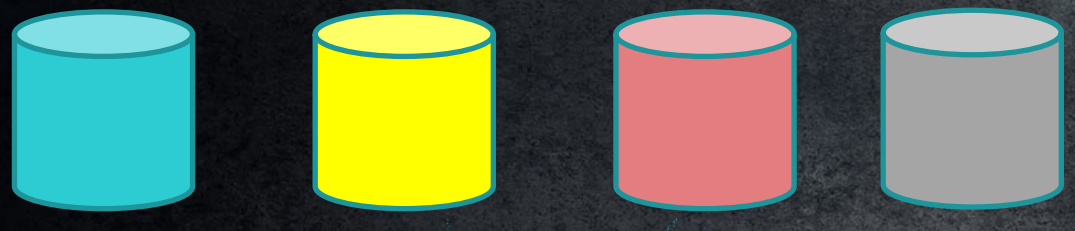

EFPSignal
agile content

EFP-SIGNAL DECLARES CONTENT

Different active data sources



EFPSignal

EFP

Our shipper who delivers the content

EFP-SIGNAL NOTIFIES RECEIVER

Our receiver

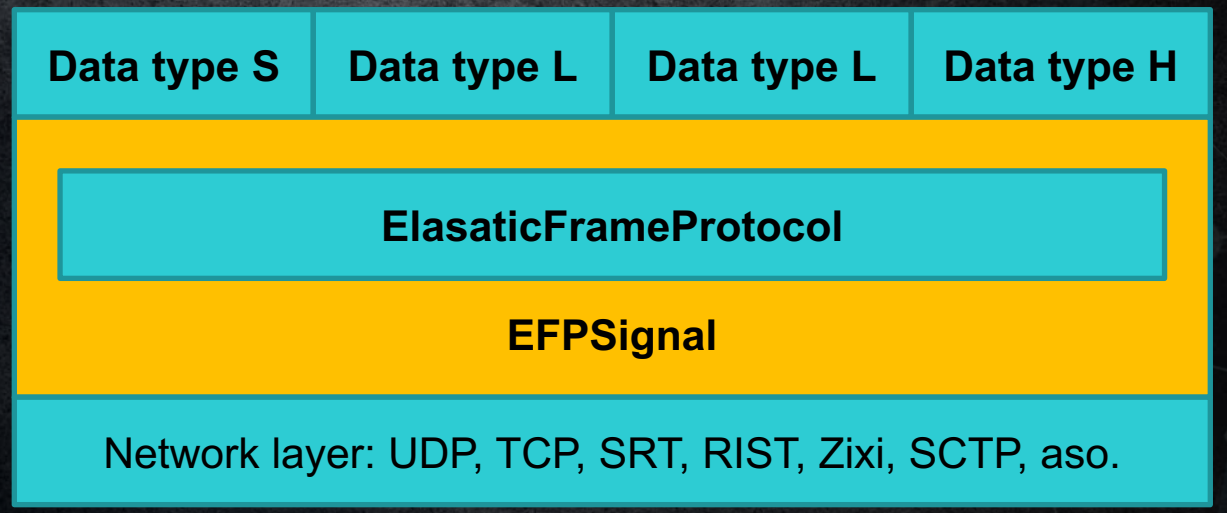
EFPSignal

gets information about available
data sources + detailed information

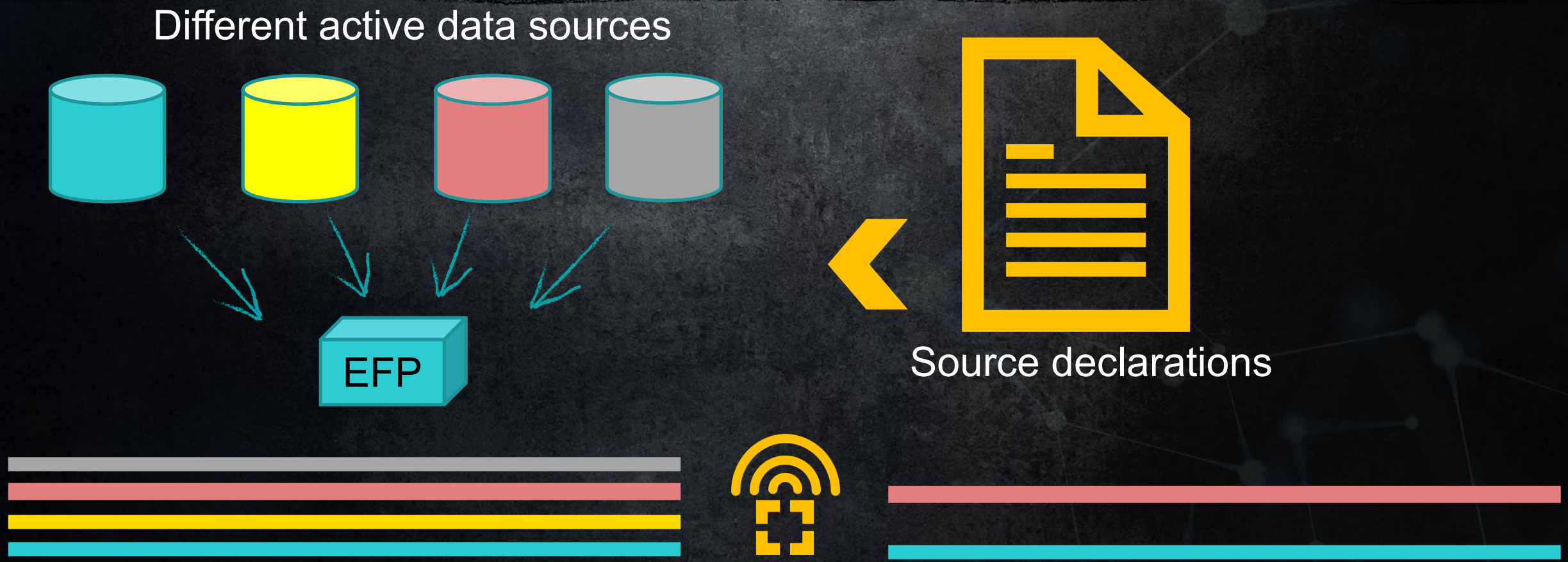
EFPSIGNAL WRAPS AROUND EFP

Is declaring content and can handle reading declarations

- From data -> Declare
- From network -> Notify



EFP-SIGNAL CAN ACT AS A FILTER

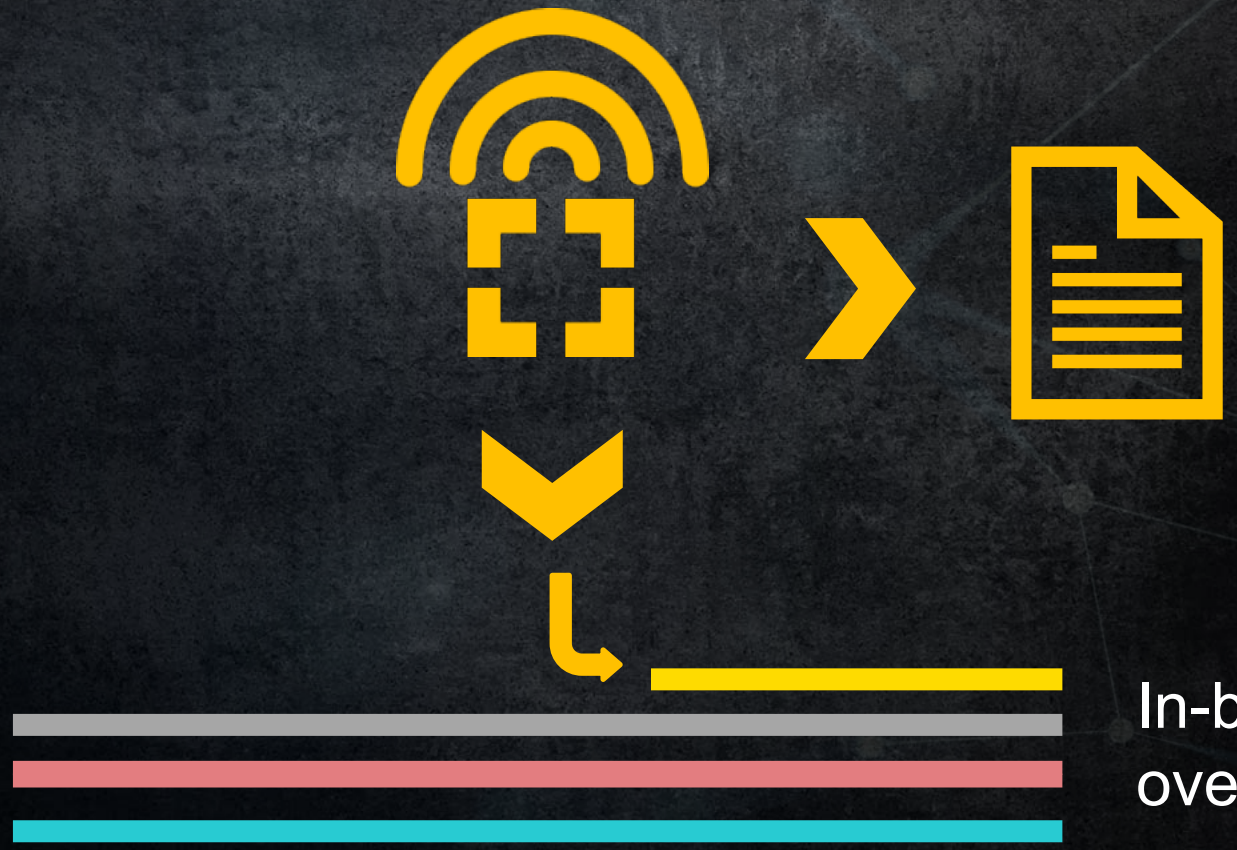


DATA FORMAT

Declaration payload is JSON or Binary

EFPSignal

EFP-SIGNAL DECLARE THE CONTENT IN-BAND OR OUT OF BAND



Send OOB
HTTP
Websocket
(anything)

In-band signaling
over reserved EFP ID 0

EFP-SIGNAL RECEIVE DECLARATIONS IN-BAND OR OUT OF BAND

Receive OOB
HTTP
Websocket
(anything)

In-band signaling
As reserved EFP ID 0

C++ API

EFP-SIGNAL (SEND)

Subclasses ElasticFrameProtocolSend.

Creation of the source

```
EFPSignalSend myEFPSignalSend(MTU, 5000);
```

Register callback

```
myEFPSignalSend.declareContentCallback = std::bind(&declareContent, std::placeholders::_1);
```

When EFP-Signal see undeclared content

```
void declareContent(EFPStreamContent* content) {
```

```
//Setings
bool mDropUnknown = false;
bool mAutoRegister = true;
bool mEmbeddInStream = true;
bool mEmbeddOnlyChanges = false;
bool mEmbeddBinary = false;
uint32_t mEmbedInterval100msSteps = 1;
```

Drop unknown data

Auto register unknown data (callback is triggered if unknown data appears)

Put current declaration in EFP-Stream id 0

Only send declaration if changes are seen

Embed binary else JSON format is selected

If embedded and mEmbedOnlyChanges == false. This is the interval declaration is embedded

EFP-SIGNAL (RECEIVE)

Subclasses ElasticFrameProtocolReceive.

Creation of the receiver (callbacks + unfinished frames timeout + head of line blocking flush timeout)

```
EFPSignalReceive myEFPSignalReceive(5,2);
 myEFPSignalReceive.receiveCallback = std::bind(&gotData, std::placeholders::_1);
 myEFPSignalReceive.contentInformationCallback = std::bind(&gotContentInformation, std::placeholders::_1);
```

In-band content declarations will trigger the callback -> contentInformationCallback
Access the declaration like this:

```
void gotContentInformation(std::unique_ptr<EFPSignalReceive::EFPSignalReceiveData>& data) {
  for (auto &rItem: data->contentList) {
  }
}
```

EFPSIGNAL COMMUNICATION

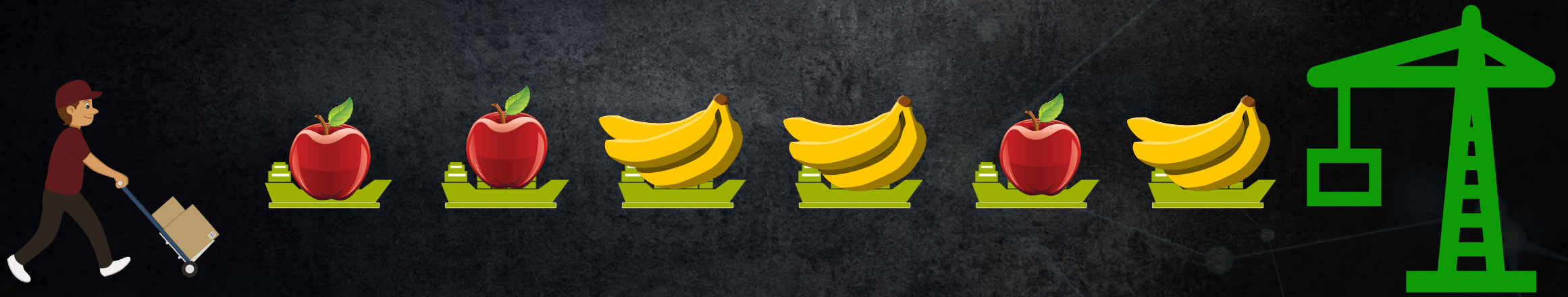
BASIC CONCEPT

I got:
Bananas
Oranges
Melons
Apples

OK then I want:
Bananas
Apples

BASIC CONCEPT

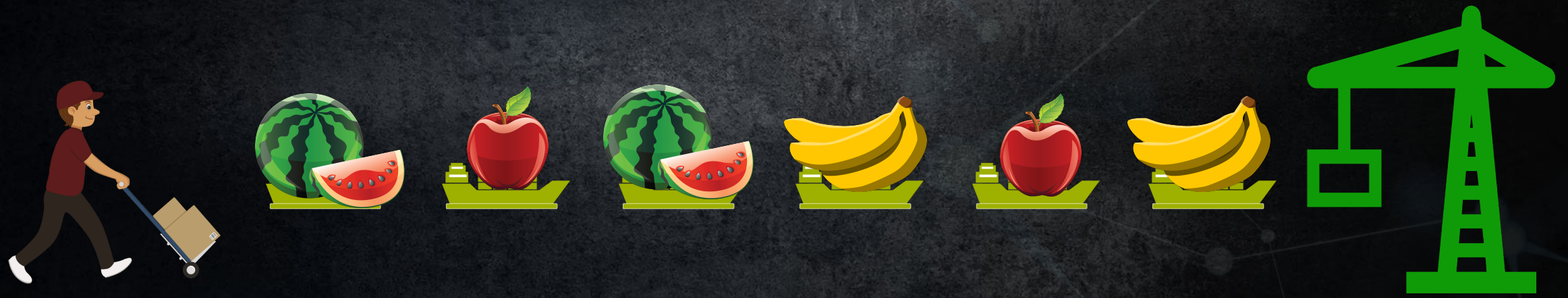
BASIC CONCEPT



Now I also
want melons.

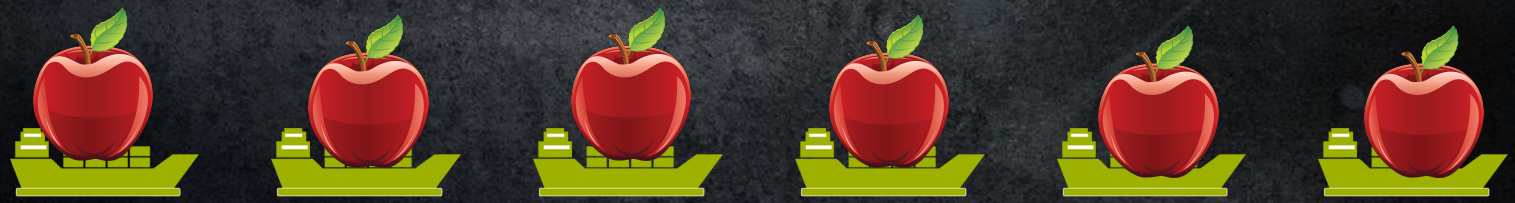
BASIC CONCEPT

BASIC CONCEPT



Stop sending
Bananas and melons

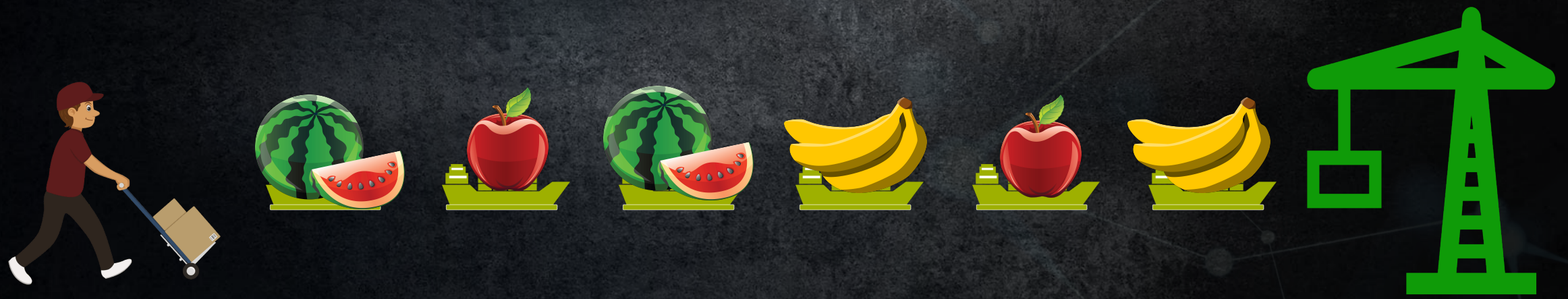
BASIC CONCEPT



TRANSPORT OF DECLARATIONS

BASIC CONCEPT

So when our supplier supplier is active sending us gods, he can send us declarations in different ways



BASIC CONCEPT

So when our supplier supplier is active sending us gods, he can send us declarations in different ways

Declare what he got out of band

I got:
Bananas
Oranges
Melons
Apples

BASIC CONCEPT

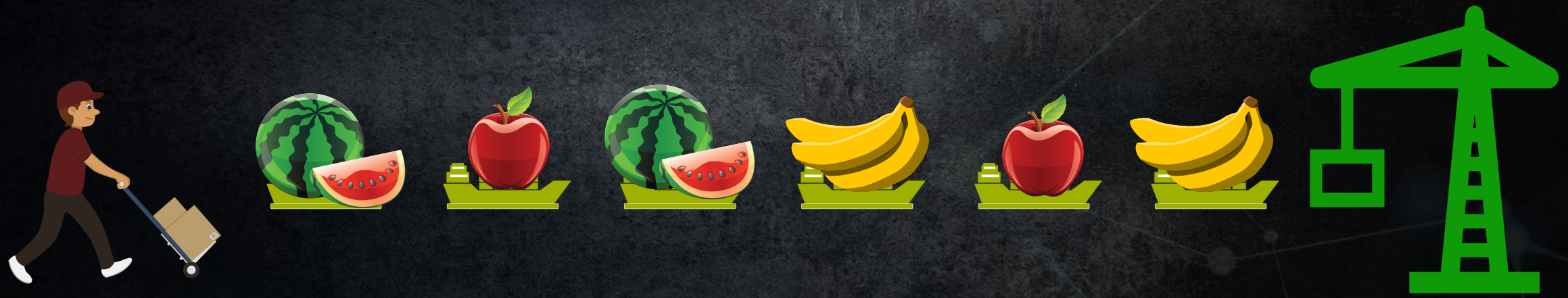
So when our supplier supplier is active sending us gods, he can send us declarations in different ways

Declare what he got in-band

I got:
Bananas
Oranges
Melons
Apples

BASIC CONCEPT

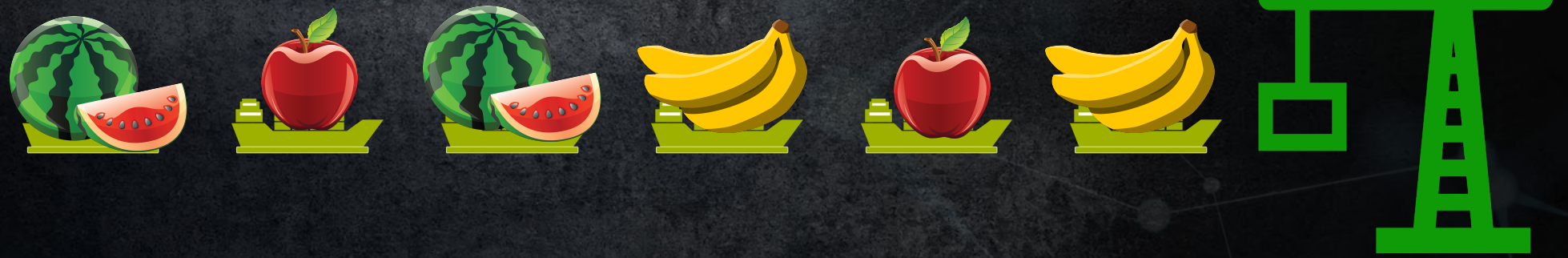
We can also configure our shipper to periodically send declarations



I got:
Bananas
Oranges
Melons
Apples

BASIC CONCEPT

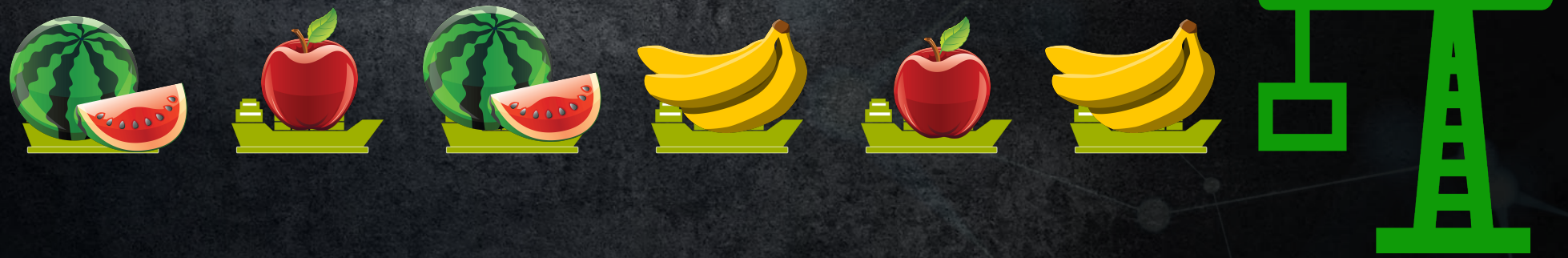
Or only when his stock changes



I’m out of Bananas

BASIC CONCEPT

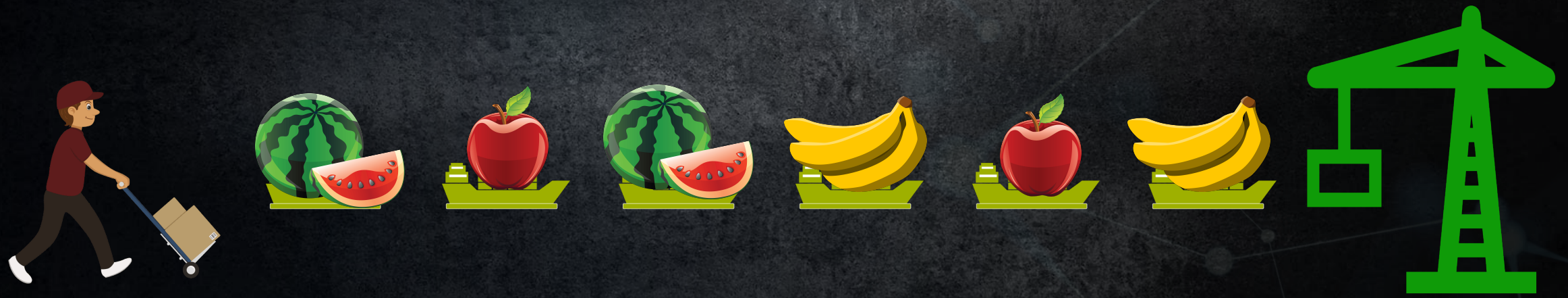
Or only when his stock changes



I now got Kiwis

BASIC CONCEPT

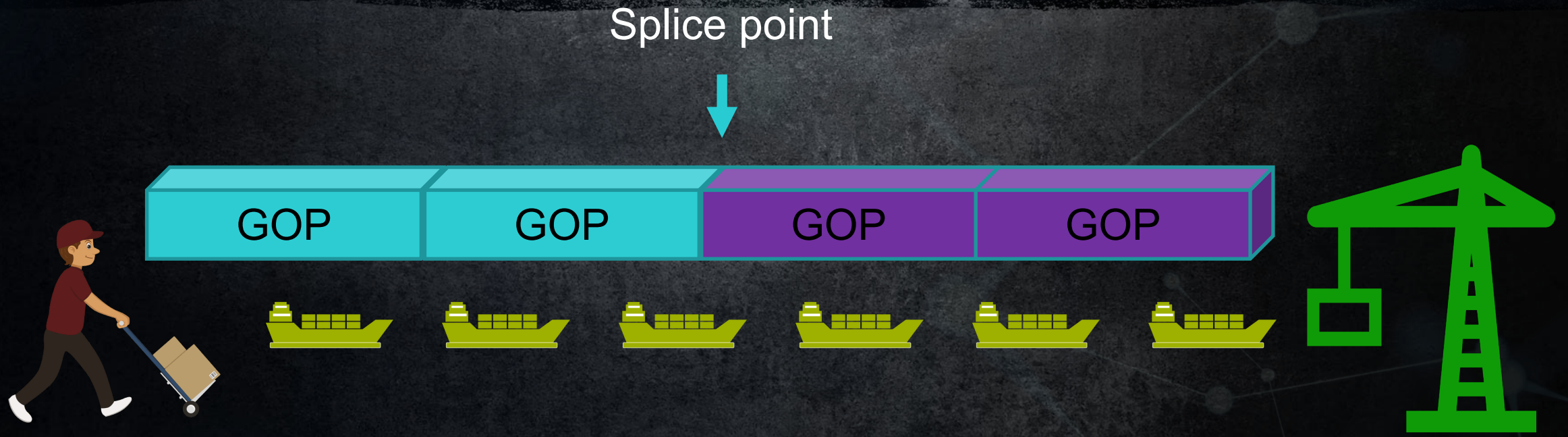
Or other properties changes



The price of apples changed

EFP-SIGNAL SIGNALING SPECIAL VIDEO MODE

SYNCHRONIZED VIDEO SWITCHING



Video source 1

mGSyncGroupID = 5

Video source 2

mGSyncGroupID = 5

For seamless splice points. Assets must be of same SyncGroup and ’I frame’ PTS jitter of max +-3ms. (This version assumes 90kHZ PTS)

DECLARATION OF MEDIA GROUP

Video source 1
Video source 3
Audio source 1
Audio source 2
Aux source 1

Group of EFP streams

Video source 2
Video source 4
Audio source 3
Audio source 2
Aux source 2

A group can re-use streams declared in other groups. In this example Audio Source 2 is used in both declared groups.

agile content

EFPSignal

THANKS FOR ATTENDING

ElasticFrame Protocol