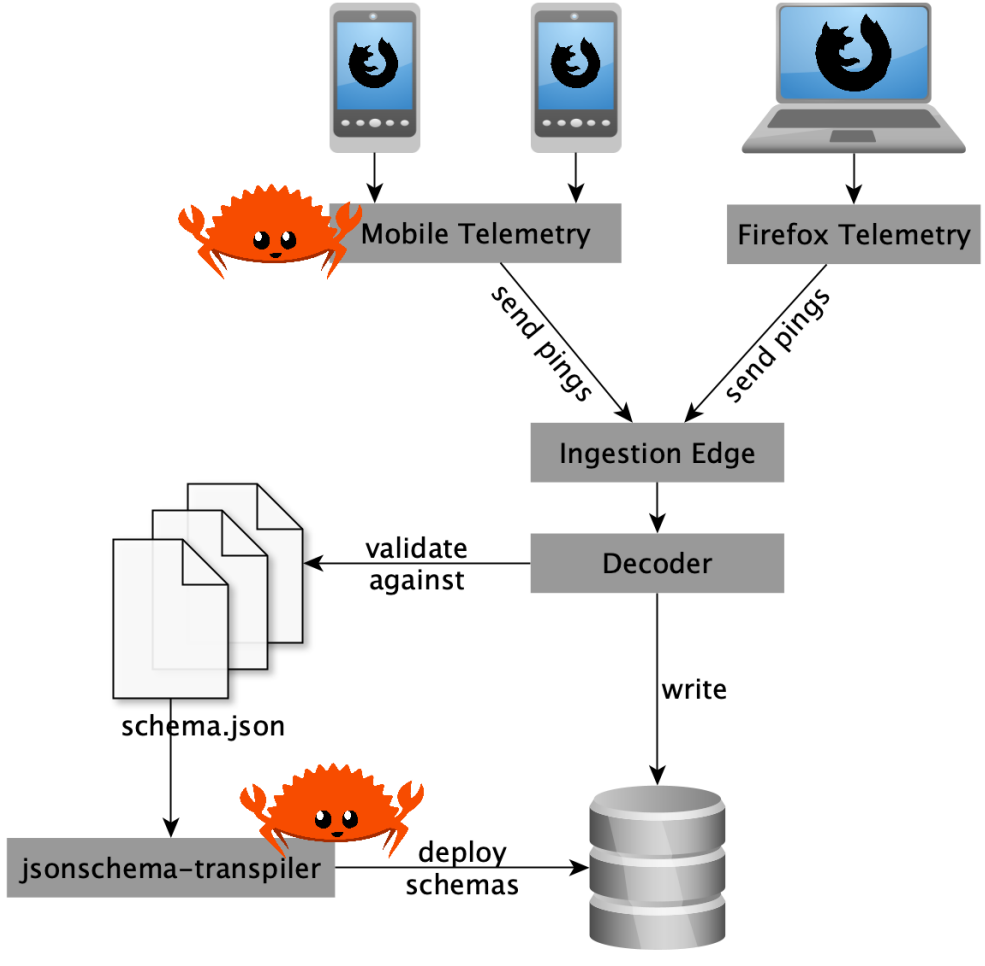


Anna Scholtz

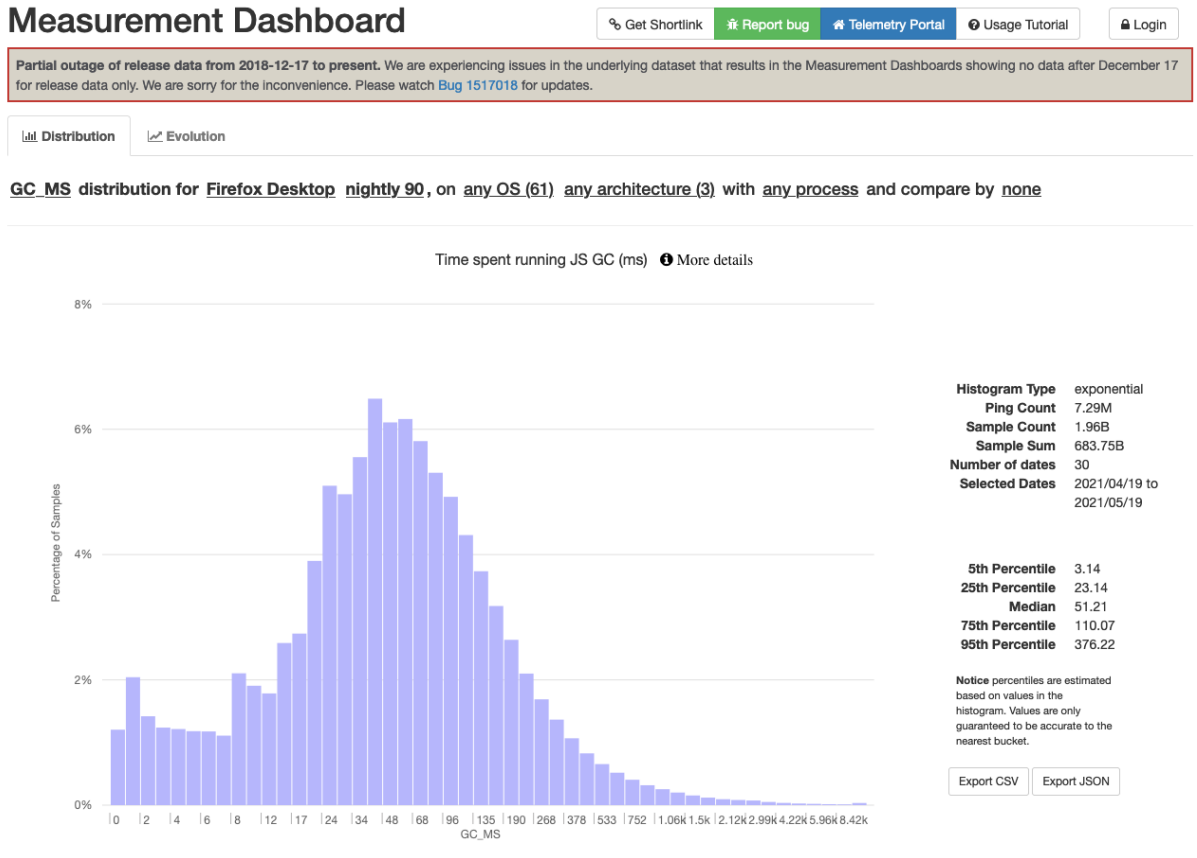
Data Engineer @ Mozilla

@scholtzan

Where is Rust used?



Why is data collected?



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

Lean Data Practices

Stay Lean

Decide if all your data collection delivers value.

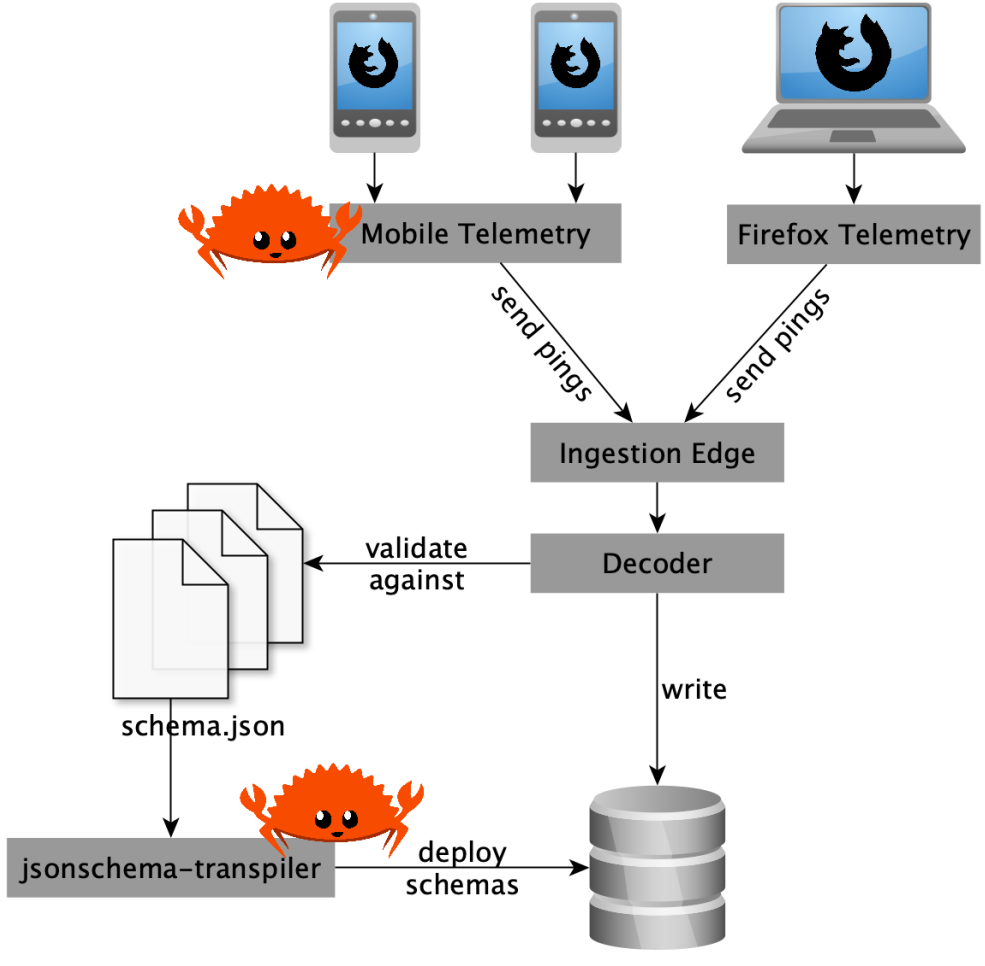
Build Security

Learn how to protect customer data.

Engage Your Users

Keep customers informed and empowered.

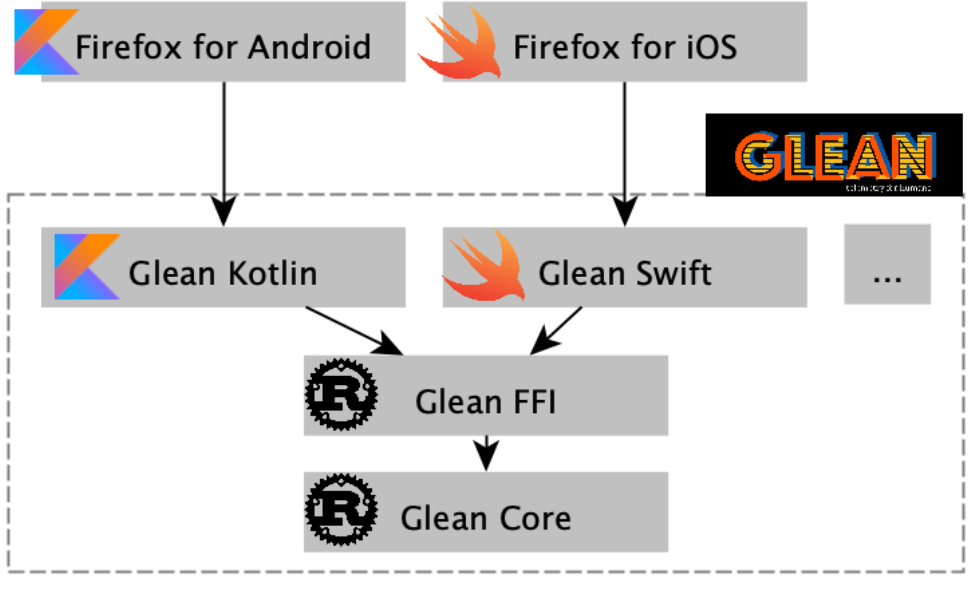
Where is Rust used?



Telemetry library that performs measurements and sends data from Firefox products

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

Glean SDK


Firefox for Android
Firefox for iOS
GLEAN
Glean Kotlin
Glean Swift
...
Glean FFI
Glean Core

Glean Core

```rust
#[derive(Debug)]
pub struct Glean {
    upload_enabled: bool,
    data_store: Option,
    event_data_store: EventDatabase,
    core_metrics: CoreMetrics,
    additional_metrics: AdditionalMetrics,
    // ...
}
```

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

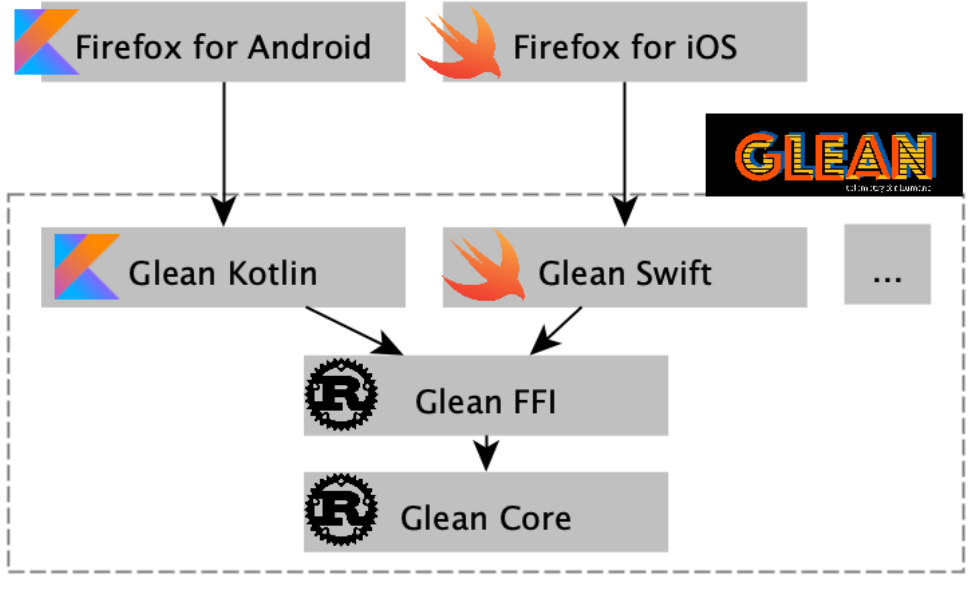
Metric Types

```rust
pub struct CounterMetric {
    meta: CommonMetricData,
}

impl CounterMetric {
    pub fn add(&self, glean: &Glean, amount: i32) {
        glean
            .storage()
            .record_with(
              glean,
              &self.meta,
              old_value.add(amount)
            )
    }
}
```

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

Glean SDK


Firefox for Android
Firefox for iOS
GLEAN
Glean Kotlin
Glean Swift
...
Glean FFI
Glean Core

Glean FFI

FFI - Foreign Function Interface

Connector to lower-level Rust crate

Glean FFI

```rust
#[no_mangle]
pub unsafe extern "C" fn glean_initialize(cfg: *const FfiConfiguration) -> u8 {
    assert!(!cfg.is_null());

    handlemap_ext::handle_result(|| {
        let glean_cfg = glean_core::Configuration::try_from(&*cfg)?;
        let glean = Glean::new(glean_cfg)?;
        glean_core::setup_glean(glean)?;
        log::info!("Glean initialized");
        Ok(true)
    })
}
```

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

Glean FFI

```rust
#[no_mangle]
pub unsafe extern "C" fn glean_initialize(cfg: *const FfiConfiguration) -> u8 {
    assert!(!cfg.is_null());

    handlemap_ext::handle_result(|| {
        let glean_cfg = glean_core::Configuration::try_from(&*cfg)?;
        let glean = Glean::new(glean_cfg)?;
        glean_core::setup_glean(glean)?;
        log::info!("Glean initialized");
        Ok(true)
    })
}
```

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

Glean FFI

```rust
#[no_mangle]
pub unsafe extern "C" fn glean_initialize(cfg: *const FfiConfiguration) -> u8 {
    assert!(!cfg.is_null());

    handlemap_ext::handle_result(|| {
        let glean_cfg = glean_core::Configuration::try_from(&*cfg)?;
        let glean = Glean::new(glean_cfg)?;
        glean_core::setup_glean(glean)?;
        log::info!("Glean initialized");
        Ok(true)
    })
}
```

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

Glean FFI

```rust
#[no_mangle]
pub unsafe extern "C" fn glean_initialize(cfg: *const FfiConfiguration) -> u8 {
    assert!(!cfg.is_null());

    handlemap_ext::handle_result(|| {
        let glean_cfg = glean_core::Configuration::try_from(&*cfg)?;
        let glean = Glean::new(glean_cfg)?;
        glean_core::setup_glean(glean)?;
        log::info!("Glean initialized");
        Ok(true)
    })
}
```

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

```c
#include <stdint.h>
#include <stdlib.h>

typedef struct FfiConfiguration {
    // ...
} FfiConfiguration;

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

```rust
unsafe trait IntoFfi: Sized {
    type Value;
    fn ffi_default() -> Self::Value;
    fn into_ffi_value(self) -> Self::Value;
}

unsafe impl IntoFfi for String {
    type Value = *mut c_char;
    // ...
}
```

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

```rust
unsafe trait IntoFfi: Sized {
    type Value;
    fn ffi_default() -> Self::Value;
    fn into_ffi_value(self) -> Self::Value;
}

unsafe impl IntoFfi for String {
    type Value = *mut c_char;
    // ...
}
```

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

```rust
unsafe trait IntoFfi: Sized {
    type Value;
    fn ffi_default() -> Self::Value;
    fn into_ffi_value(self) -> Self::Value;
}

unsafe impl IntoFfi for String {
    type Value = *mut c_char;
    // ...
}
```

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

```rust
unsafe trait IntoFfi: Sized {
    type Value;
    fn ffi_default() -> Self::Value;
    fn into_ffi_value(self) -> Self::Value;
}

unsafe impl IntoFfi for String {
    type Value = *mut c_char;
    // ...
}
```

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

Compile Targets

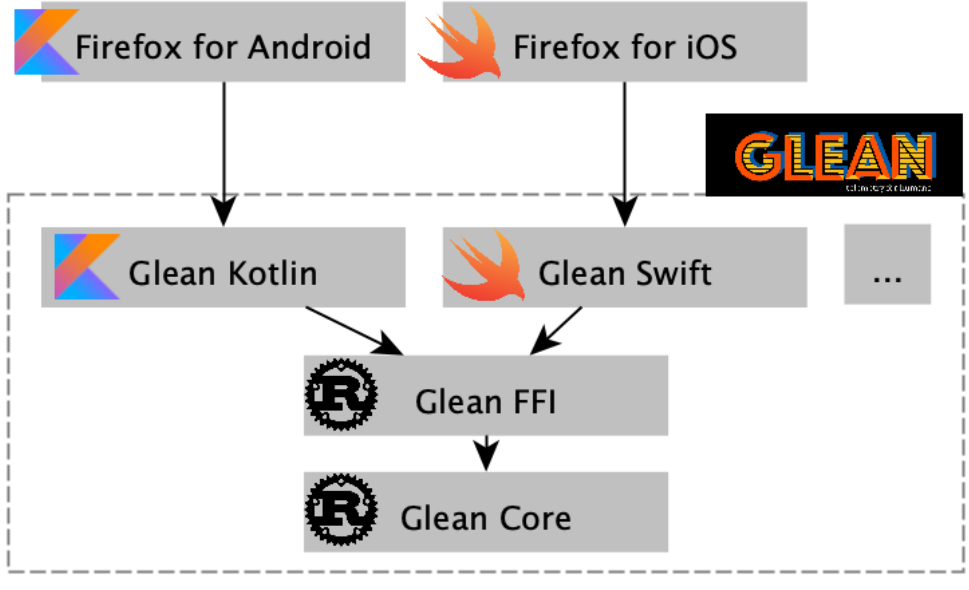
```
$ rustup target list
  aarch64-apple-ios
  aarch64-fuchsia
  aarch64-linux-android
  aarch64-pc-windows-msvc
  ...
```

```
$ rustup target add aarch64-linux-android
$ rustup target add aarch64-apple-ios
$ rustup target add x86_64-apple-darwin
  ...
```

+ 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


Firefox for Android
Firefox for iOS
GLEAN
Glean Kotlin
Glean Swift
...
Glean FFI
Glean Core

Glean Implementations

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

Future of Glean - uniffi

multi-language bindings generator

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

```
struct TodoList { // Rust implementation
    items: Vec<String>
}

impl TodoList {
    fn new() -> Self {
        TodoList { items: Vec::new() }
    }

    fn add_item(&mut self, todo: String) {
        self.items.push(todo);
    }
}
```

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

uniffi - Usage

Python

```python
from todo import *

todolist = TodoList()
todolist.add_item("Write documentation")
```

Swift

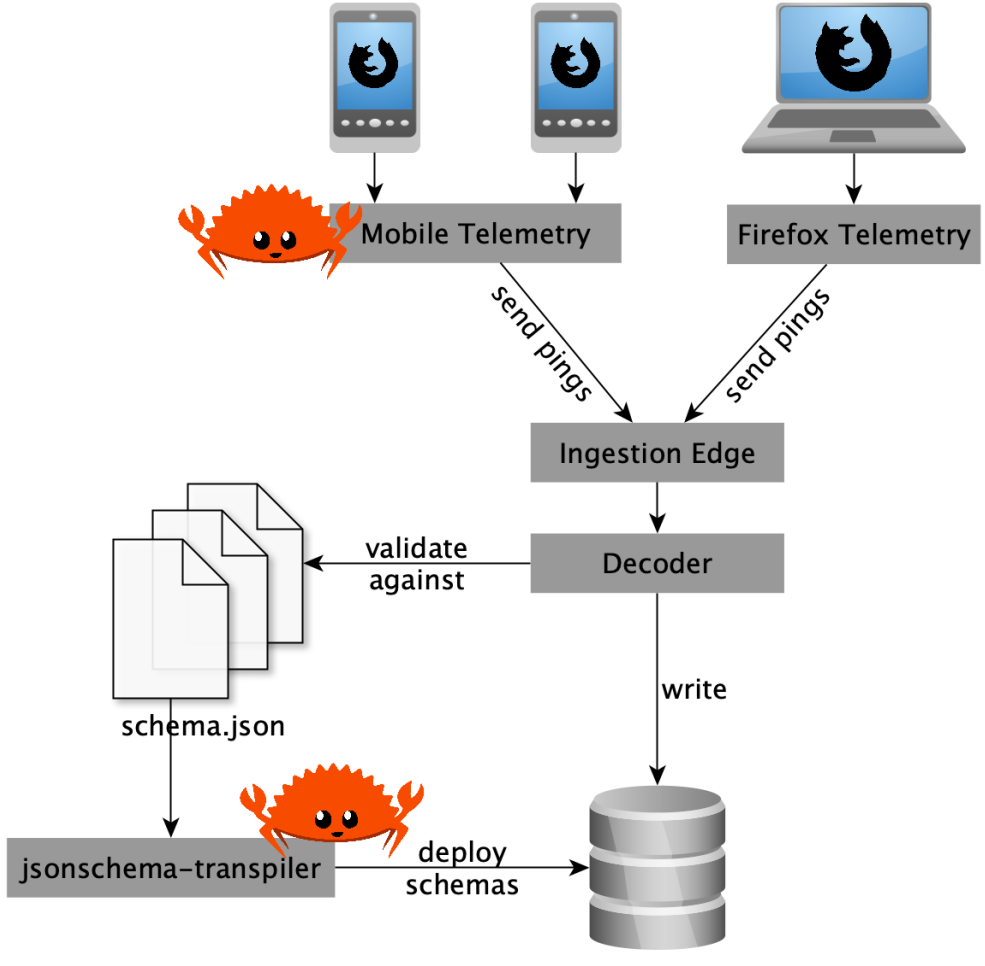
```swift
import todo

let todoList = TodoList()
todoList.addItem(todo: "Write documentation")
```

Kotlin

...

Where is Rust used?



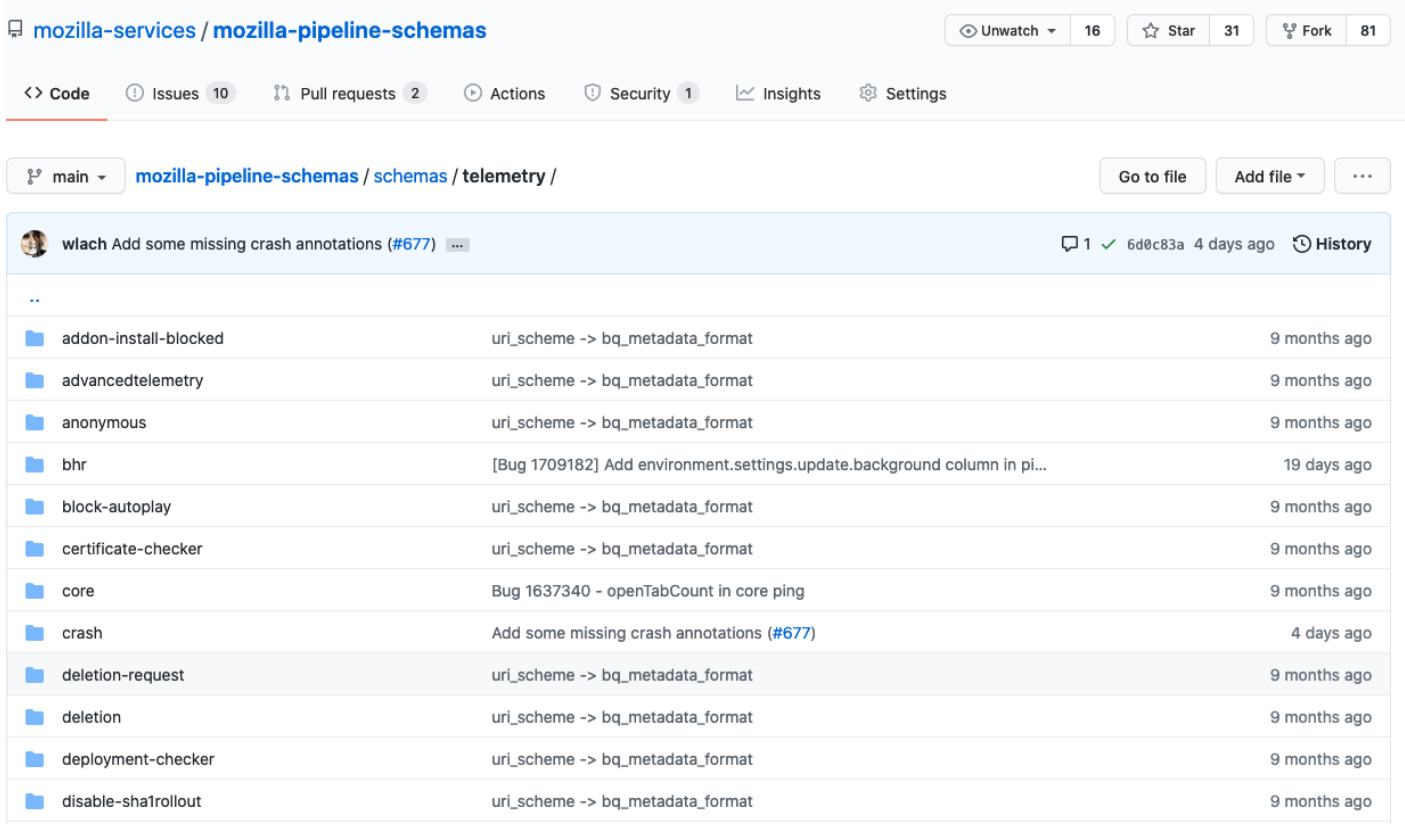
jsonschema-transpiler

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.



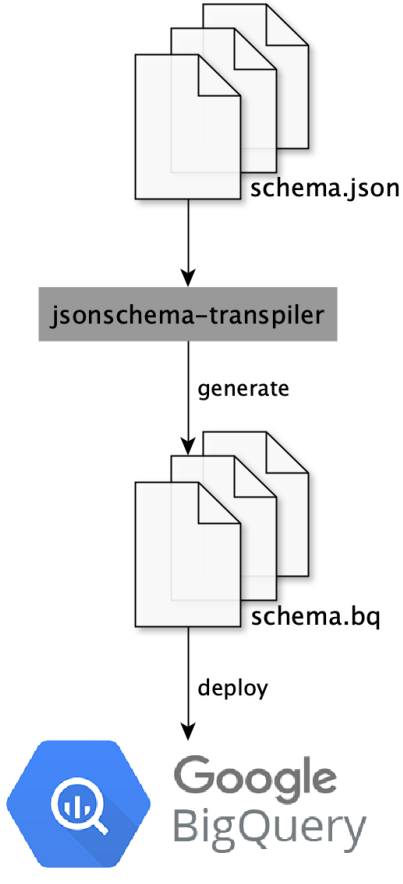
https://github.com/mozilla-services/mozilla-pipeline-schemas

JSON Schema

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "properties": {
    "clientId": {
      "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{12}$",
      "type": "string"
    },
    "creationDate": {
      "pattern": "^[0-9]{4}-[0-9]{2}-[0-9]{2}T[0-9]{2}:[0-9]{2}:[0-9]{2}\\.[0-9]{3}Z$",
      "type": "string"
    },
    "payload": {
      "properties": {
        "crashDate": {
          "pattern": "^[0-9]{4}-[0-9]{2}-[0-9]{2}$",
          "type": "string"
        },
        "crashId": {
          "description": "Optional, ID of the associated crash. Added to schema in bug 1604666.",
          "type": "string"
        },
        "processType": {
          "type": "string"
        },
        "stackTraces": {
```

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

jsonschema-transpiler Usage



Generating BigQuery Schemas

```
// JSON Schema
{
  "type": "object",
  "properties": {
    "clientId": {
      "type": "string"
    },
    "payload": {
      "properties": {
        "crashDate": {
          "pattern":
            "^[0-9]{4}-[0-9]{2}-[0-9]{2}$",
          "type": "string"
        }
      }
    }
  }
}
```

```
// BigQuery Schema
// jsonschema-transpiler --type bigquery
{
  "fields": [
    {
      "mode": "NULLABLE",
      "name": "client_id",
      "type": "STRING"
    },
    {
      "fields": [
        {
          "mode": "NULLABLE",
          "name": "crash_date",
          "type": "STRING"
        },
      ],
      "mode": "NULLABLE",
      "name": "payload",
      "type": "RECORD"
    }
  ]
}
```

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

```rust
#[derive(Serialize, Deserialize, Debug)]
#[serde(rename_all = "kebab-case")]
enum Atom {
    Null,
    Boolean,
    Integer,
    String,
    // ...
}

enum Type {
    Atom(Atom),
    List(Vec<Atom>),
}

let root = Type::List(vec![
    Type::Null,
    Type::Atom(Atom::Boolean),
    Type::List(vec![
        Type::Null,
        Type::Atom(Atom::Integer)
    ])
]);
```

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

Types and Tags

Tag: attribute data associated with a type

```rust
#[derive(Serialize, Deserialize, Debug)]
#[serde(rename_all = "UPPERCASE")]
pub enum Atom {
    Int64,
    Float64,
    Bool,
    // ...
}

pub struct Record {
    fields: HashMap<String, Box<Tag>>,
}

pub enum Type {
    Atom(Atom),
    Record(Record),
}

pub struct Tag {
    name: Option<String>,
    #[serde(flatten, rename = "type")]
    data_type: Box<Type>,
    description: Option<String>,
}
```

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

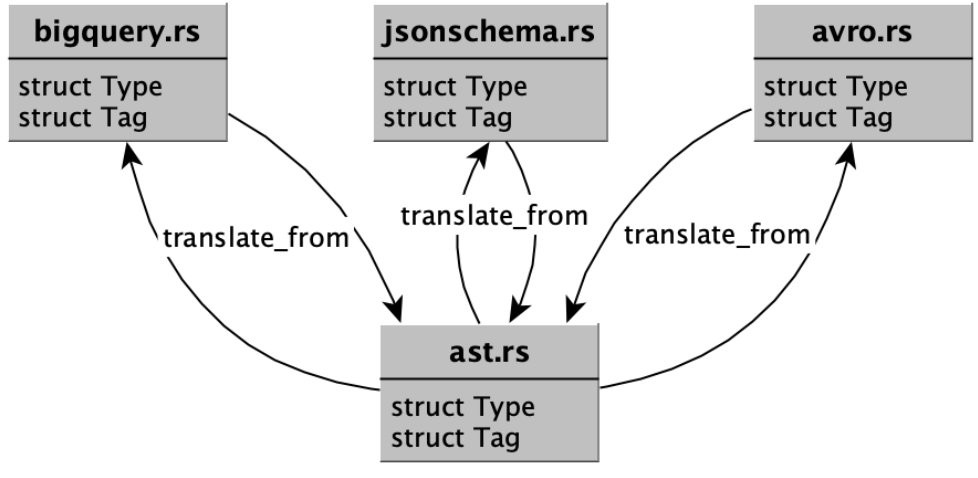
Types and Tags

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

```
{
    "name": "object",
    "type": "string",
    "description": "Some cool object"
}
```

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

Transpiling Tags and Types



TranslateFrom

```rust
impl TranslateFrom for Tag {
    fn translate_from(tag: ast::Tag, context: Context) -> Result {
        // ...
        let data_type = match &tag.data_type {
            ast::Type::Atom(atom) => Type::Atom(match atom {
                ast::Atom::Boolean => Atom::Bool,
                ast::Atom::Integer => Atom::Int64,
                ast::Atom::String => Atom::String,
                // ...
            }),
            // ...
        }

        // ...

        Ok(Tag {
            name: tag.name.clone(),
            data_type: Box::new(data_type),
            description,
        })
    }
}
```

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

TranslateFrom

```rust
impl TranslateFrom for Tag {
    fn translate_from(tag: ast::Tag, context: Context) -> Result {
        // ...
        let data_type = match &tag.data_type {
            ast::Type::Atom(atom) => Type::Atom(match atom {
                ast::Atom::Boolean => Atom::Bool,
                ast::Atom::Integer => Atom::Int64,
                ast::Atom::String => Atom::String,
                // ...
            }),
            // ...
        }

        // ...

        Ok(Tag {
            name: tag.name.clone(),
            data_type: Box::new(data_type),
            description,
        })
    }
}
```

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

TranslateFrom

```rust
impl TranslateFrom for Tag {
    fn translate_from(tag: ast::Tag, context: Context) -> Result {
        // ...
        let data_type = match &tag.data_type {
            ast::Type::Atom(atom) => Type::Atom(match atom {
                ast::Atom::Boolean => Atom::Bool,
                ast::Atom::Integer => Atom::Int64,
                ast::Atom::String => Atom::String,
                // ...
            }),
            // ...
        }

        // ...

        Ok(Tag {
            name: tag.name.clone(),
            data_type: Box::new(data_type),
            description,
        })
    }
}
```

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

Transpile! ✨

```rust
pub fn convert_bigquery(input: &Value, context: Context) -> Value {
    let bq = bigquery::Tag::translate_from(into_ast(input, context), context).unwrap();
    json!(bq)
}

fn into_ast(input: &Value, context: Context) -> ast::Tag {
    let jsonschema: jsonschema::Tag = match serde_json::from_value(json!(input)) {
        Ok(tag) => tag,
        Err(e) => panic!(format!("{:#?}", e)),
    };
    ast::Tag::translate_from(jsonschema, context).unwrap()
}
```

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

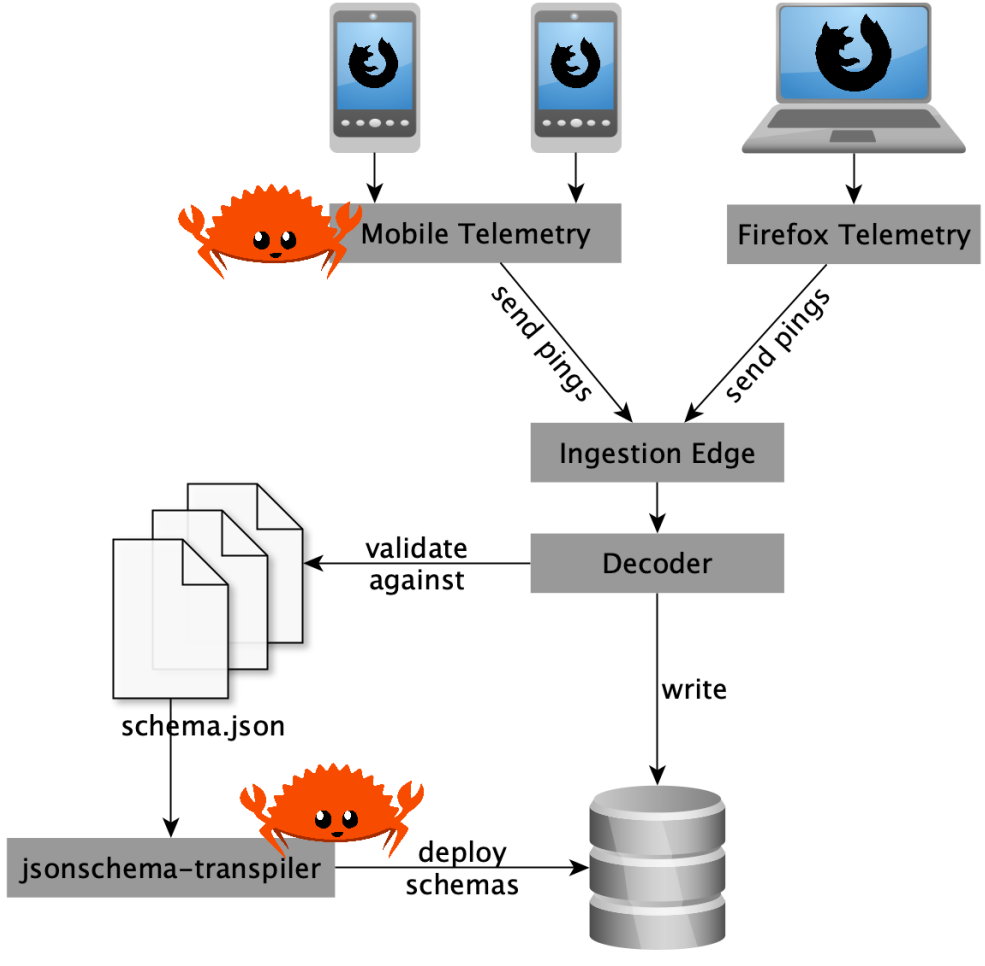
Transpile! ✨

```rust
pub fn convert_bigquery(input: &Value, context: Context) -> Value {
    let bq = bigquery::Tag::translate_from(into_ast(input, context), context).unwrap();
    json!(bq)
}

fn into_ast(input: &Value, context: Context) -> ast::Tag {
    let jsonschema: jsonschema::Tag = match serde_json::from_value(json!(input)) {
        Ok(tag) => tag,
        Err(e) => panic!(format!("{:#?}", e)),
    };
    ast::Tag::translate_from(jsonschema, context).unwrap()
}
```

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

Where is Rust used?



Questions?