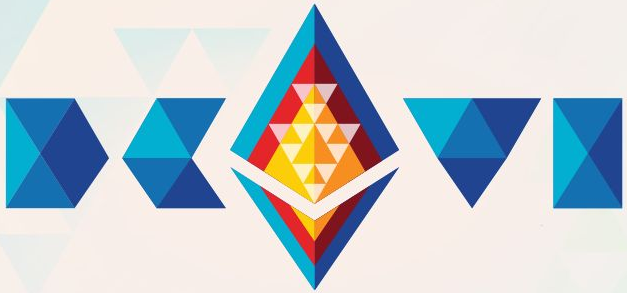


Introducing Substreams

The Graph

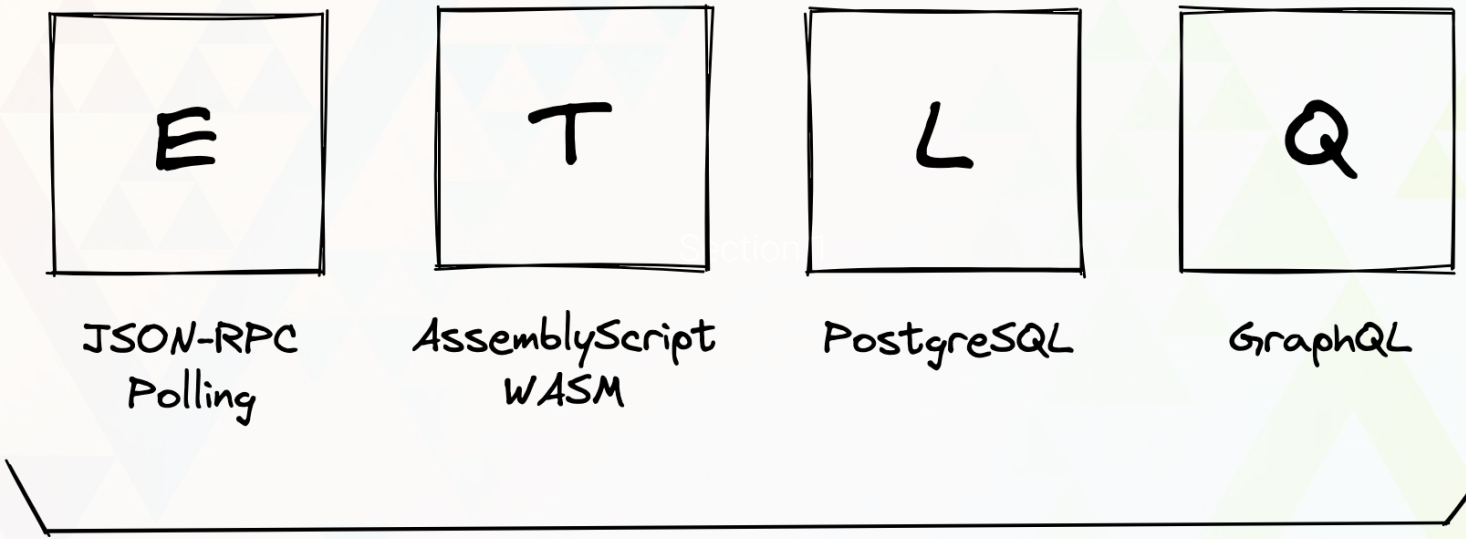
Alexandre Bourget

CTO, StreamingFast

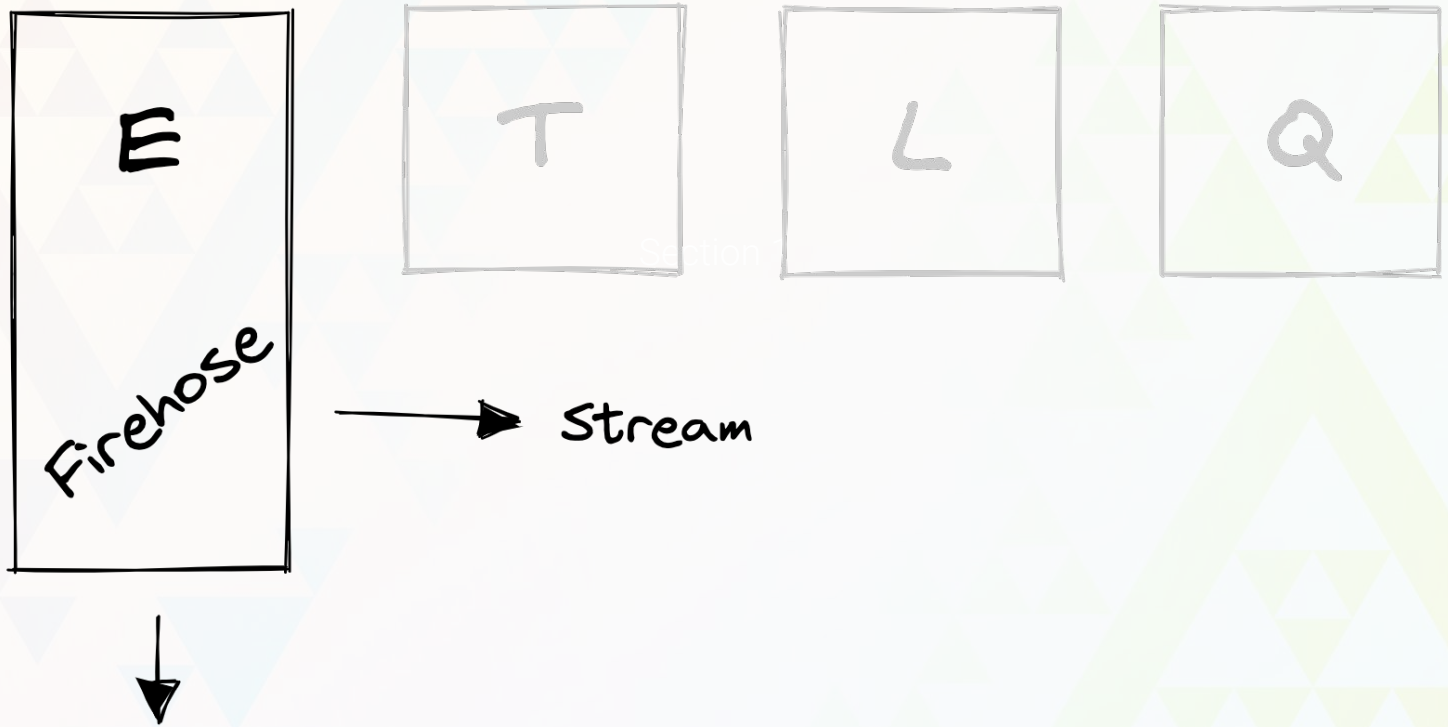
STREAMING FAST

joins

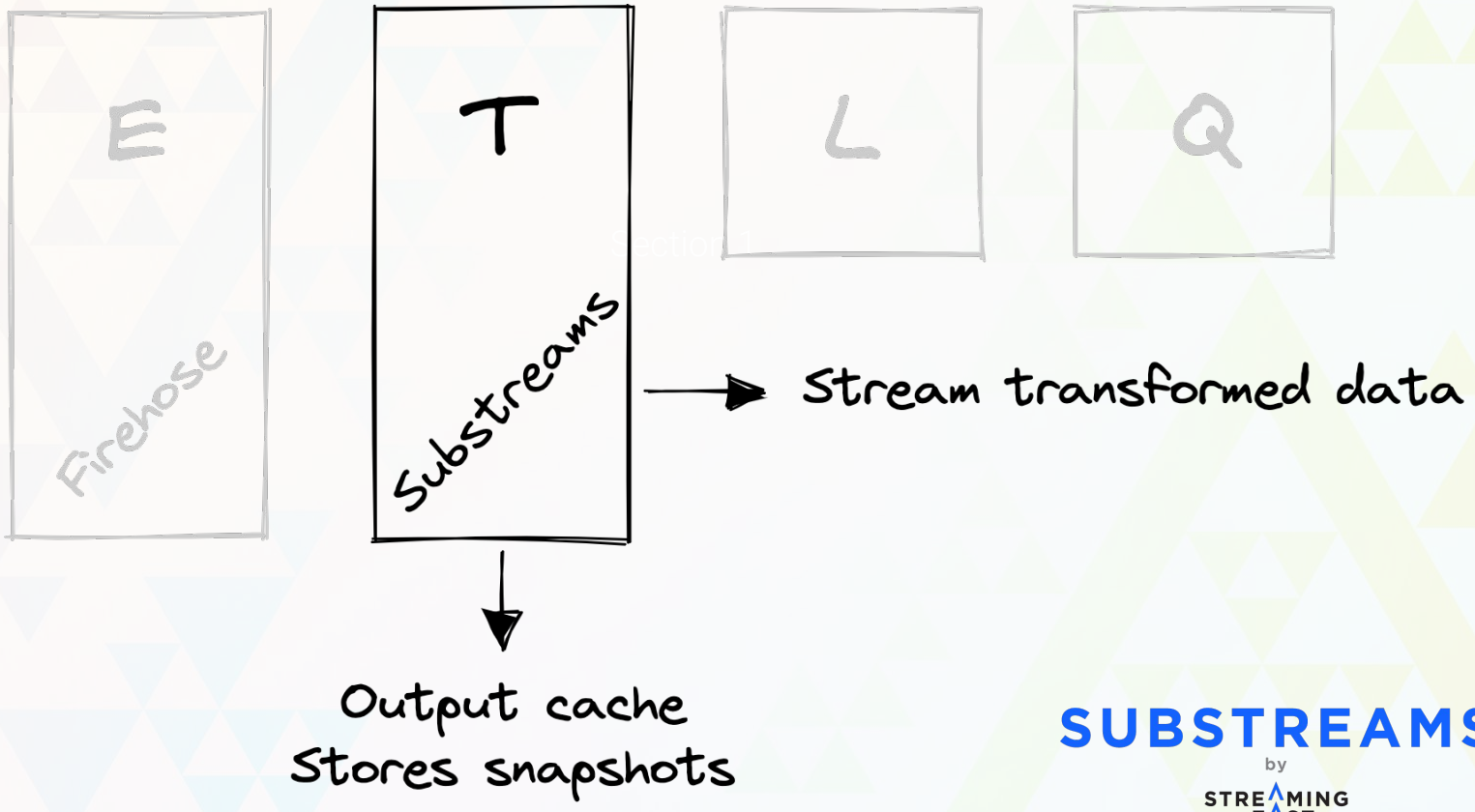
the graph



github.com/graphprotocol/graph-node



FIREHOSE
by
STREAMING FAST



SUBSTREAMS
by
STREAMING FAST

Problems

1. Latency
2. Stateful Processes
3. Costs
4. Coupled data
5. Reliability
6. Performance

Problems

1. Latency
2. Stateful Processes
3. Costs
4. Coupled data
5. Reliability
6. Performance

Solutions

1. Streaming First
2. Flat files
3. Flat files ++
4. Rich protobuf models
5. Stream cursors
6. Parallelization

FIREHOSE
by
STREAMING FAST

SUBSTREAMS

by

STREAMING FAST

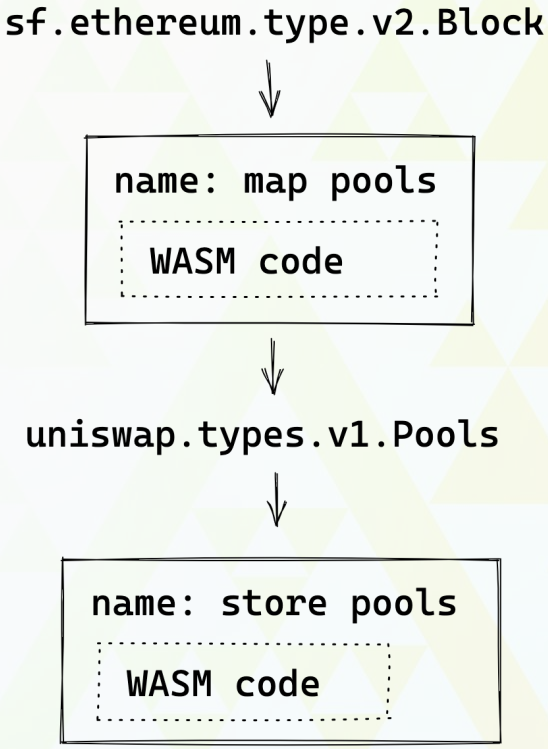
Substreams is

- A gRPC service
- Rust Modules
- Deterministic
- Parallelizable engine
- Infused with Firehose guarantees

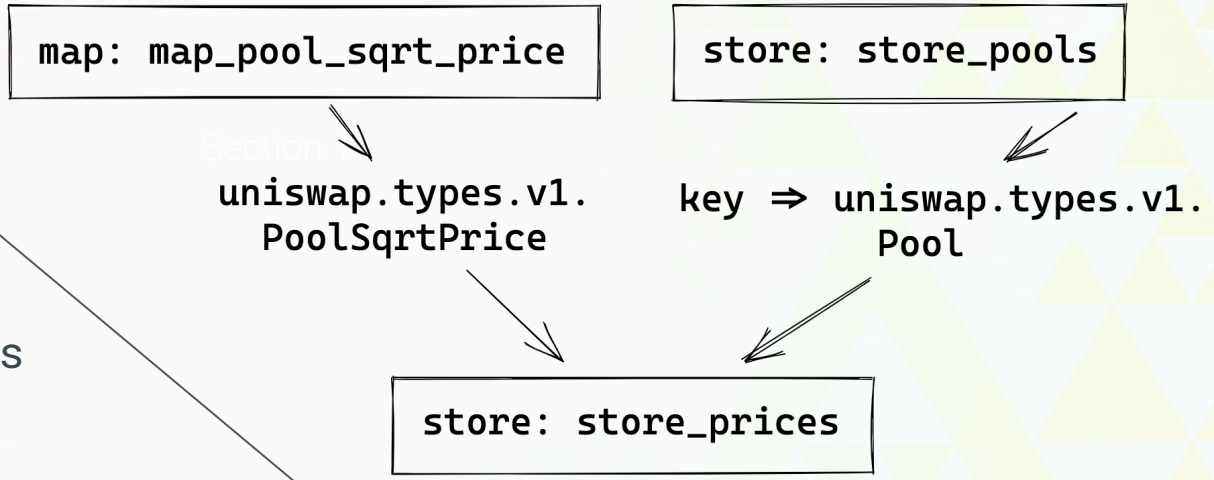
substreams.yaml

```yaml
specVersion: v0.1.0
package:
  name: uniswap_v3
  version: v0.1.0-beta
  url: https://github.com/streamingfast/substreams-uniswap-v3
  doc: |
    These Substreams modules make up all of Uniswap v3 entities
...
protobuf:
  files:
    - uniswap/uniswap.proto
...
binaries: {default: {file: ...wasm}}...
imports: ...
```

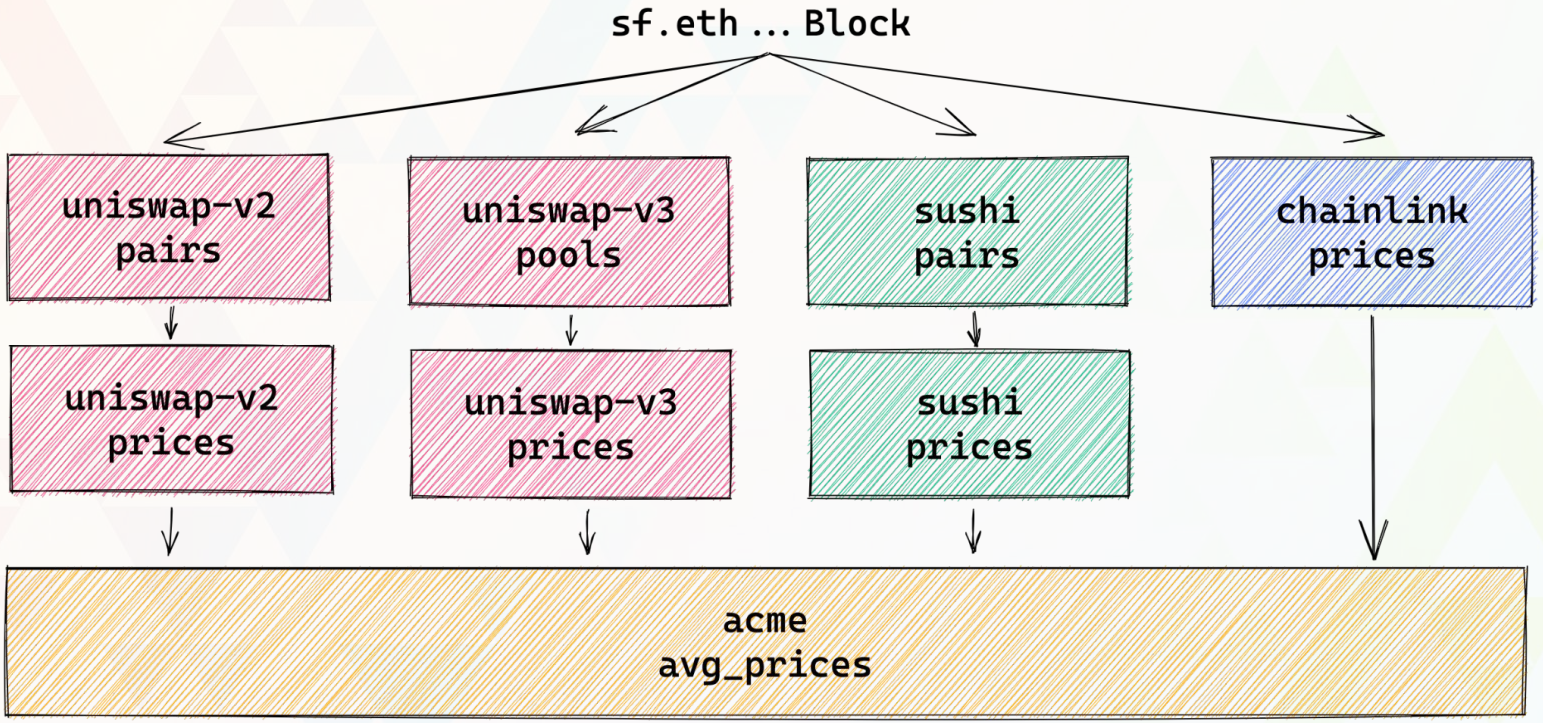
```yaml
modules:
  - name: map_pools
    kind: map
    initialBlock: 12369621
    inputs:
      - source: sf.ethereum.type.v2.Block
    output:
      type: proto:uniswap.types.v1.Pools

  - name: store_pools
    kind: store
    updatePolicy: set
    valueType: proto:uniswap.types.v1.Pool
    inputs:
      - map: map_pools

...
```



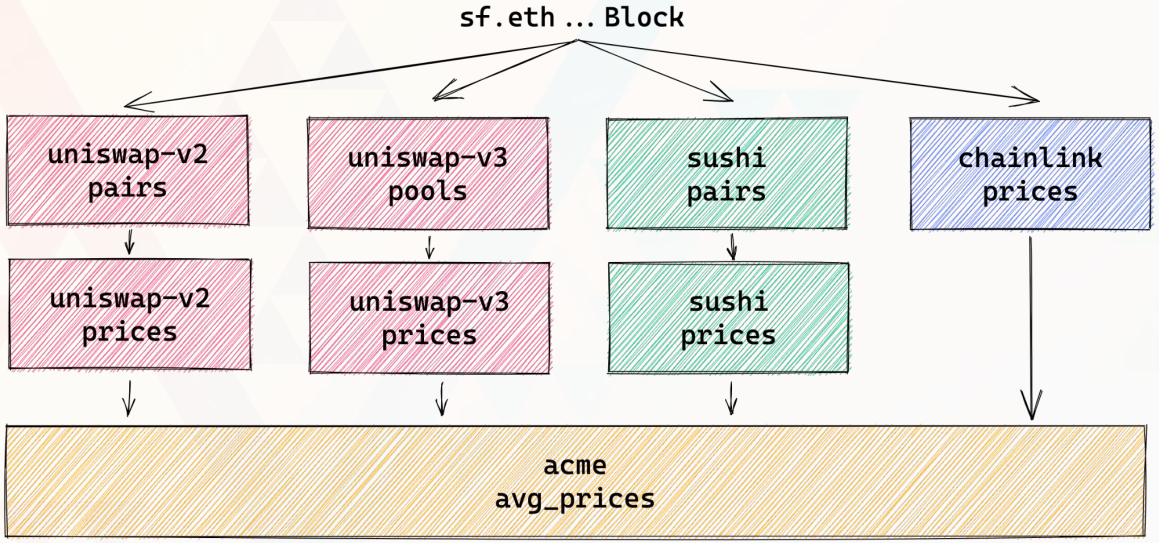
substreams.yaml



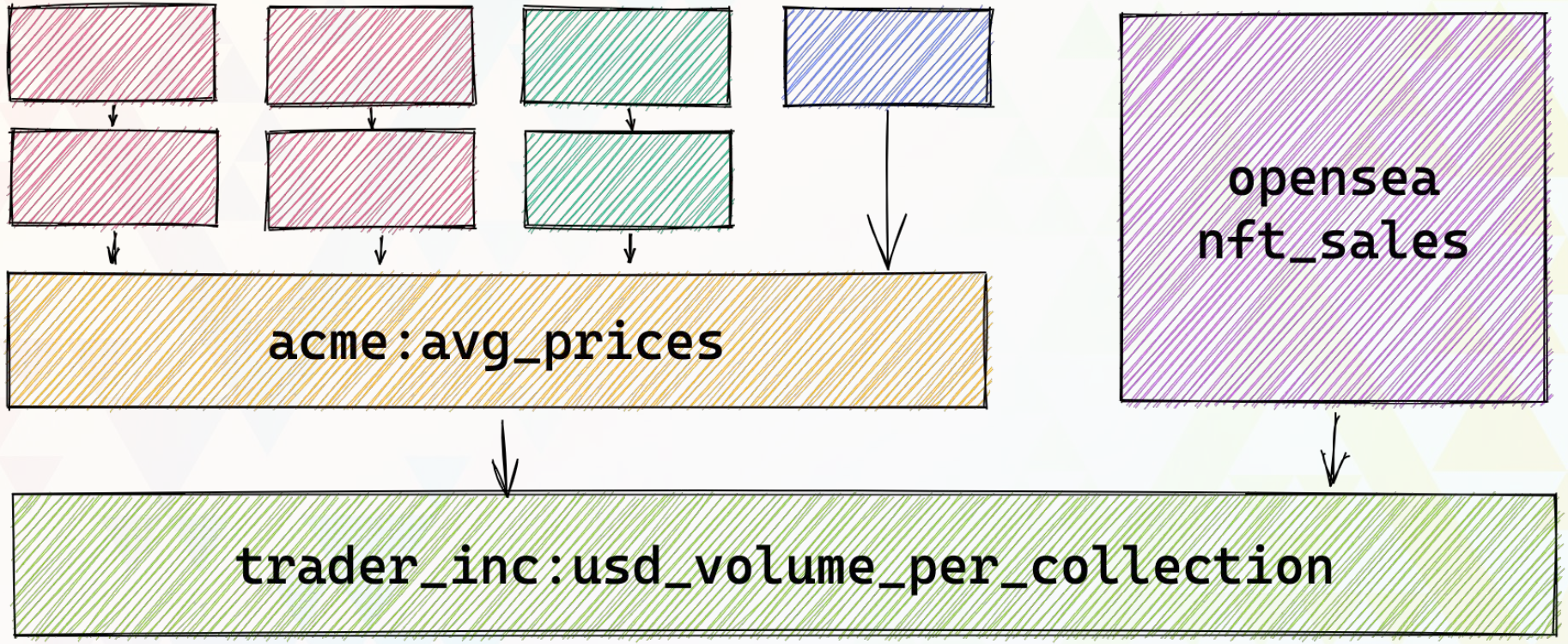

```yaml
modules:
...
- name: store_prices
  kind: store
  updatePolicy: set
  initialBlock: 12369621
  valueType: bigfloat
  inputs:
    - map: map_pool_sqrt_price
    - store: store_pools
```



* Different **color** == different **author**



* Different **color** == different **author**



* Different **color** == different **author**

Sinks

- PostgreSQL, MongoDB, or any other database
- Kafka, Fluvio, RabbitMQ
- S3 Buckets, Clickhouse, Redshift, BigQuery
- Real-time bots, trading ops, Slack notif.
- Ad-hoc analysis, small one-offs, Jupyter Notebooks
- Any program written in any supported language
- Subgraphs through `graph-node` (Soon™)

Consume **Packages** with python

```
# pip3 install grpcio-tools protobuf==3.20.1
def main():
    with open("whale-alert-v1.0.0.spkg", 'rb') as f:
        pkg = Package()
        pkg.ParseFromString(f.read())

    stream = substreams_service().Blocks(Request(
        modules=pkg.modules,
        fork_steps=[STEP_IRREVERSIBLE],
        start_block_num=15_000_000,
        output_modules="map_whale_alert",
    ))

    for response in stream:
        send_to_slack(response)
```

https://github.com/streamingfast/substreams-playground/tree/master/consumers/python

uniswap.proto

```
package uniswap.types.v1;

message Pools {
  repeated Pool pools = 1;
}
message Pool {
  string address = 1;
  uint64 created_at_block = 4;
  ERC20Token token0 = 5;
  ERC20Token token1 = 6;
  uint32 fee_tier = 7;
  string transaction_id = 32;
}
```

```
message ERC20Token {
  string address = 1;
  string name = 2;
  string symbol = 3;
  uint64 decimals = 4;
}
```

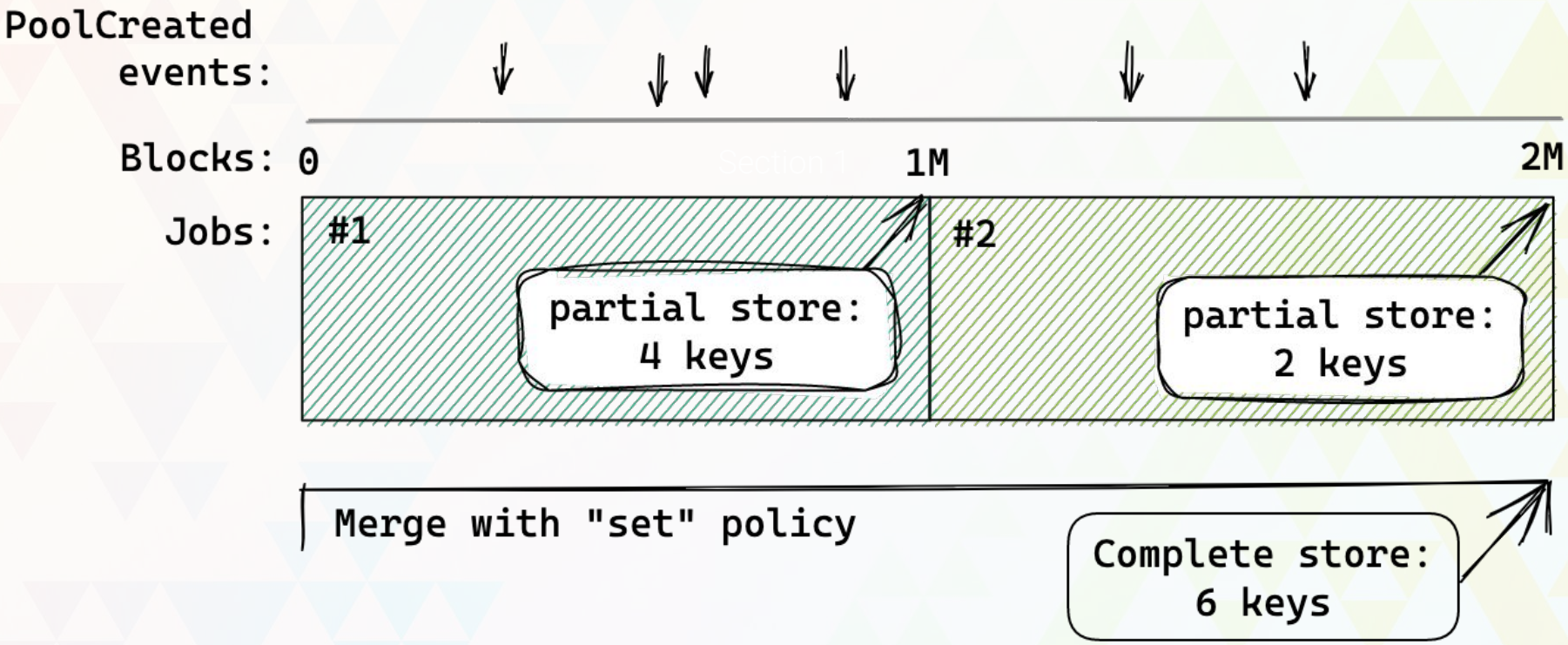
A **map** module

```rust
#[substreams::handlers::map]
pub fn map_pools(block: Block) -> Result<Pools, Error> {
    Ok(Pools {
        pools: block
            .events::<PoolCreated>(&[&UNISWAP_V3_FACTORY])
            .filter_map(|(event, log)| {
                Some(Pool {
                    address: Hex(&log.data()[44..64]).to_string(),
                    fee_tier: event.fee.as_u32(),
                    token0: rpc::create_uniswap_token(&event.token0),
                    token1: rpc::create_uniswap_token(&event.token1),
                    created_at_block: block.number,
                    transaction_id: trx_id_from_log(&log),
                    ...
                })
            })
            .collect(),
    })
}
```

A **store** module

```rust
#[substreams::handlers::store]
pub fn store_pools(pools: Pools, output: ProtoStoreSet<Pool>) {
    for pool in pools.pools {
        log::info!("pool addr: {}", pool.address);
        output.set(pool.log_ordinal,  format!("pool:{}", pool.address),
&pool);
    }
}
```

Parallel execution of a **store**



Section 1

Show me

Fin

firehose.streamingfast.io
substreams.streamingfast.io

Alexandre Bourget
CTO, StreamingFast
alex@streamingfast.io

@bourgetalexndre