Becoming a Logger Node: Native Consumption of zeek Logs

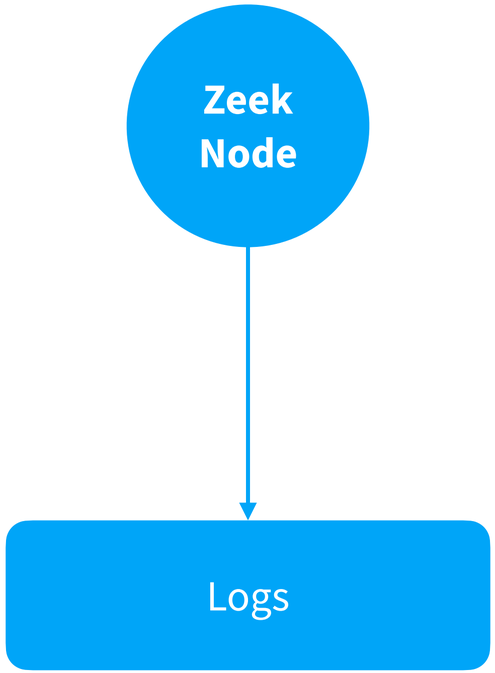
Matthias Vallentin

TENZIR

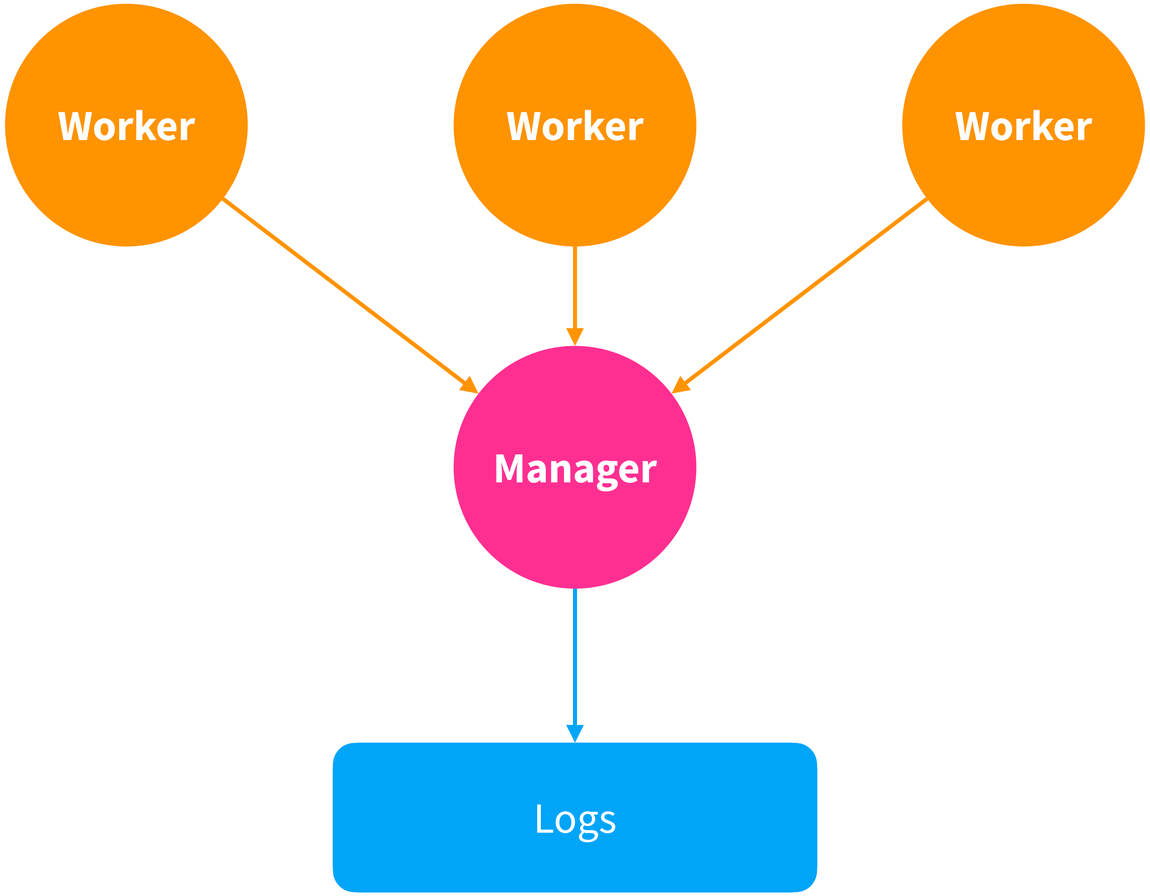
Zeek Week '21

TLP: WHITE

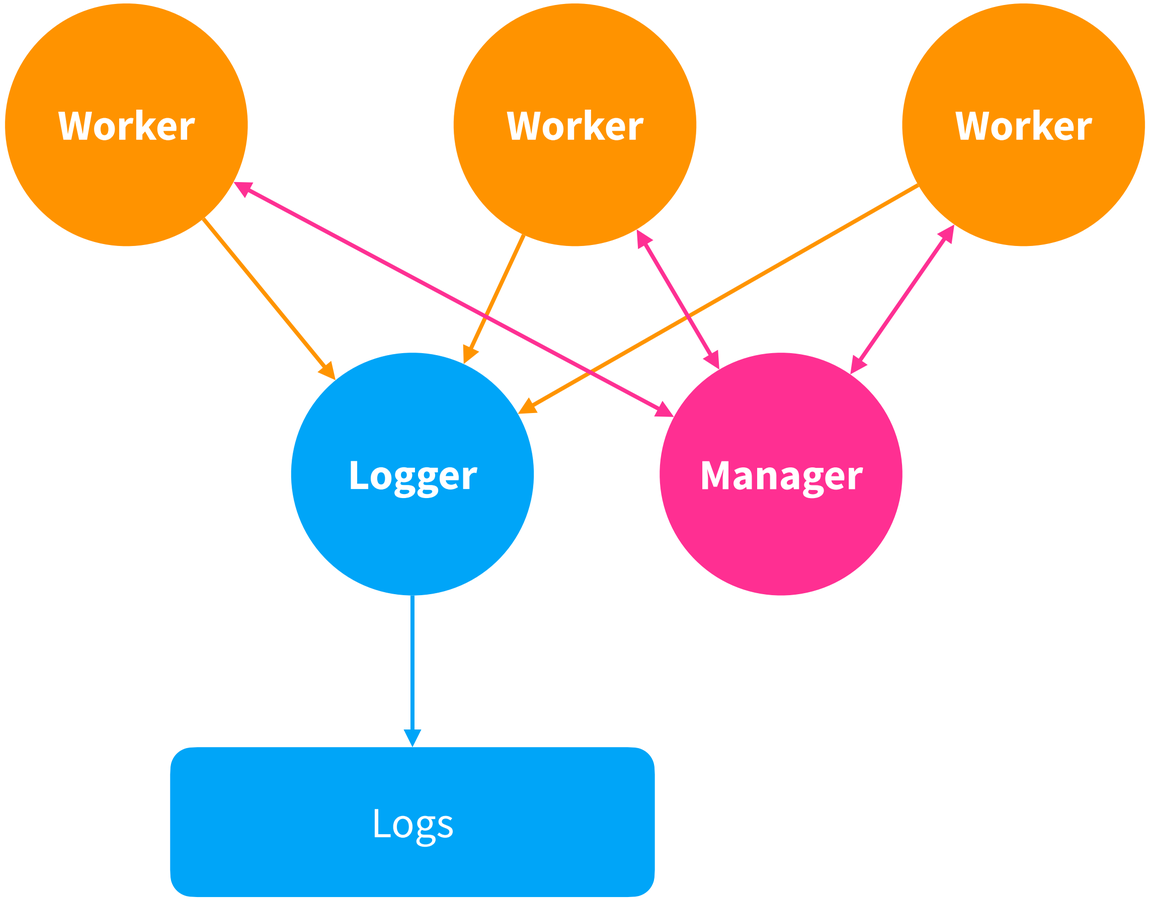
Single Node



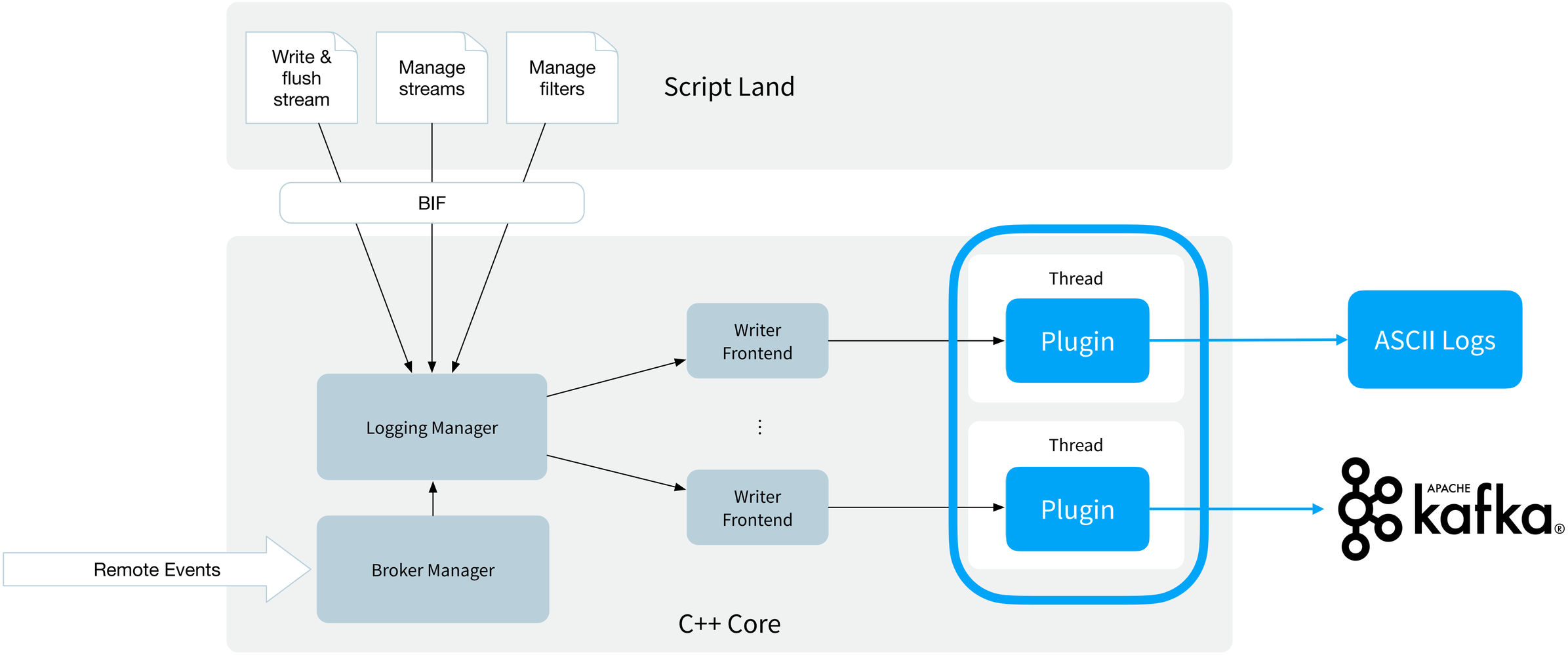
Simple Cluster



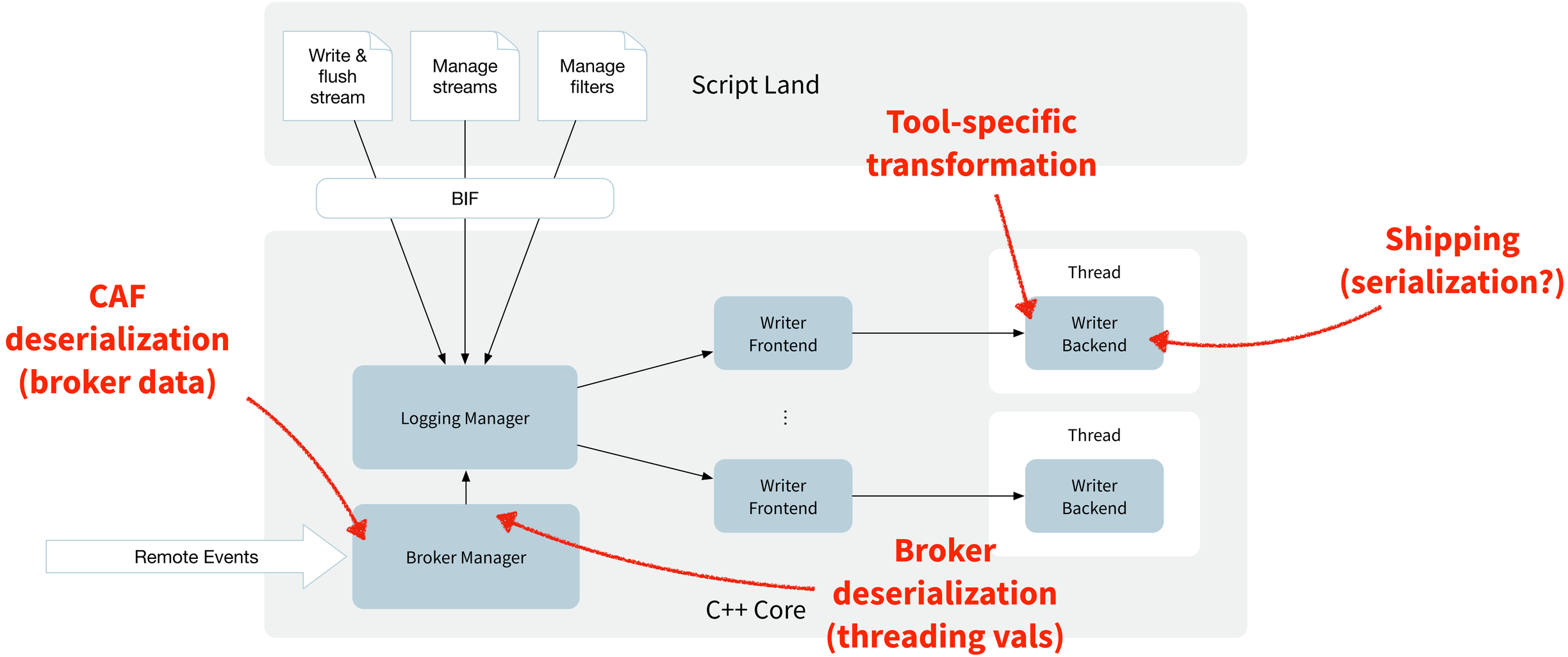
Cluster with Logger



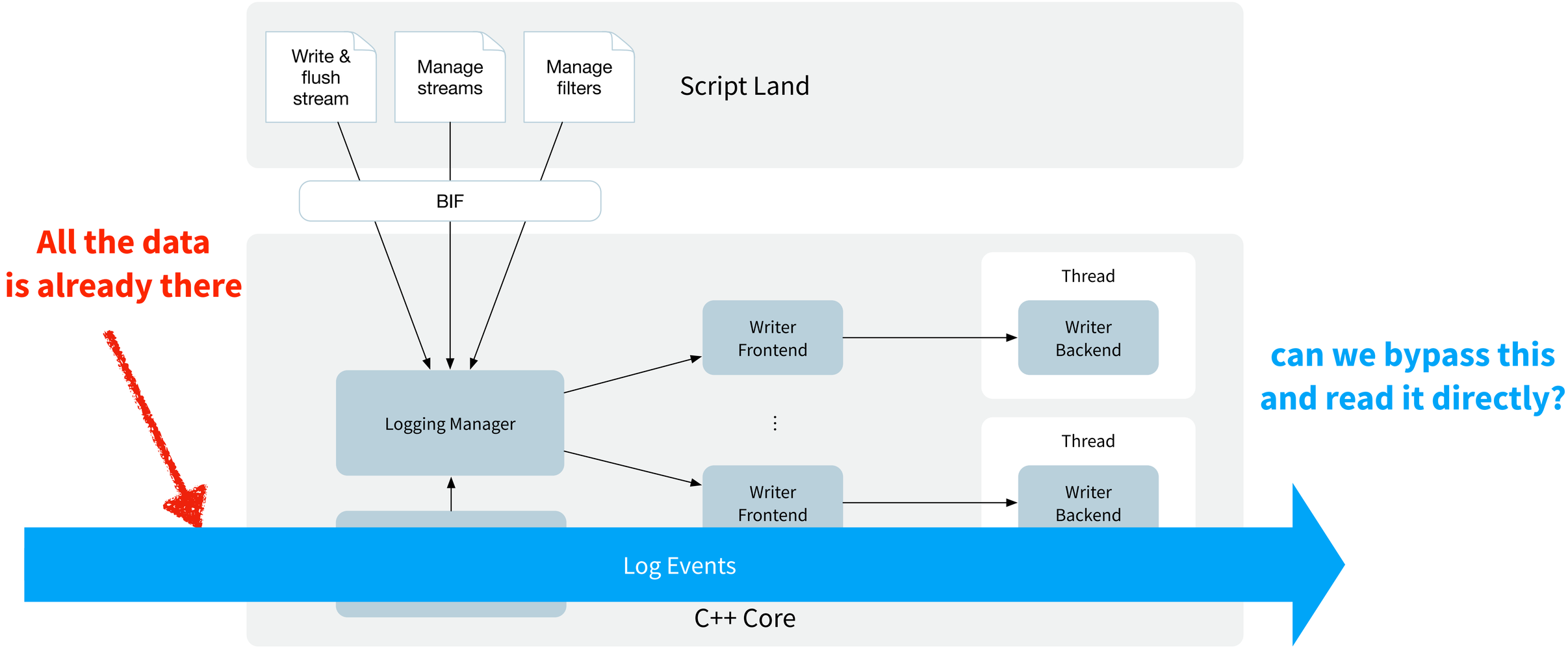
Remote Logging Architecture



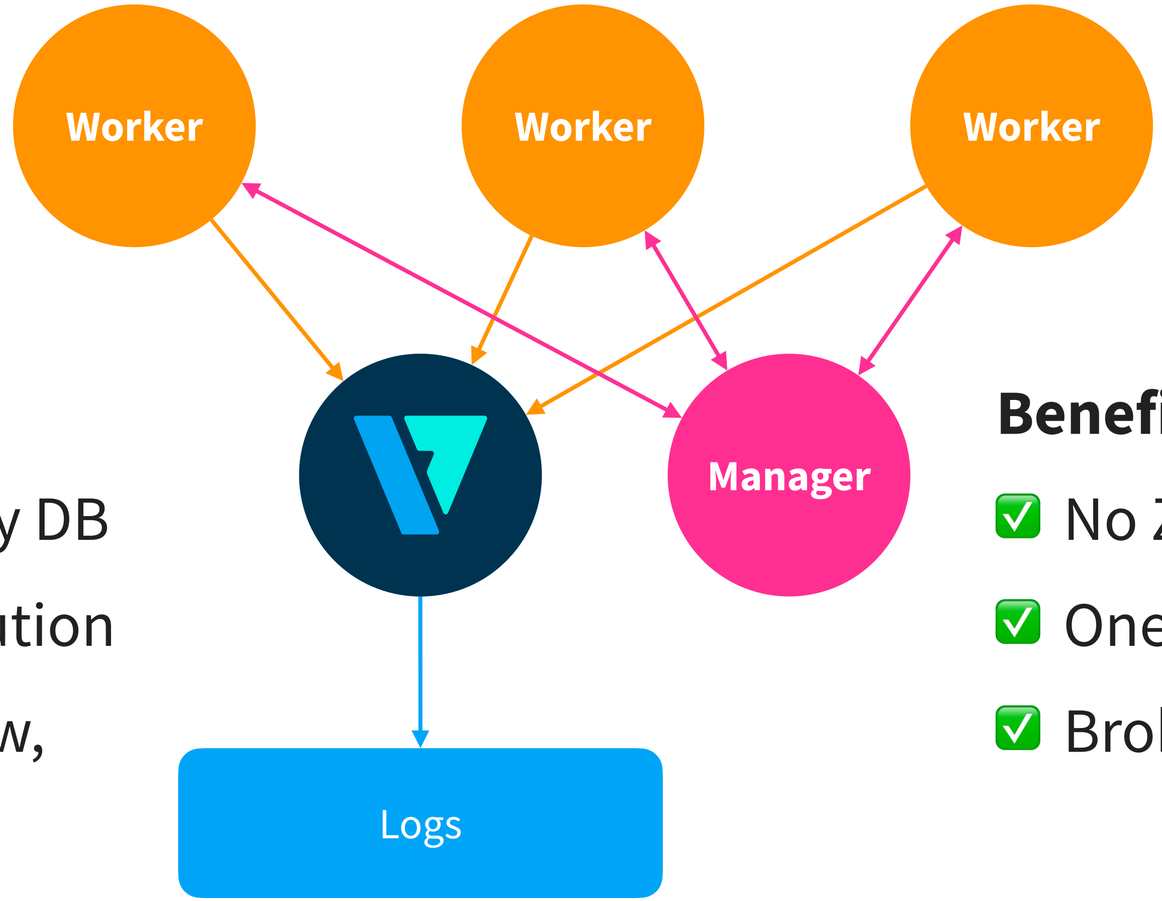
Logger Overhead(s)



Data Access


Write & flush stream
Manage streams
Manage filters
Script Land
BIF
All the data is already there
Logging Manager
Writer Frontend
Writer Frontend
Thread
Writer Backend
Thread
Writer Backend
can we bypass this and read it directly?
Log Events
C++ Core

Cluster with Logger



VAST

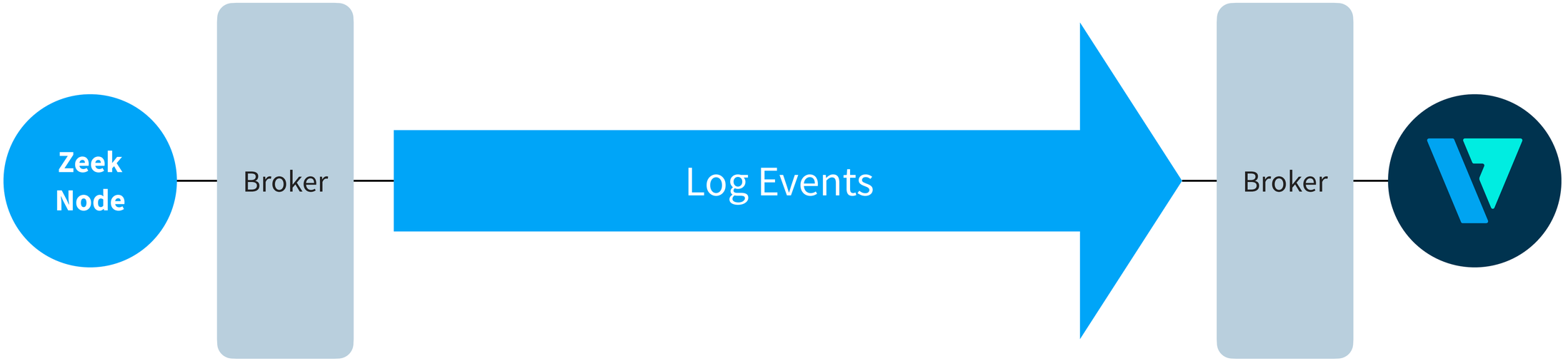
- High-volume telemetry DB
- Security content execution
- Zeek, Suricata, NetFlow, PCAP, JSON, CSV, ...

Benefits

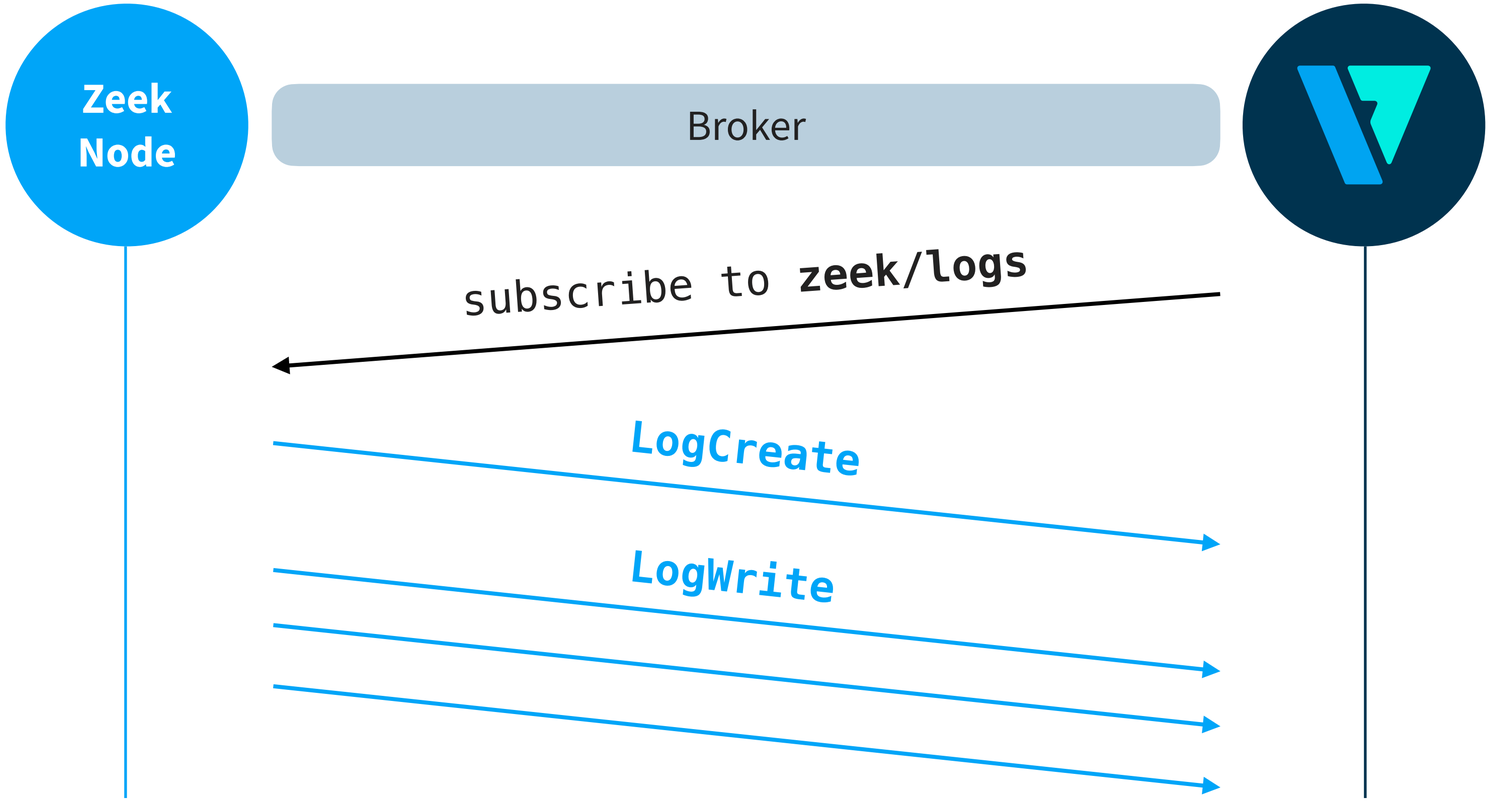
- ✅ No Zeek plugin needed
- ✅ One less Zeek node
- ✅ Broker back channel

👉 **`https://github.com/tenzir/vast`**

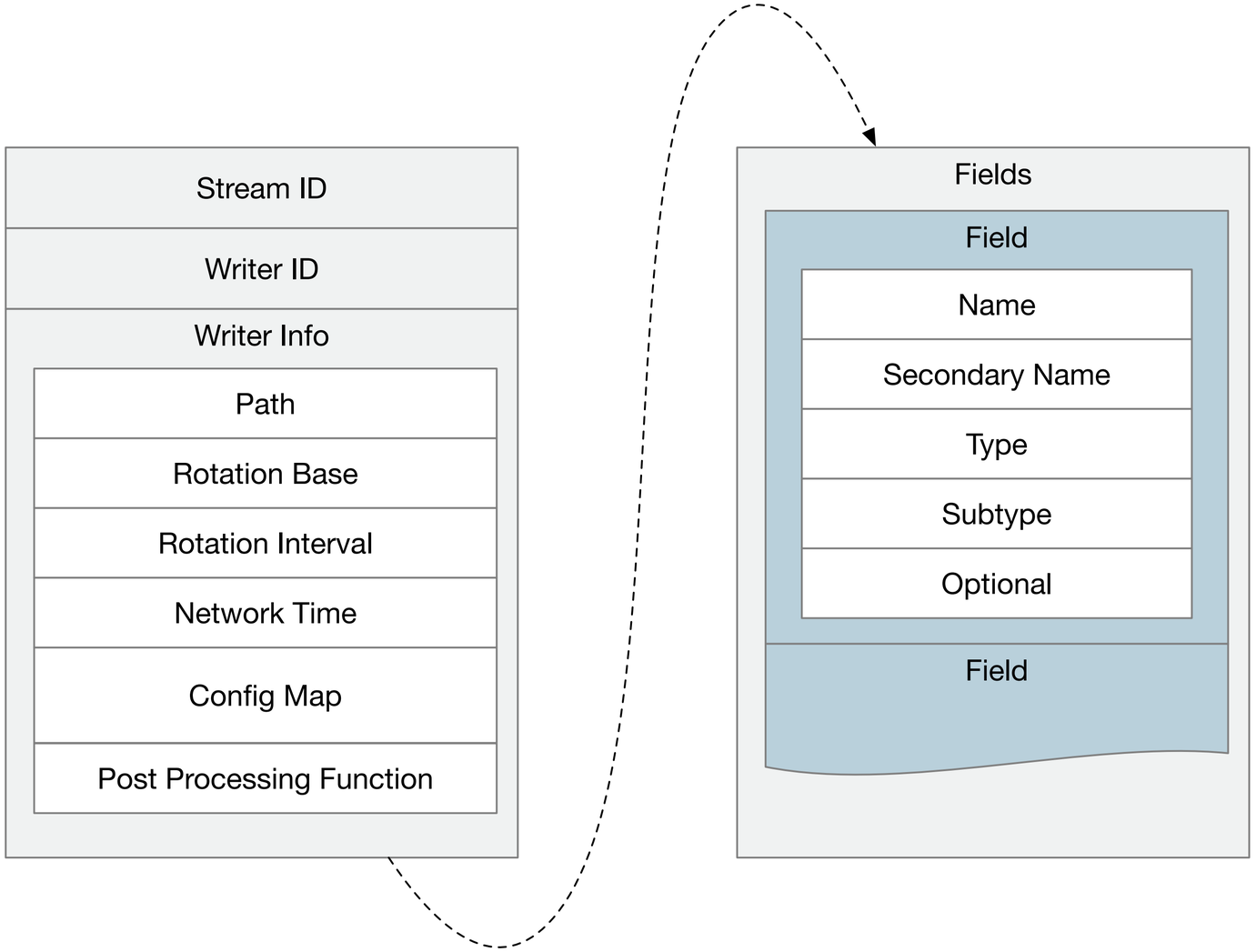
Let's try!



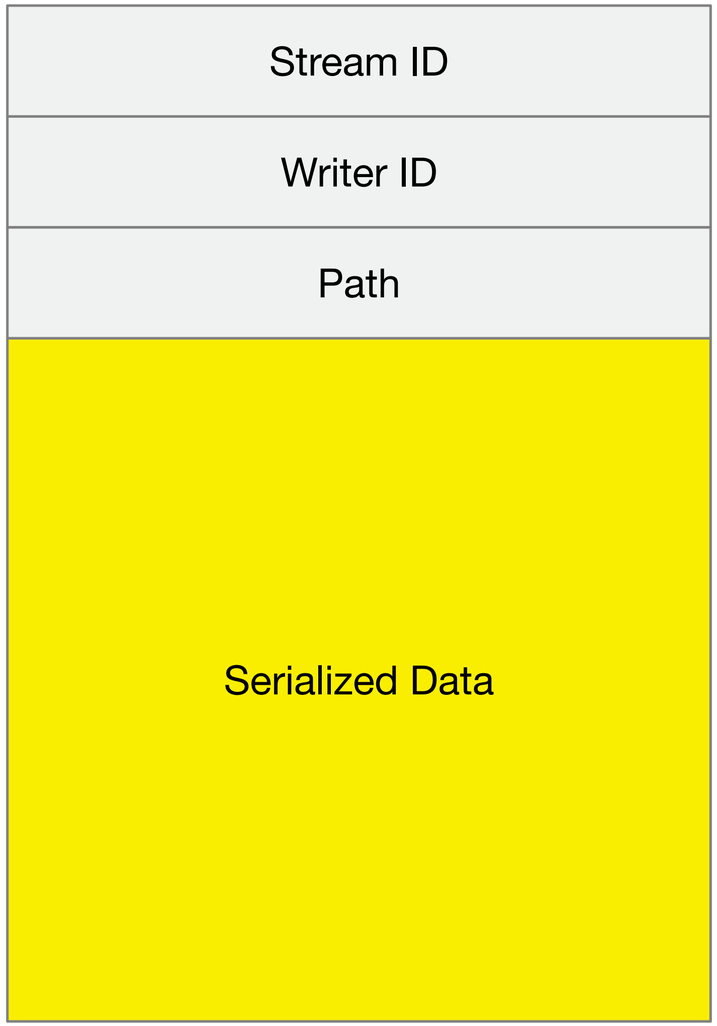
Broker Communication



LogCreate Message



LogWrite Message



Binary Data!

What does Zeek do?

Zeek-internal serialization format

```
zeek::detail::BinarySerializationFormat fmt;
fmt.StartRead(serial_data->data(), serial_data->size());

int num_fields;
bool success = fmt.Read(&num_fields, "num_fields");

if ( ! success )
  {
  reporter->Warning("failed to unserialize remote log num fields for stream: %s",
                    stream_id_name.data());
  return false;
  }

auto vals = new threading::Value*[num_fields];

for ( int i = 0; i < num_fields; ++i )
  {
  vals[i] = new threading::Value;

  if ( ! vals[i]->Read(&fmt) )
    {
    for ( int j = 0; j <= i; ++j )
      delete vals[j];

    delete[] vals;
    reporter->Warning("failed to unserialize remote log field %d for stream: %s", i,
                      stream_id_name.data());

    return false;
    }
  }
```

Deserialize value by value as "threading vals"

Decoding in VAST

```
caf::expected<std::vector<data>> process(const ::broker::zeek::LogWrite&
  auto serial_data = caf::get_if<std::string>(&msg.serial_data());
  if (!serial_data)
    return caf::make_error(ec::parse_error, "serial_data not a string");
  auto bytes = as_bytes(*serial_data);
  // Read the number of fields.
  uint32_t num_fields{};
  if (auto err = zeek::extract(num_fields, bytes))
    return err;
  // Read as many "threading values" as there are fields.
  std::vector<data> result;
  result.reserve(num_fields);
  for (size_t i = 0; i < num_fields; ++i) {
    data x;
    if (auto err = zeek::extract_value(x, bytes))
      return err;
    else
      result.push_back(std::move(x));
  }
  if (bytes.size() > 0)
    VAST_WARN("incomplete read, {} bytes remaining", bytes.size());
  return result;
}
```

Rebuilt deserialization logic

```
};

/// Parses a value out of binary Zeek data.
/// @param bytes The raw bytes to parse.
/// @returns An error on failure.
/// @post *bytes* is advanced by the number of bytes of the extract value.
template <class T>
caf::error extract(T& x, std::span<const std::byte>& bytes) {
  if constexpr (std::is_same_v<T, char>) {
    if (bytes.empty())
      return caf::make_error(ec::parse_error, "input exhausted");
    x = static_cast<char>(bytes[0]);
    bytes = bytes.subspan(1);
  } else if constexpr (std::is_same_v<T, bool>) {
    char c;
    if (auto err = extract(c, bytes))
      return err;
    x = (c == '\1');
  } else if constexpr (std::is_same_v<T, int>) {
    // In Zeek, an int has always 32 bits on the wire.
    uint32_t result{};
    if (auto err = extract(result, bytes))
      return err;
    x = static_cast<int>(result);
```

Expected type as in LogCreate

Static recursive parsing dispatch

Does it work?


```
vast start
```

```
vast import broker
```

```
redef Log::enable_local_logging = F;

event zeek_init()
  {
  Broker::listen(Broker::default_listen_address,
                 Broker::default_port,
                 Broker::default_listen_retry);
  }
```

TENZIR zeeks you!

Join our community: http://slack.tenzir.com

VAST

https://github.com/tenzir/vast

- Fast ingest of network telemetry
- Fast search via multi-level indexing
- Native Zeek support

THREAT BUS

https://github.com/tenzir/threatbus

- Publish-subscribe STIX bus
- Backbone agnostic (Kafka, Rabbit)
- Apps, e.g., sync Zeek with TIP

@tenzir_company

PS: We're hiring! 🙌