Cap'n Proto and Rust
Type Systems for Sharing

David Renshaw
@dwrensha

19 September 2014

Sandstorm.io

CAP'N PROTO

things that are not Cap’n Proto

things that are not Cap’n Proto

- XML

things that are not Cap’n Proto

- XML
- JSON

things that are not Cap’n Proto

- XML
- JSON
- Protocol Buffers

things that are not Cap’n Proto

- XML
- JSON
- Protocol Buffers
- Thrift

things that are not Cap’n Proto

- XML
- JSON
- Protocol Buffers
- Thrift
- Avro

things that are not Cap’n Proto

- XML
- JSON
- Protocol Buffers
- Thrift
- Avro
- ICE

things that are not Cap’n Proto

- XML
- JSON
- Protocol Buffers
- Thrift
- Avro
- ICE
- CORBA

things that are not Cap’n Proto

- XML
- JSON
- Protocol Buffers
- Thrift
- Avro
- ICE
- CORBA
- SOAP

things that are not Cap’n Proto

- XML
- JSON
- Protocol Buffers
- Thrift
- Avro
- ICE
- CORBA
- SOAP
- ASN.1

things that are not Cap’n Proto

- XML
- JSON
- Protocol Buffers
- Thrift
- Avro
- ICE
- CORBA
- SOAP
- ASN.1
- FlatBuffers

things that are not Cap’n Proto

- XML
- JSON
- Protocol Buffers
- Thrift
- Avro
- ICE
- CORBA
- SOAP
- ASN.1
- FlatBuffers
- SBE

things that are not Cap’n Proto

- XML
- JSON
- Protocol Buffers
- Thrift
- Avro
- ICE
- CORBA
- SOAP
- ASN.1
- FlatBuffers
- SBE
- …

Cap'n Proto is a **Type System** for fast, robust, secure, multi-language **Distributed Computing**

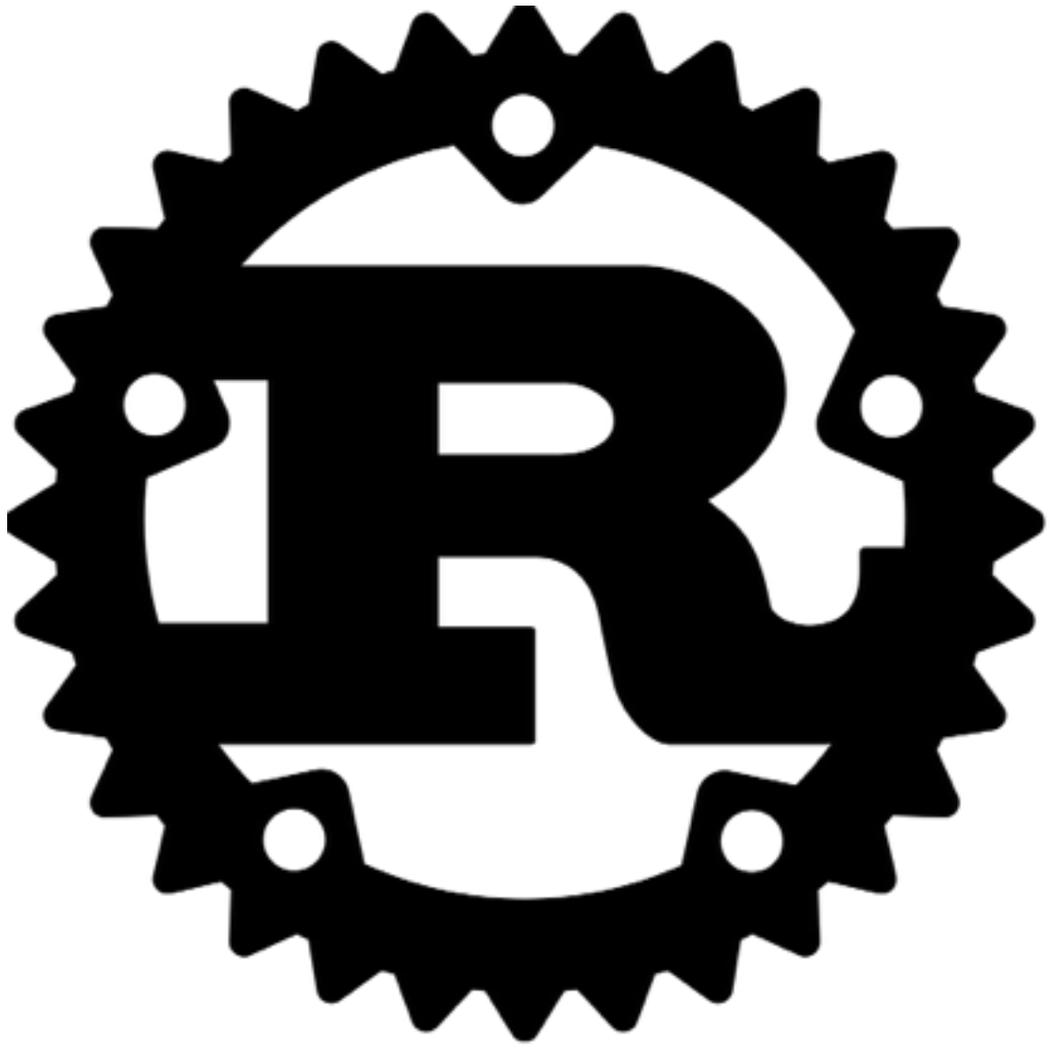


Photo Album

Image Analyzer

`analyze_image.rs`

image.capnp

Photo Album

Image Analyzer

analyze_image.rs

`image.capnp`

↓

Photo Album

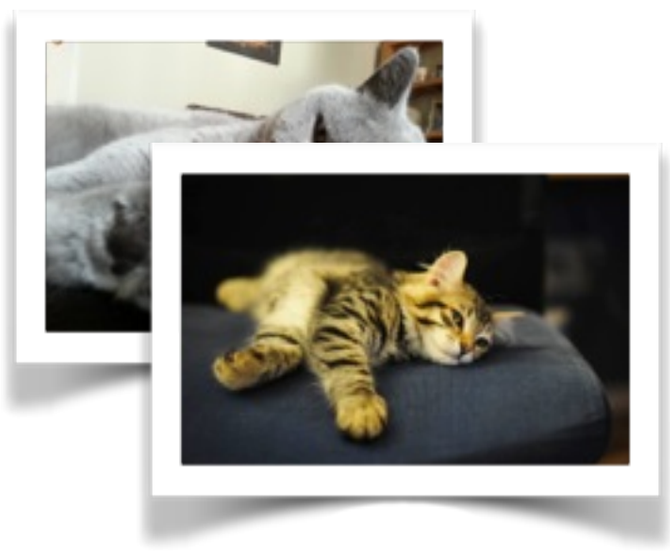


Image Analyzer

`analyze_image.rs`

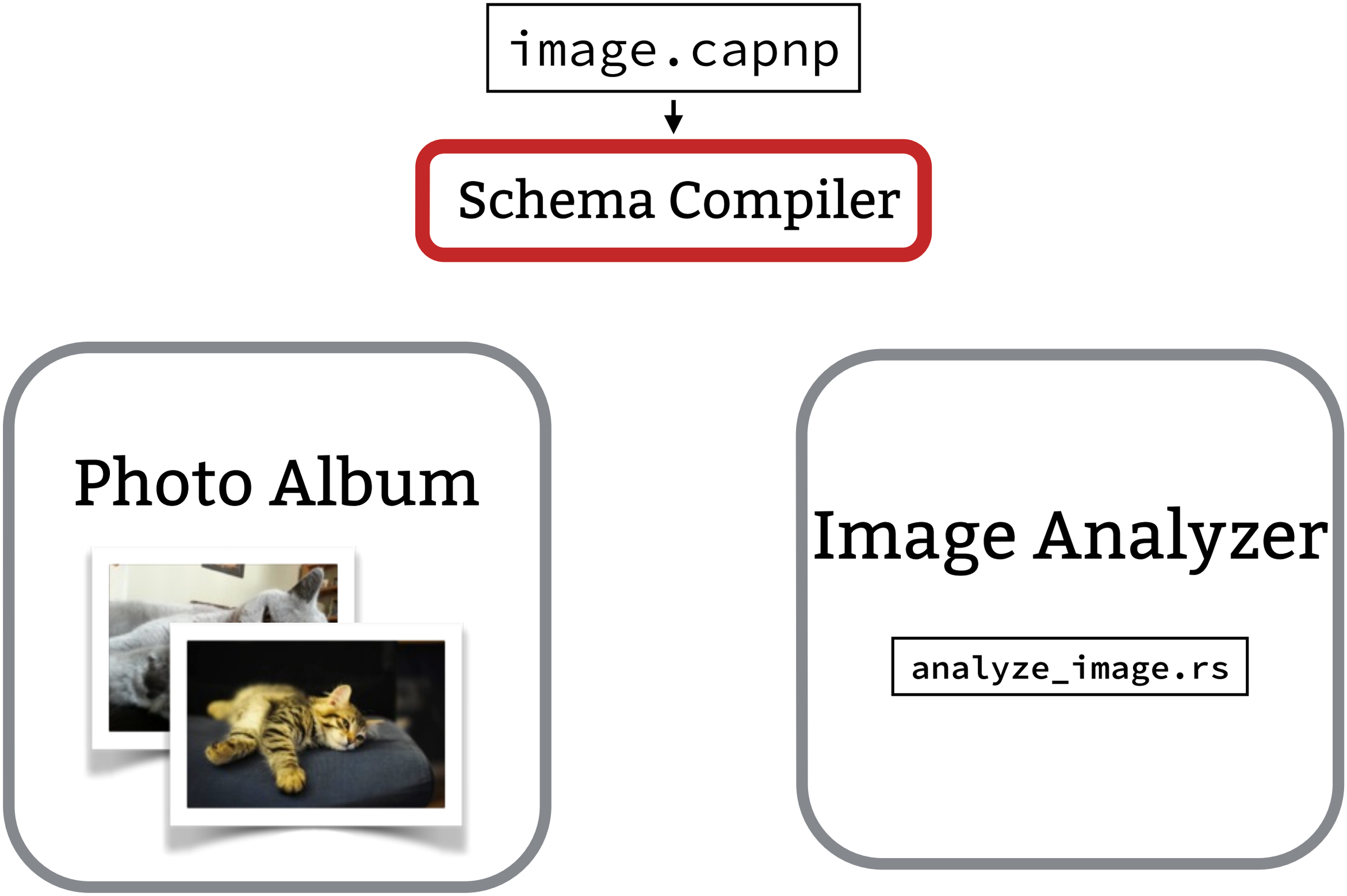


image.capnp
Schema Compiler
Photo Album
Image Analyzer
analyze_image.rs

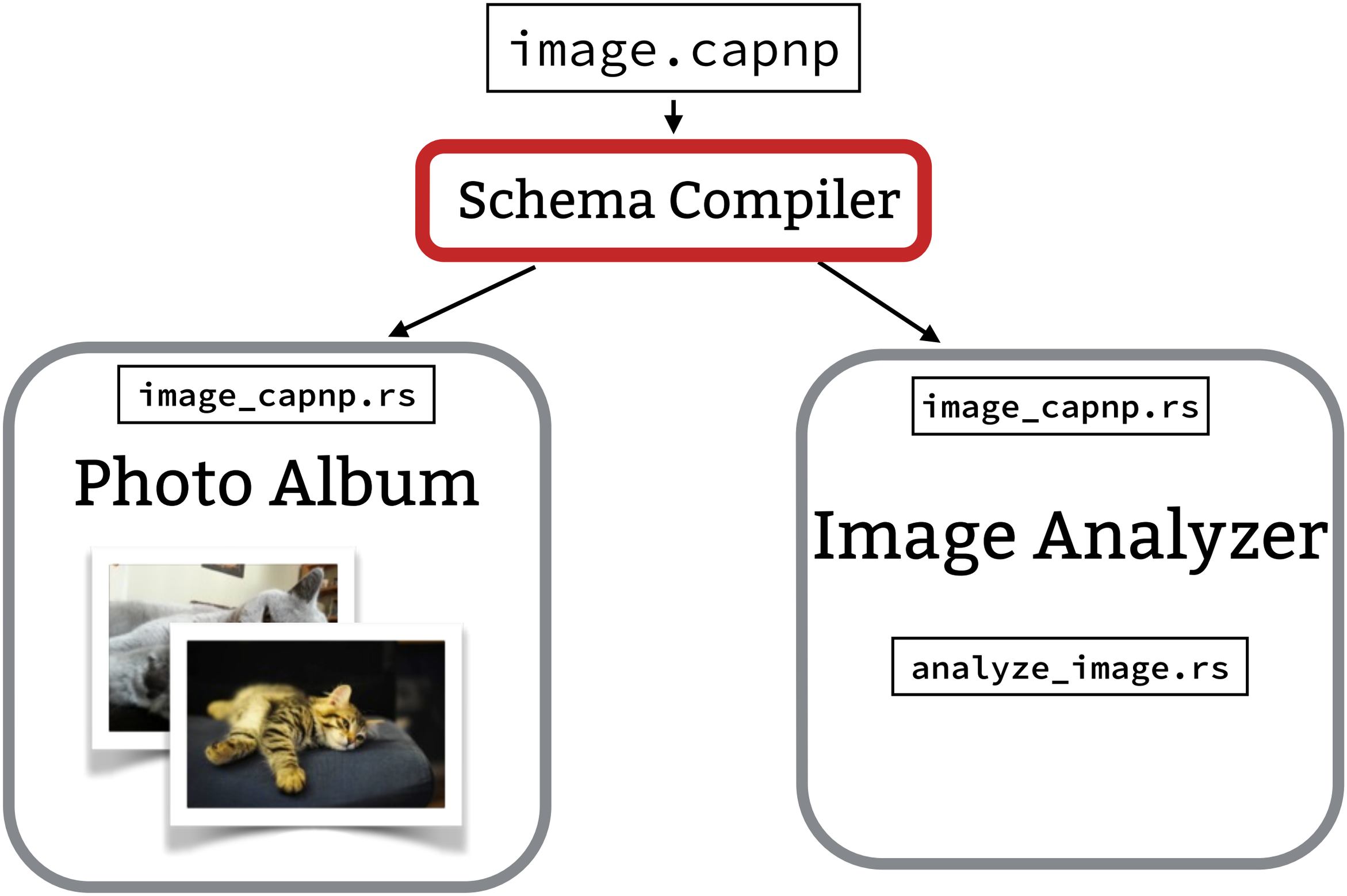


image.capnp
Schema Compiler
image_capnp.rs
Photo Album
image_capnp.rs
Image Analyzer
analyze_image.rs

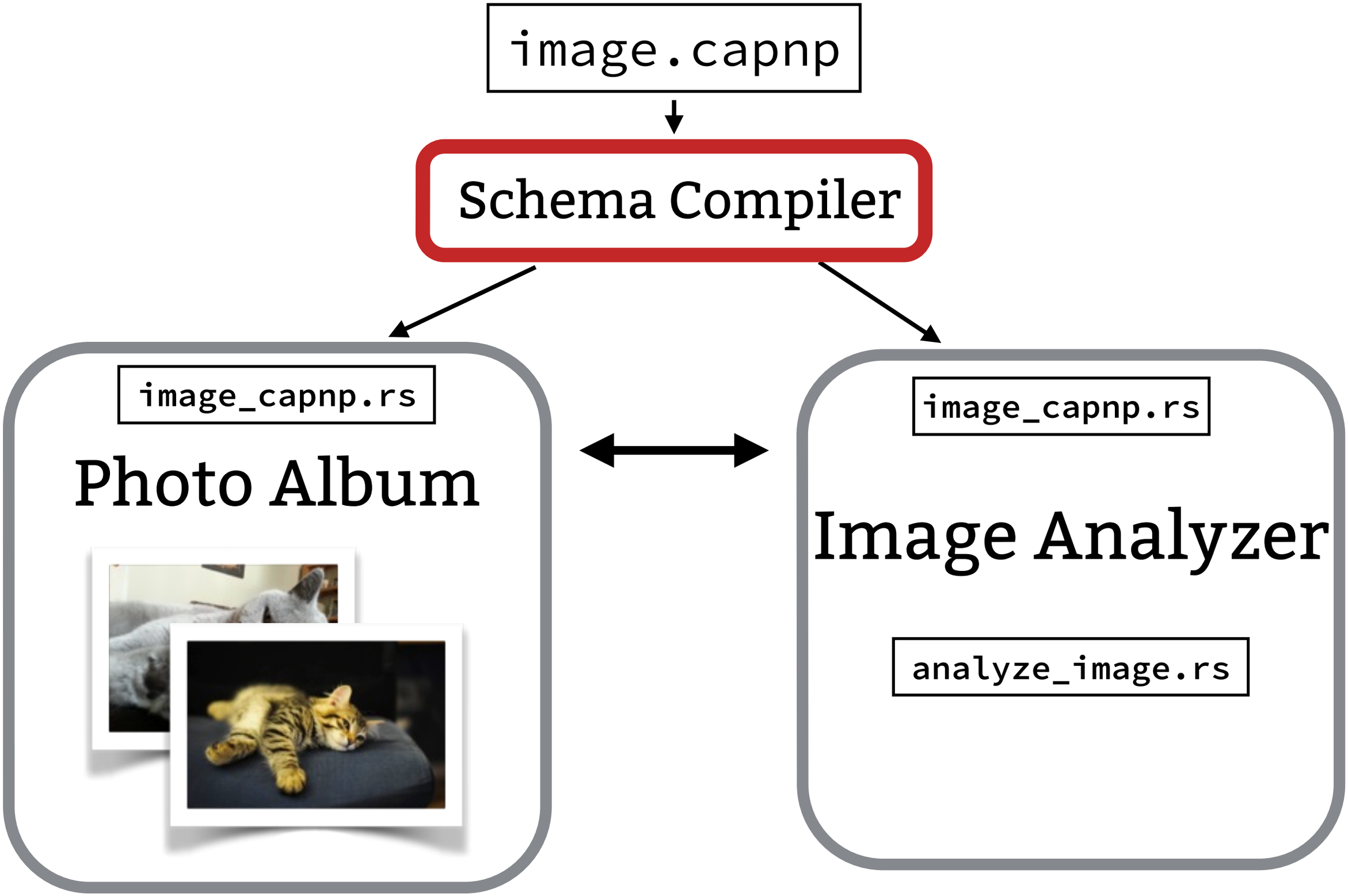


image.capnp
Schema Compiler
image_capnp.rs
Photo Album
image_capnp.rs
Image Analyzer
analyze_image.rs

data and capabilities

data and capabilities

⋮

(serialization)

data and capabilities

(serialization)

(remote procedure calls)

image.capnp

```
struct Color {
  red   @0 : UInt8;
  green @1 : UInt8;
  blue  @2 : UInt8;
}

struct Image {
  width  @0 : UInt16;
  height @1 : UInt16;
  pixels @2 : List(Color);
  # width * height pixels in row-major order
}
```

image_capnp.rs

```rust
pub mod color {
  pub struct Reader<'a> { /* ... */ }

  impl <'a> Reader<'a> {
    pub fn get_red(&self) -> u8 { /* ... */ }
    pub fn get_green(&self) -> u8 { /* ... */ }
    pub fn get_blue(&self) -> u8 { /* ... */ }
  }

  pub struct Builder<'a> { /* ... */ }

  impl <'a> Builder<'a> {
    /* ... */
  }
}
```

image_capnp.rs

```rust
pub mod color {
  pub struct Reader<'a> { /* ... */ }

  impl <'a> Reader<'a> {
    pub fn get_red(&self) -> u8 { /* ... */ }
    pub fn get_green(&self) -> u8 { /* ... */ }
    pub fn get_blue(&self) -> u8 { /* ... */ }
  }

  pub struct Builder<'a> { /* ... */ }

  impl <'a> Builder<'a> {
    /* ... */
  }
}
```

image_capnp.rs

```rust
pub mod color {
  pub struct Reader<'a> { /* ... */ }

  impl <'a> Reader<'a> {
    pub fn get_red(&self) -> u8 { /* ... */ }
    pub fn get_green(&self) -> u8 { /* ... */ }
    pub fn get_blue(&self) -> u8 { /* ... */ }
  }

  pub struct Builder<'a> { /* ... */ }

  impl <'a> Builder<'a> {
    /* ... */
  }
}
```

image_capnp.rs

```rust
pub mod color {
  pub struct Reader<'a> { /* ... */ }

  impl <'a> Reader<'a> {
    pub fn get_red(&self) -> u8 { /* ... */ }
    pub fn get_green(&self) -> u8 { /* ... */ }
    pub fn get_blue(&self) -> u8 { /* ... */ }
  }

  pub struct Builder<'a> { /* ... */ }

  impl <'a> Builder<'a> {
    /* ... */
  }
}
```

image_capnp.rs

```rust
pub mod color {
  pub struct Reader<'a> { /* ... */ }

  impl <'a> Reader<'a> {
    pub fn get_red(&self) -> u8 { /* ... */ }
    pub fn get_green(&self) -> u8 { /* ... */ }
    pub fn get_blue(&self) -> u8 { /* ... */ }
  }

  pub struct Builder<'a> { /* ... */ }

  impl <'a> Builder<'a> {
    /* ... */
  }
}
```

image_capnp.rs

```rust
pub mod color {
  pub struct Reader<'a> { /* ... */ }

  impl <'a> Reader<'a> {
    pub fn get_red(&self) -> u8 { /* ... */ }
    pub fn get_green(&self) -> u8 { /* ... */ }
    pub fn get_blue(&self) -> u8 { /* ... */ }
  }

  pub struct Builder<'a> { /* ... */ }

  impl <'a> Builder<'a> {
    /* ... */
  }
}
```

image_capnp.rs

```rust
pub mod color {
  pub struct Reader<'a> { /* ... */ }

  impl <'a> Reader<'a> {
    pub fn get_red(&self) -> u8 { /* ... */ }
    pub fn get_green(&self) -> u8 { /* ... */ }
    pub fn get_blue(&self) -> u8 { /* ... */ }
  }

  pub struct Builder<'a> { /* ... */ }

  impl <'a> Builder<'a> {
    /* ... */
  }
}
```

client code

```
pub fn pixel_at(image : image::Reader,
                x : u16,
                y : u16) -> color::Reader {

    image.get_pixels()
         .get(x as uint +
              y as uint * image.get_width() as uint)

}
```

client code

generated types

```
pub fn pixel_at(image : image::Reader,
                x : u16,
                y : u16) -> color::Reader {

    image.get_pixels()
         .get(x as uint +
              y as uint * image.get_width() as uint)

}
```

client code

generated types

```
pub fn pixel_at(image : image::Reader,
                x : u16,
                y : u16) -> color::Reader {

    image.get_pixels()
         .get(x as uint +
              y as uint * image.get_width() as uint)

}
```

generated accessors

client code

generated types

```
pub fn pixel_at(image : image::Reader,
                x : u16,
                y : u16) -> color::Reader {

    image.get_pixels()
         .get(x as uint +
              y as uint * image.get_width() as uint)

}
```

generated accessors

method of `capnp::struct_list<color::Reader>`

client code

```
pub fn average_pixel(image         : image::Reader,
                     average_pixel : color::Builder) {
    let mut red_total   : u64 = 0;
    let mut green_total : u64 = 0;
    let mut blue_total  : u64 = 0;

    for pixel in image.get_pixels().iter() {
        red_total   += pixel.get_red() as u64;
        green_total += pixel.get_green() as u64;
        blue_total  += pixel.get_blue() as u64;
    }

    let size = image.get_pixels().size() as u64;
    average_pixel.set_red((red_total / size) as u8);
    average_pixel.set_green((green_total / size) as u8);
    average_pixel.set_blue((blue_total / size) as u8);
}
```

client code

```
pub fn average_pixel(image         : image::Reader,
                     average_pixel : color::Builder) {
    let mut red_total   : u64 = 0;
    let mut green_total : u64 = 0;
    let mut blue_total  : u64 = 0;

    for pixel in image.get_pixels().iter() {
        red_total   += pixel.get_red() as u64;
        green_total += pixel.get_green() as u64;
        blue_total  += pixel.get_blue() as u64;
    }

    let size = image.get_pixels().size() as u64;
    average_pixel.set_red((red_total / size) as u8);
    average_pixel.set_green((green_total / size) as u8);
    average_pixel.set_blue((blue_total / size) as u8);
}
```

iteration over `capnp::struct_list<color::Reader>`

client code

```
pub fn average_pixel(image         : image::Reader,
                     average_pixel : color::Builder) {
    let mut red_total   : u64 = 0;
    let mut green_total : u64 = 0;
    let mut blue_total  : u64 = 0;

    for pixel in image.get_pixels().iter() {
        red_total   += pixel.get_red() as u64;
        green_total += pixel.get_green() as u64;
        blue_total  += pixel.get_blue() as u64;
    }

    let size = image.get_pixels().size() as u64;
    average_pixel.set_red((red_total / size) as u8);
    average_pixel.set_green((green_total / size) as u8);
    average_pixel.set_blue((blue_total / size) as u8);
}
```

client code

```
pub fn average_pixel(image         : image::Reader,
                     average_pixel : color::Builder) {
    let mut red_total   : u64 = 0;
    let mut green_total : u64 = 0;
    let mut blue_total  : u64 = 0;

    for pixel in image.get_pixels().iter() {
        red_total   += pixel.get_red() as u64;
        green_total += pixel.get_green() as u64;
        blue_total  += pixel.get_blue() as u64;
    }

    let size = image.get_pixels().size() as u64;
    average_pixel.set_red((red_total / size) as u8);
    average_pixel.set_green((green_total / size) as u8);
    average_pixel.set_blue((blue_total / size) as u8);
}
```

setter methods

image.capnp

(continued)

```
struct AnalysisResult {
  objects @0 : List(DetectedObject);
}

struct DetectedObject {
  union {
    person @0 : Person;
    cat    @1 : Cat;
  }
  boundingBox @2 : AxisAlignedBoundingBox;
}
```

client code

```
pub fn print_object(object : detected_object::Reader) {
    match object.which() {
        Some(detected_object::Person(p)) => {
            println!("  a person of height {}", p.get_height());
        }
        Some(detected_object::Cat(c)) => {
            println!("  a cat with {} colors",
                     c.get_fur_colors().size());
        }
        None => println!("  unknown object")
    }
}
```

client code

```
pub fn print_object(object : detected_object::Reader) {
    match object.which() {
        Some(detected_object::Person(p)) => {
            println!("  a person of height {}", p.get_height());
        }
        Some(detected_object::Cat(c)) => {
            println!("  a cat with {} colors",
                     c.get_fur_colors().size());
        }
        None => println!("  unknown object")
    }
}
```

method of `person::Reader`

client code

```
pub fn print_object(object : detected_object::Reader) {
    match object.which() {
        Some(detected_object::Person(p)) => {
            println!("  a person of height {}", p.get_height());
        }
        Some(detected_object::Cat(c)) => {
            println!("  a cat with {} colors",
                     c.get_fur_colors().size());
        }
        None => println!("  unknown object")
    }
}
```

method of `person::Reader`

method of `cat::Reader`

client code

```
pub fn print_object(object : detected_object::Reader) {
    match object.which() {
        Some(detected_object::Person(p)) => {
            println!("  a person of height {}", p.get_height());
        }
        Some(detected_object::Cat(c)) => {
            println!("  a cat with {} colors",
                     c.get_fur_colors().size());
        }
        None => println!("  unknown object")
    }
}
```

method of `person::Reader`

method of `cat::Reader`

must deal with future additions

image.capnp

(continued)

```
struct DetectedObject {
  union {
    person @0 : Person;
    cat    @1 : Cat;
    dog    @3 : Dog;
  }
  boundingBox @2 : AxisAlignedBoundingBox;
}
```

image.capnp

(continued)

```
struct DetectedObject {
  union {
    person @0 : Person;
    cat    @1 : Cat;
    dog    @3 : Dog;
  }
  boundingBox @2 : AxisAlignedBoundingBox;
}
```

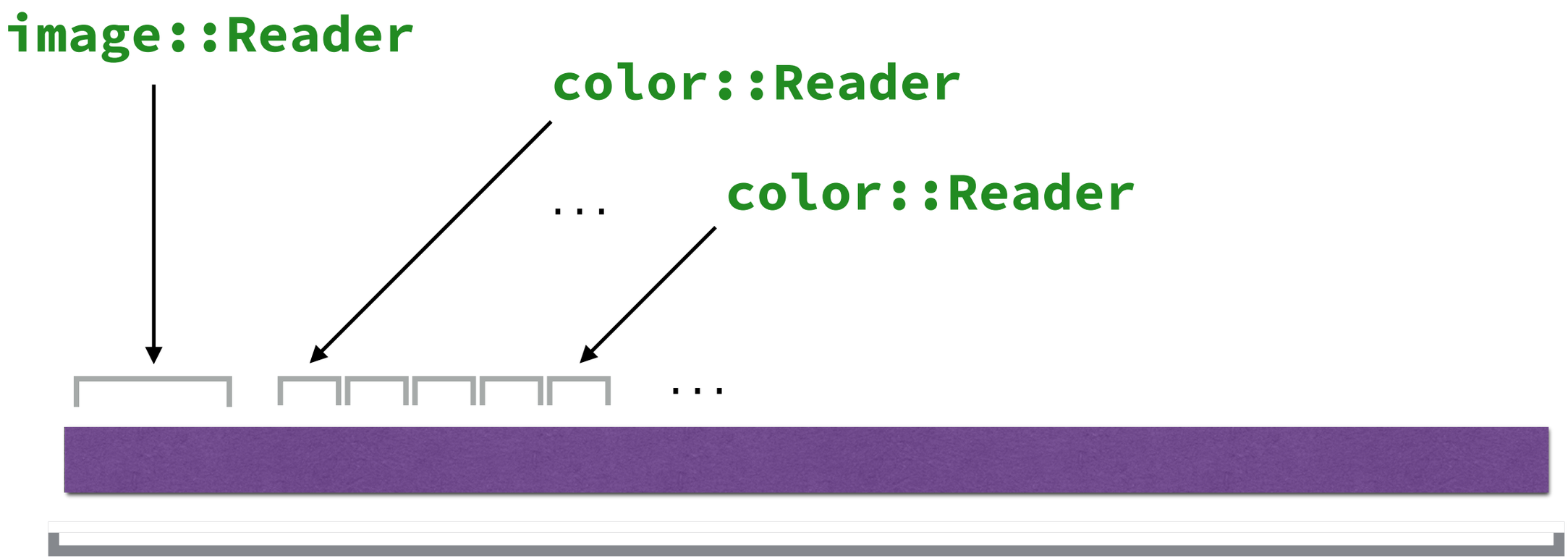
new variant

Cap’n Proto makes data **mobile**

uniformity of representation
allows zero-copy
`mmap()`-ready serialization

MessageReader


image::Reader
MessageReader



MessageReader

```rust
fn pixel_at<'a>(image : image::Reader<'a>,
                x : u16,
                y : u16) -> color::Reader<'a> {

    /* ... */

}
```

```rust
fn pixel_at<'a>(image : image::Reader<'a>,
                x : u16,
                y : u16) -> color::Reader<'a> {

    /* ... */

}
```

lifetime variables

```rust
fn pixel_at<'a>(image : image::Reader<'a>,
                x : u16,
                y : u16) -> color::Reader<'a> {

    /* ... */

}
```

lifetime variables

```rust
fn pixel_at<'a>(image : image::Reader<'a>,
                x : u16,
                y : u16) -> color::Reader<'a> {

   ::capnp::MallocMessageBuilder::new_default()
        .init_root::<color::Builder>().as_reader()
}
```

```
fn pixel_at<'a>(image : image::Reader<'a>,
                x : u16,
                y : u16) -> color::Reader<'a> {

   ::capnp::MallocMessageBuilder::new_default()
        .init_root::<color::Builder>().as_reader()
}
```

```
 error: borrowed value does not live long enough
::capnp::MallocMessageBuilder::new_default()
^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```rust
fn find_match<'a,'b>(image : image::Reader<'a>,
                     color : color::Reader<'b>)
                     -> color::Reader<'a> {
    /* ... */
}
```

```rust
fn find_match<'a,'b>(image : image::Reader<'a>,
                     color : color::Reader<'b>)
                     -> color::Reader<'a> {
    return color;
}
```

```rust
fn find_match<'a,'b>(image : image::Reader<'a>,
                     color : color::Reader<'b>)
                     -> color::Reader<'a> {
    return color;
}
```

```
error: mismatched types:
   expected `::image_capnp::color::Reader<'a>`,
   found `::image_capnp::color::Reader<'b>` (lifetime mismatch)
```

```
pub fn build() {

    let mut message_builder =
        ::capnp::MallocMessageBuilder::new_default();

    let result : analysis_result::Builder =
        message_builder.init_root();
    /* ... */

}
```

```
pub fn build() {

    let mut message_builder =
        ::capnp::MallocMessageBuilder::new_default();

    let result : analysis_result::Builder =
        message_builder.init_root();
    /* ... */

}
```

```rust
pub fn build_wrong() {

    let mut message_builder =
        ::capnp::MallocMessageBuilder::new_default();

    let result : analysis_result::Builder =
        message_builder.init_root();

    let another_result : analysis_result::Builder =
        message_builder.init_root();

    result.init_objects(0);
    another_result.init_objects(1);

    /* ... */
}
```

```rust
pub fn build_wrong() {

    let mut message_builder =
        ::capnp::MallocMessageBuilder::new_default();

    let result : analysis_result::Builder =
        message_builder.init_root();

    let another_result : analysis_result::Builder =
        message_builder.init_root();

    result.init_objects(0);
    another_result.init_objects(1);

    /* ... */
}
```

```
pub fn build_wrong() {

    let mut message_builder =
        ::capnp::MallocMessageBuilder::new_default();

    let result : analysis_result::Builder =
        message_builder.init_root();

    let another_result : analysis_result::Builder =
        message_builder.init_root();

    result.init_objects(0);
    another_result.init_objects(1);

    /* ... */
}
```

```rust
pub fn build_wrong() {

    let mut message_builder =
        ::capnp::MallocMessageBuilder::new_default();

    let result : analysis_result::Builder =
        message_builder.init_root();

    let another_result : analysis_result::Builder =
        message_builder.init_root();

    result.init_objects(0);
    another_result.init_objects(1);

    /* ... */
}
```

```
error: cannot borrow `message_builder` as mutable more than once at a time
```

capnproto-rust

capabilities

a **capability** is a **reference** to an immobile, possibly remote **object**

a **capability** is a **reference** to an immobile, possibly remote **object**

plus authority to use that object

image.capnp

```
interface ImageAnalyzer {
   analyze @0 Image -> AnalysisResult;
}
```

image.capnp

```
interface ImageAnalyzer {
   analyze @0 Image -> AnalysisResult;
}
```

image_capnp.rs

```rs
pub mod image_analyzer {
  pub trait Server  {
    fn analyze<'a>(&mut self, AnalyzeContext<'a>);
  }

  pub type AnalyzeContext<'a> = /* ... */
}
```

concurrency

tasks in Rust
cannot cause data races

channels allow communication between tasks

Communicating Sequential Processes

runtime implementation code

```
spawn(proc () {
    loop {
        match receiver.recv_opt() {
            Err(_) => break,
            Ok(x) => server.dispatch_call(x),
        }
    }
});
```

client code

```
struct ObjectDetectorImpl;

impl object_detector::Server for ObjectDetectorImpl {
    fn analyze(&mut self,
               mut context : object_detector::AnalyzeContext) {
        let (params, results) = context.get();
        do_magic_analysis(params, results);
        context.done();
    }
}
```

comparison to C++

capabilities are **not** data

Photo Album

Image Analyzer

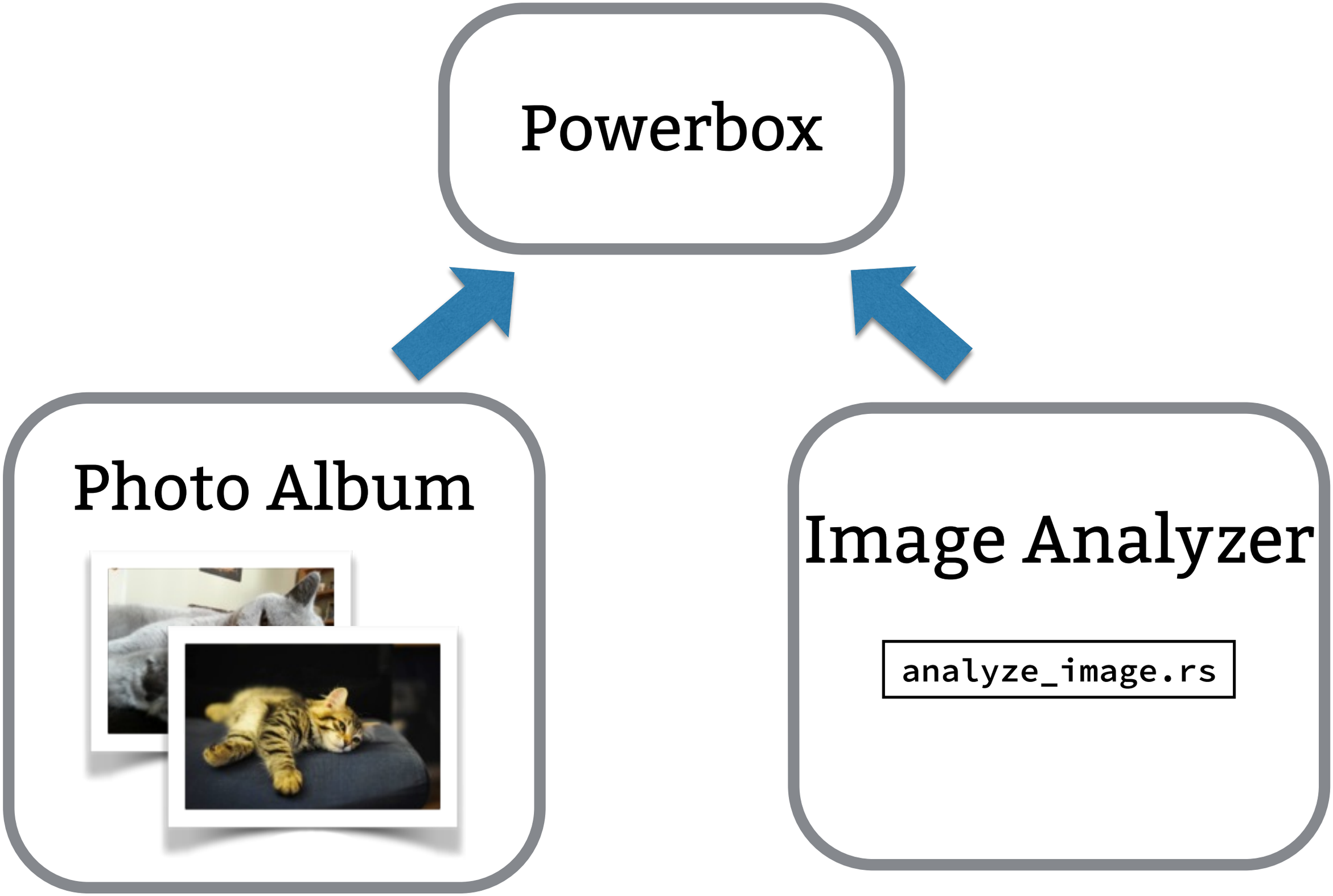
`analyze_image.rs`

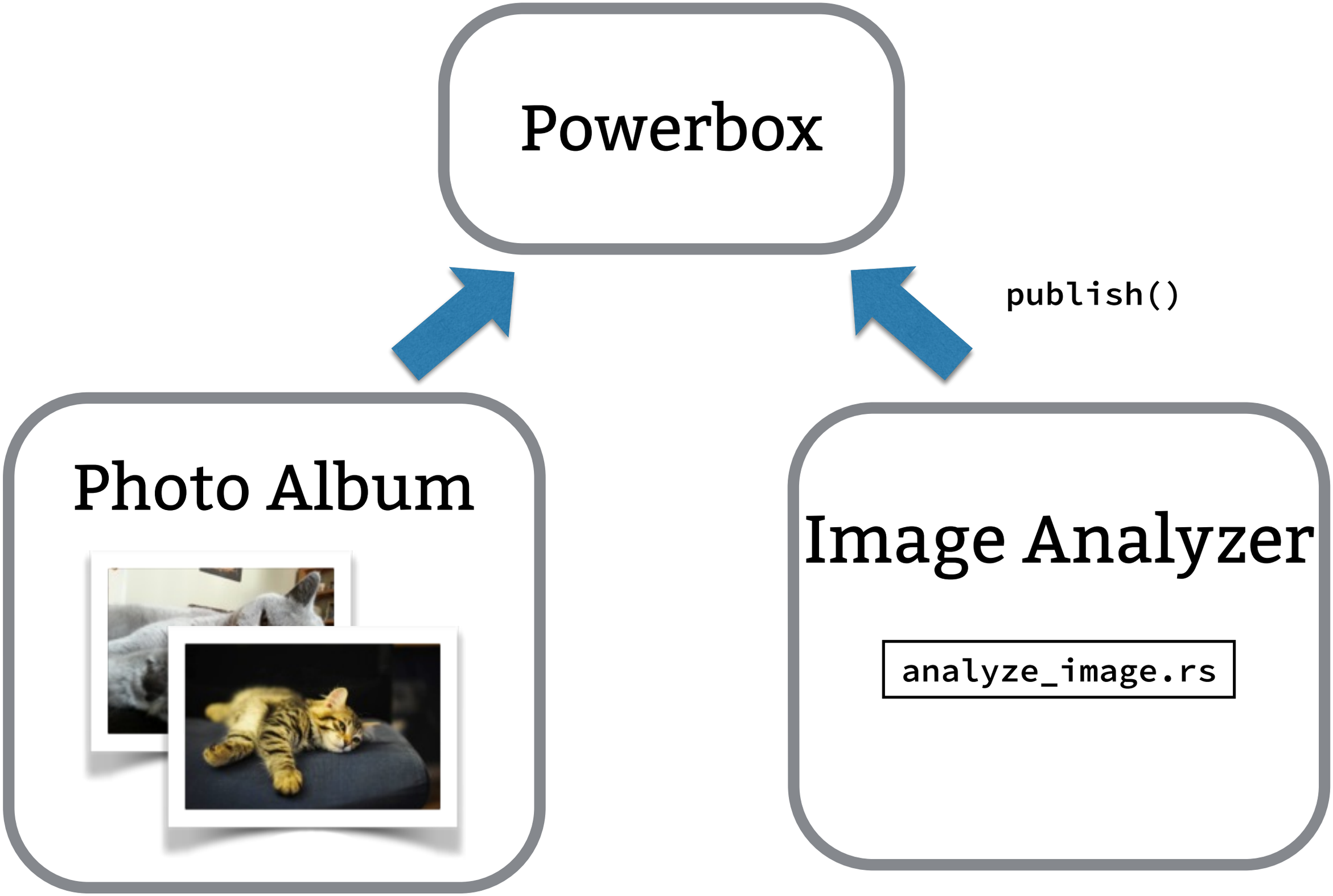
Powerbox

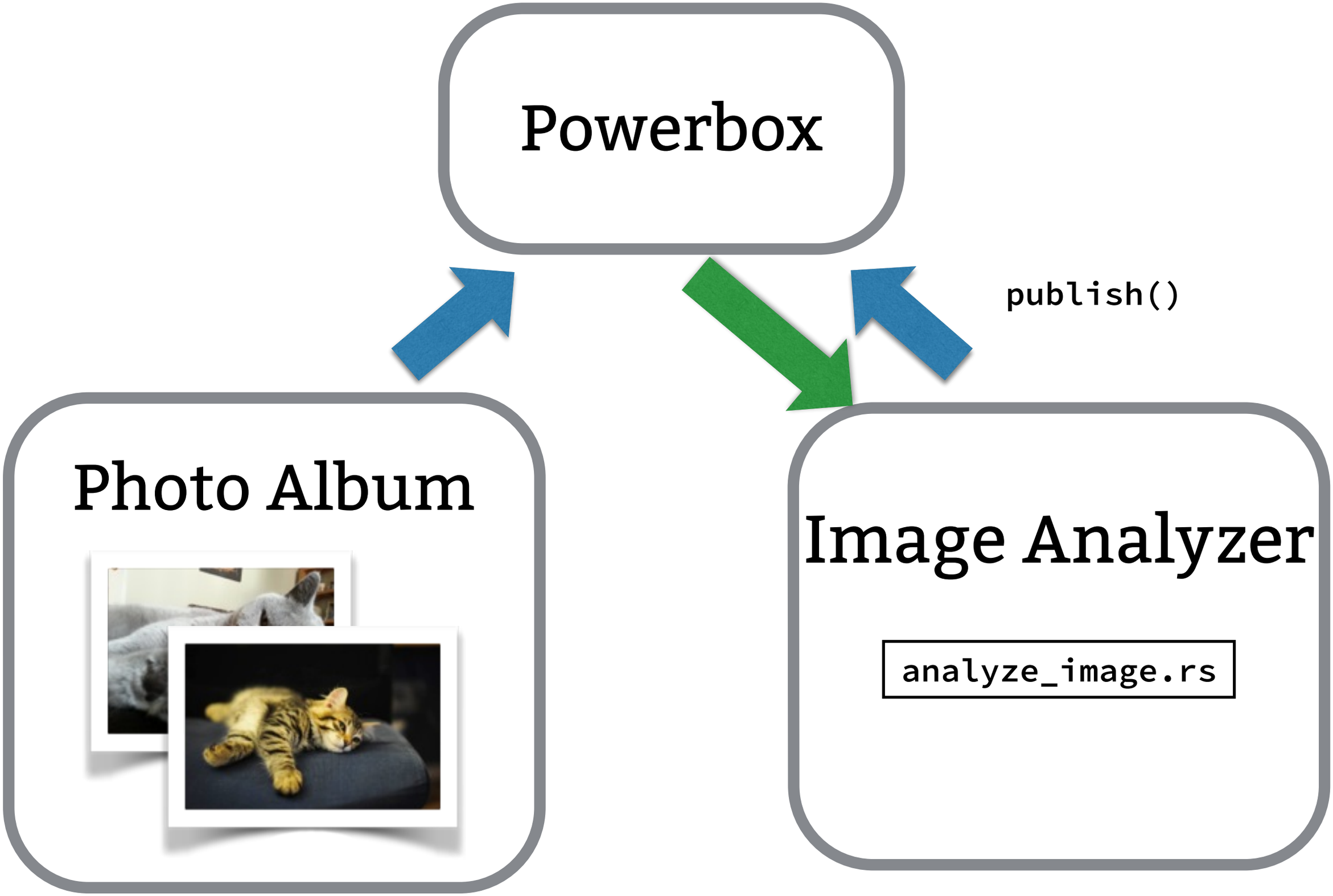
Photo Album

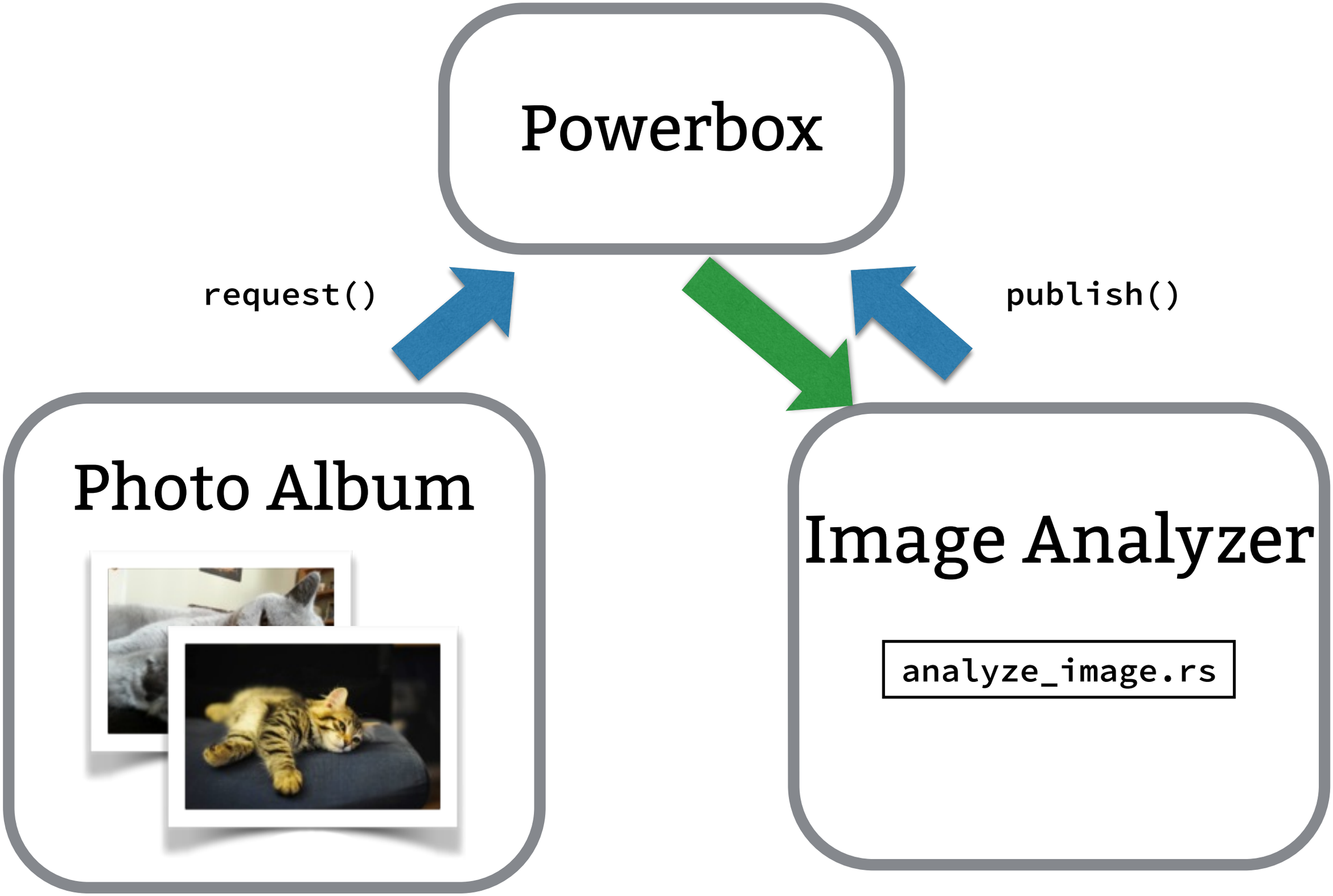
Image Analyzer

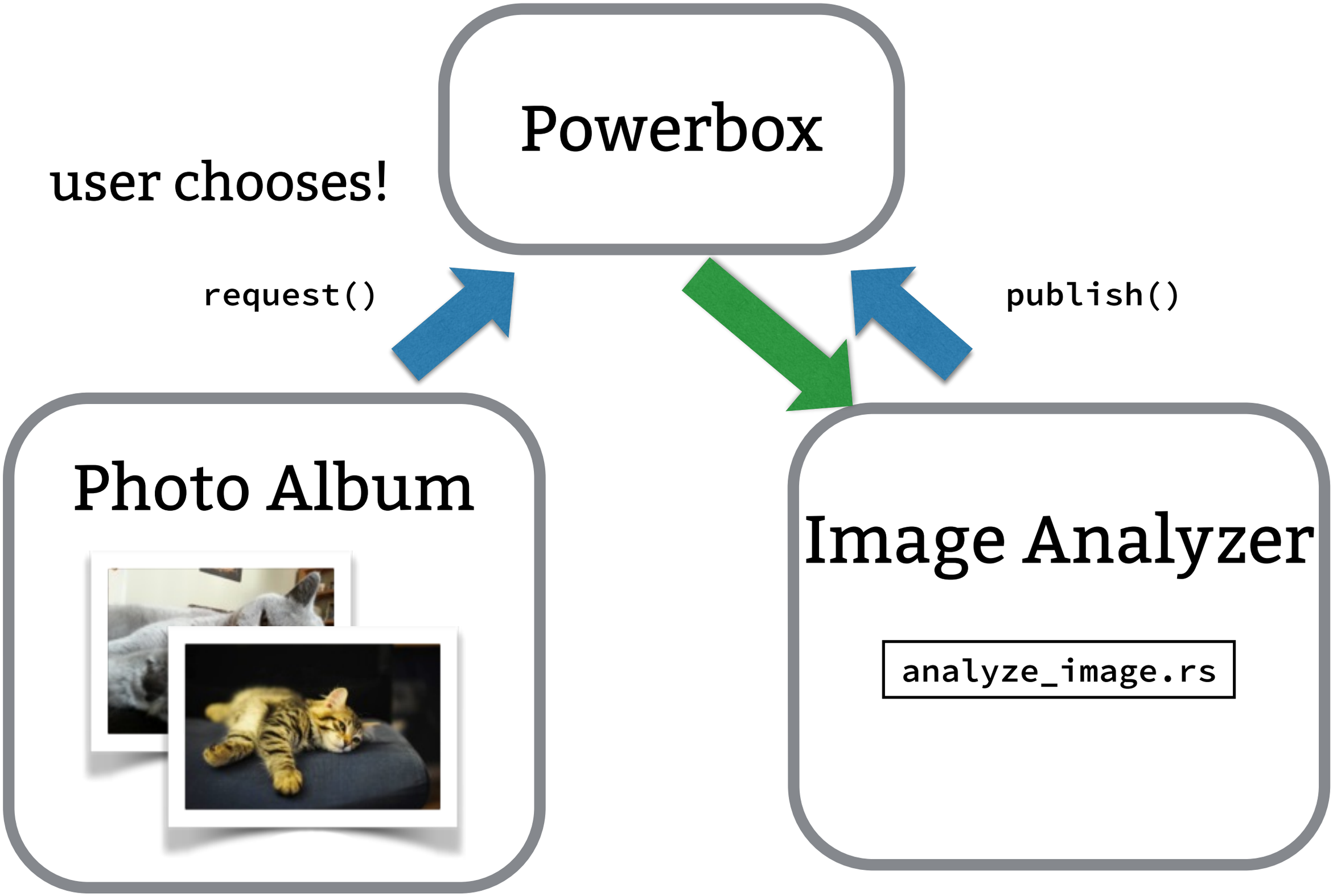
`analyze_image.rs`

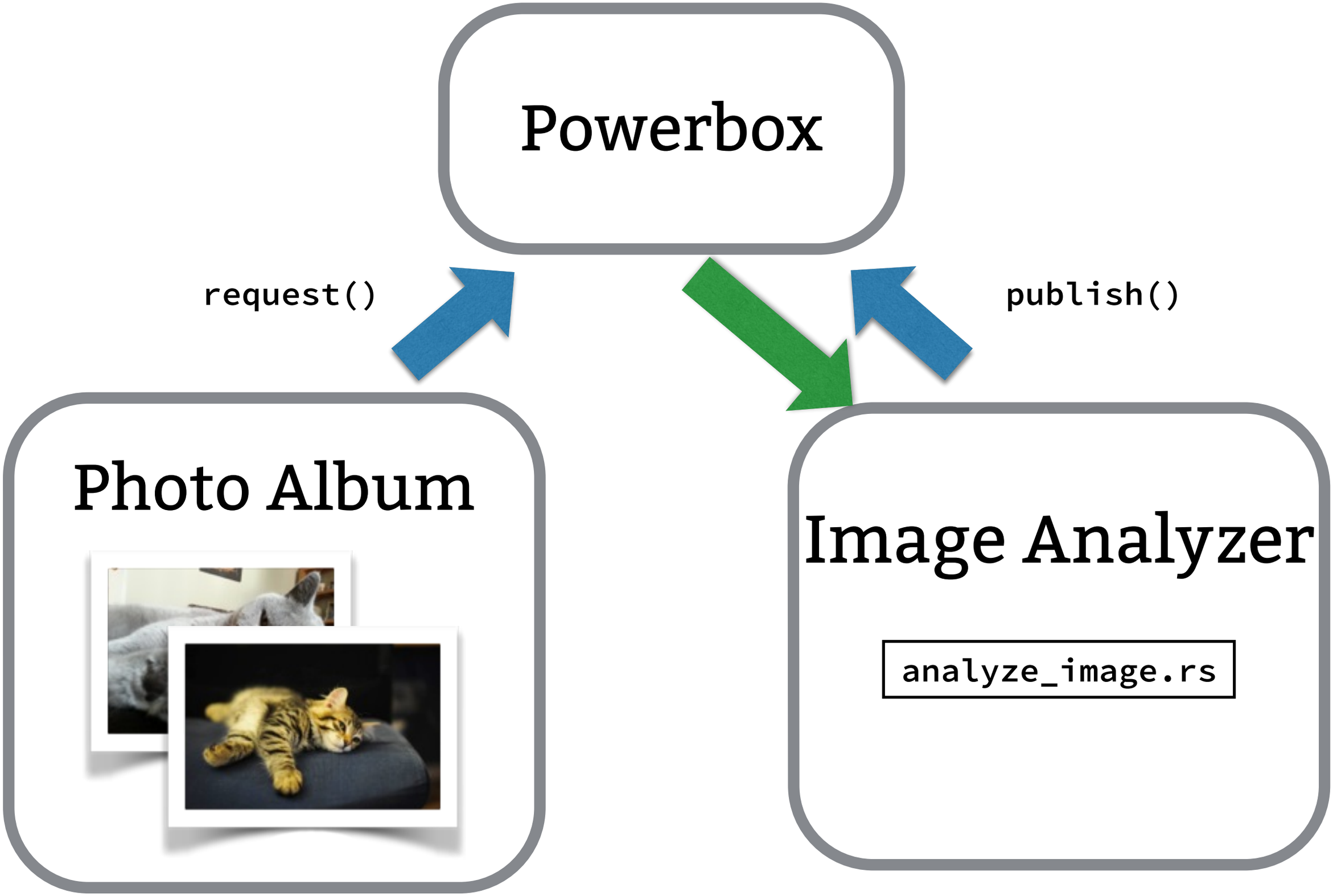

Powerbox
Photo Album
Image Analyzer
analyze_image.rs

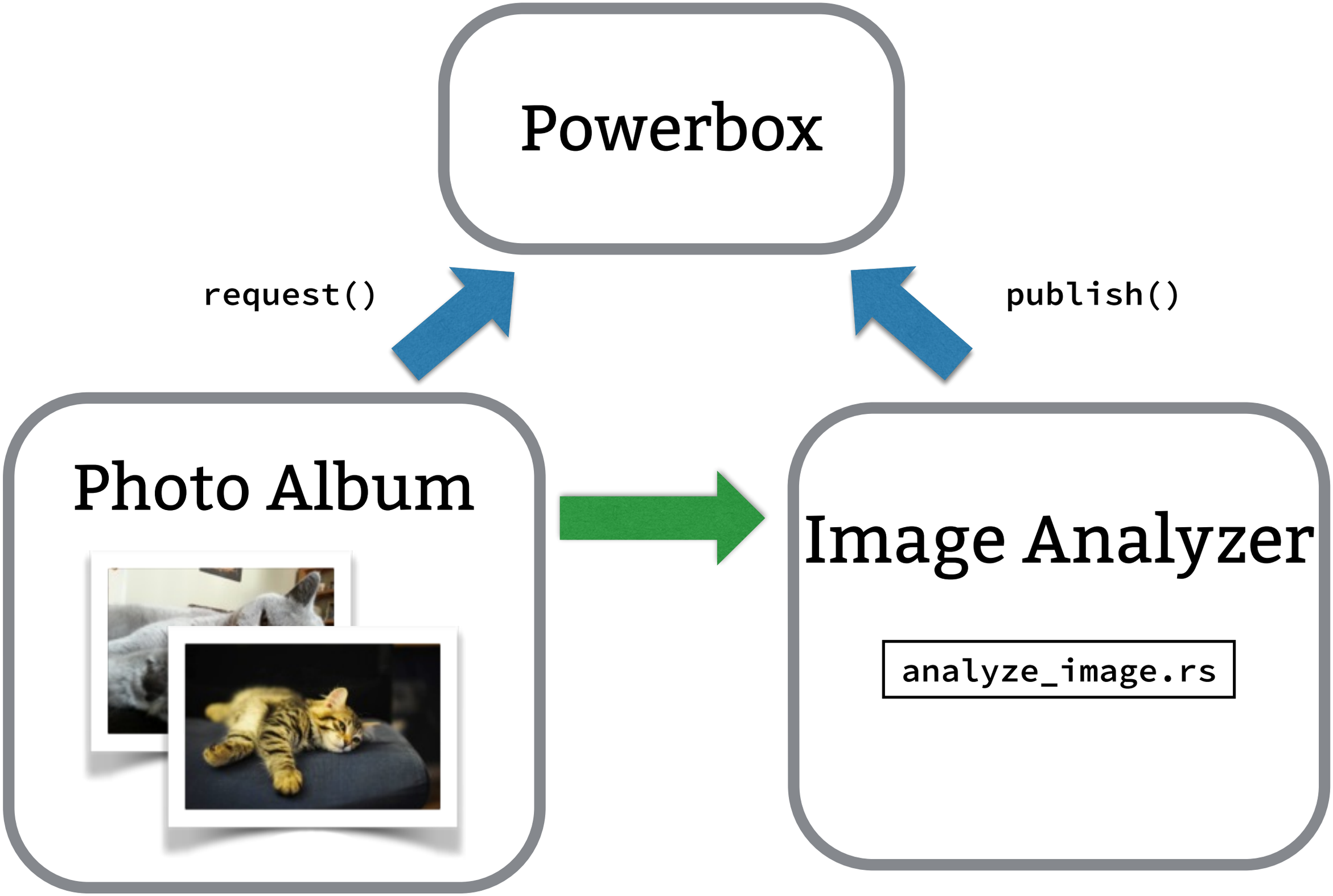

Powerbox
publish()
Photo Album
Image Analyzer
analyze_image.rs

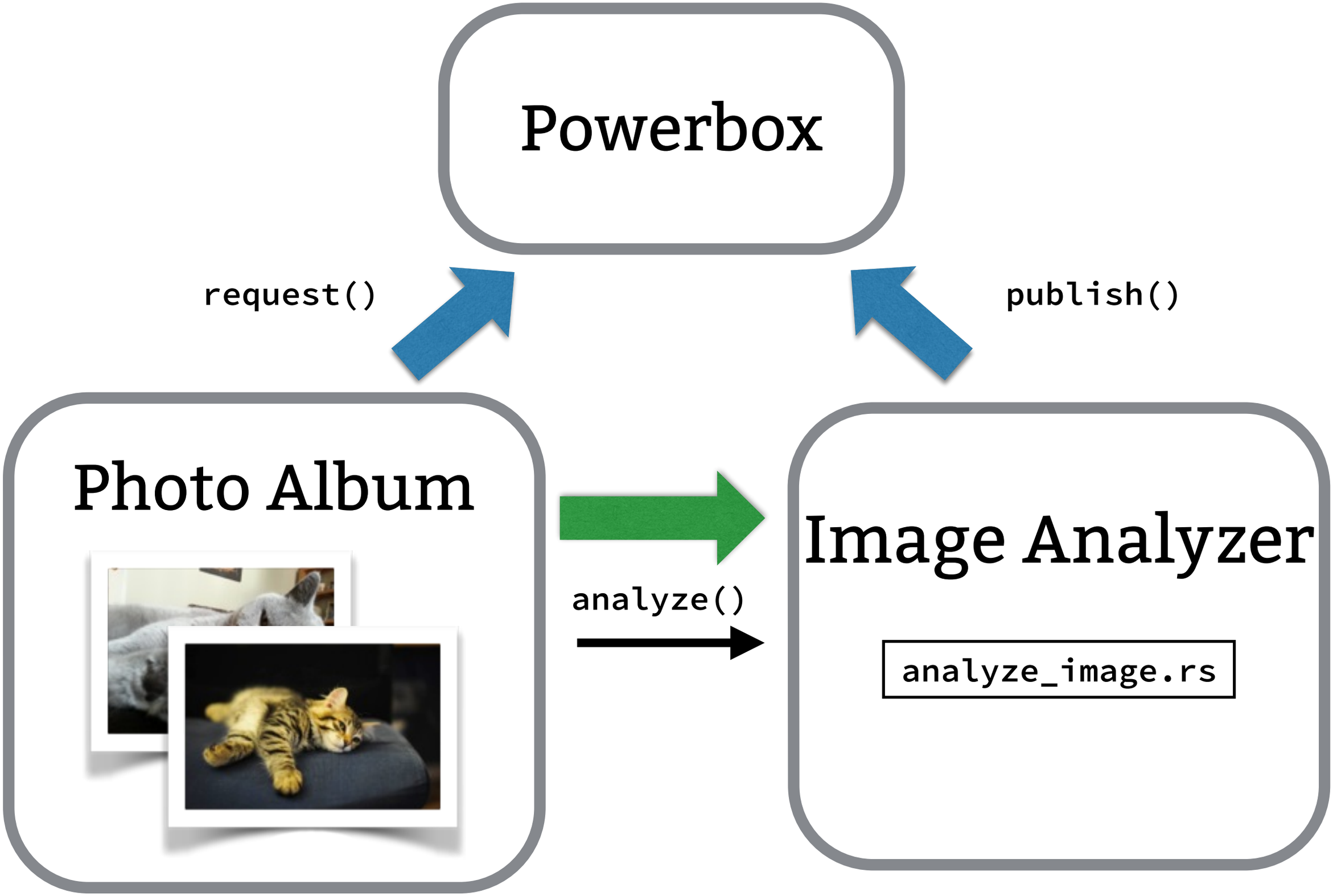

Powerbox
publish()
Photo Album
Image Analyzer
analyze_image.rs

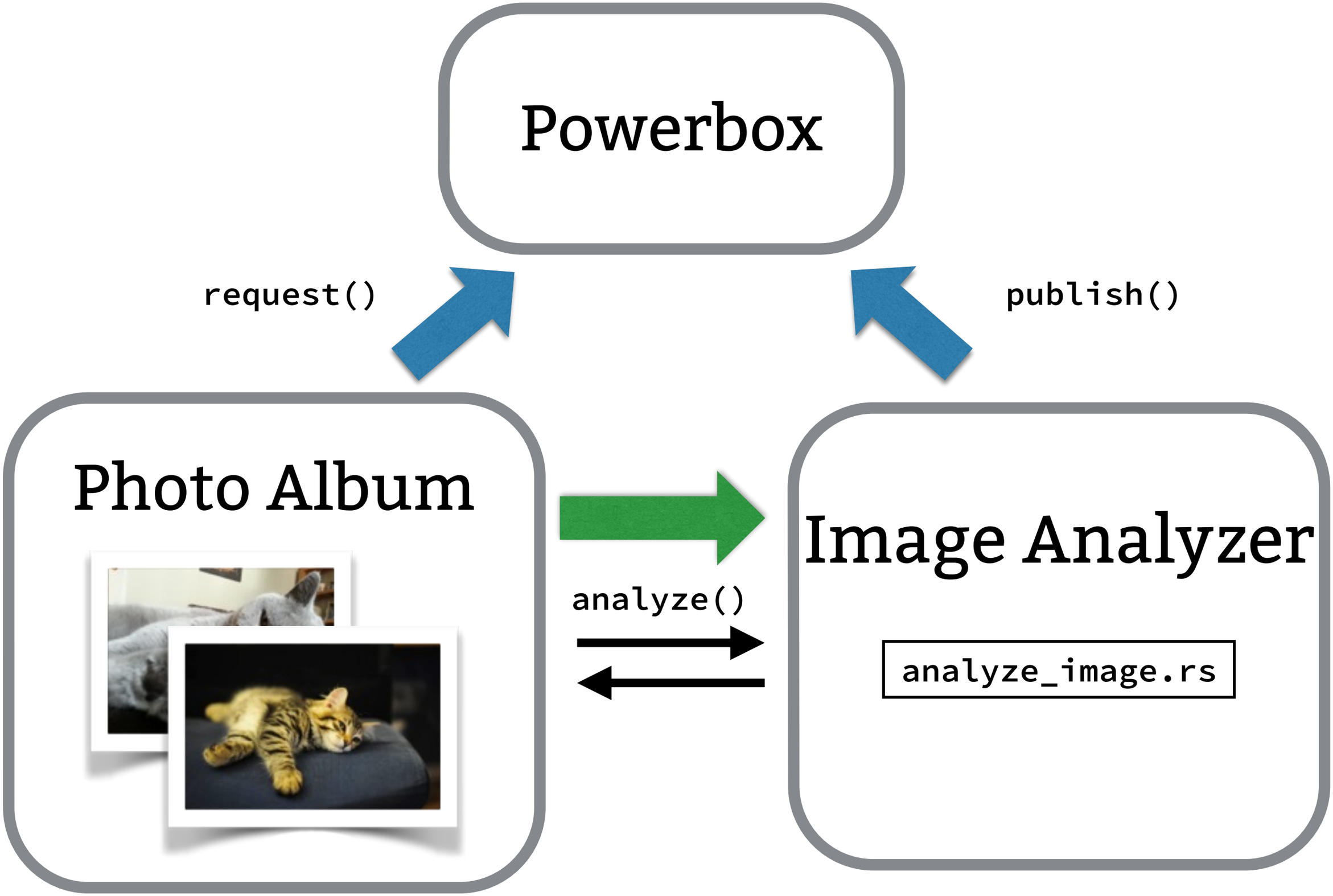

Powerbox
request()
publish()
Photo Album
Image Analyzer
analyze_image.rs


Powerbox
user chooses!
request()
publish()
Photo Album
Image Analyzer
analyze_image.rs


Powerbox
request()
publish()
Photo Album
Image Analyzer
analyze_image.rs


Powerbox
request()
publish()
Photo Album
Image Analyzer
analyze_image.rs


Powerbox
request()
publish()
Photo Album
analyze()
Image Analyzer
analyze_image.rs


Powerbox
request()
publish()
Photo Album
Image Analyzer
analyze()
analyze_image.rs

https://github.com/dwrensha/capnproto-rust

https://github.com/dwrensha/capnp-rpc-rust