Collecting and analyzing routing information with ROTONDA

RIPE91

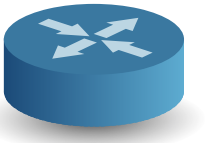
October 2025, Bucharest, Romania

Luuk Hendriks, Jasper den Hertog

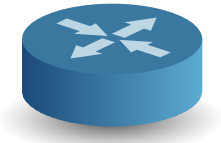
NLNETLABS

BMP?

BMP, oversimplified



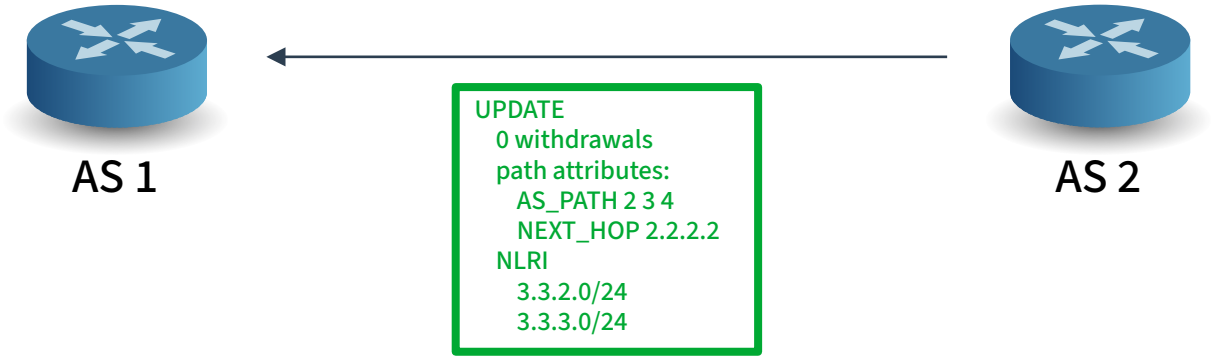
AS 1



AS 2

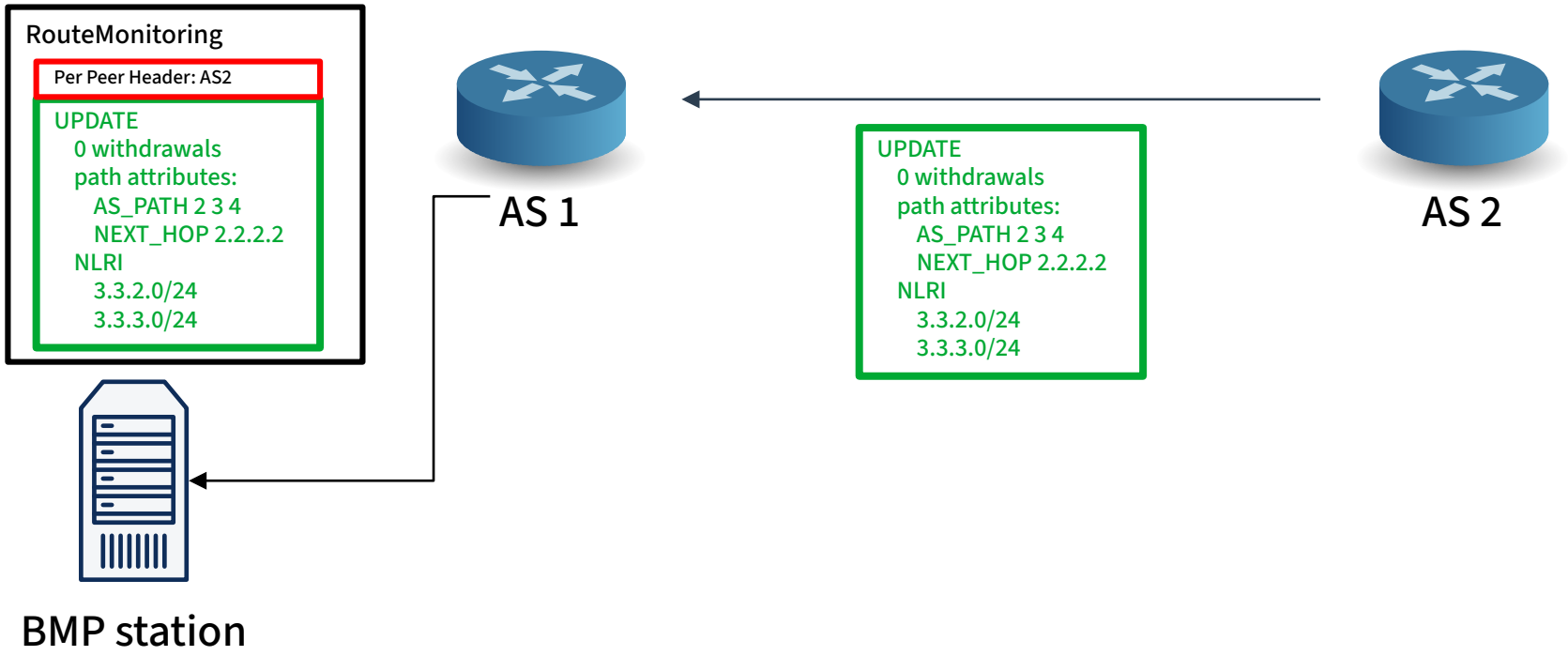
BMP station

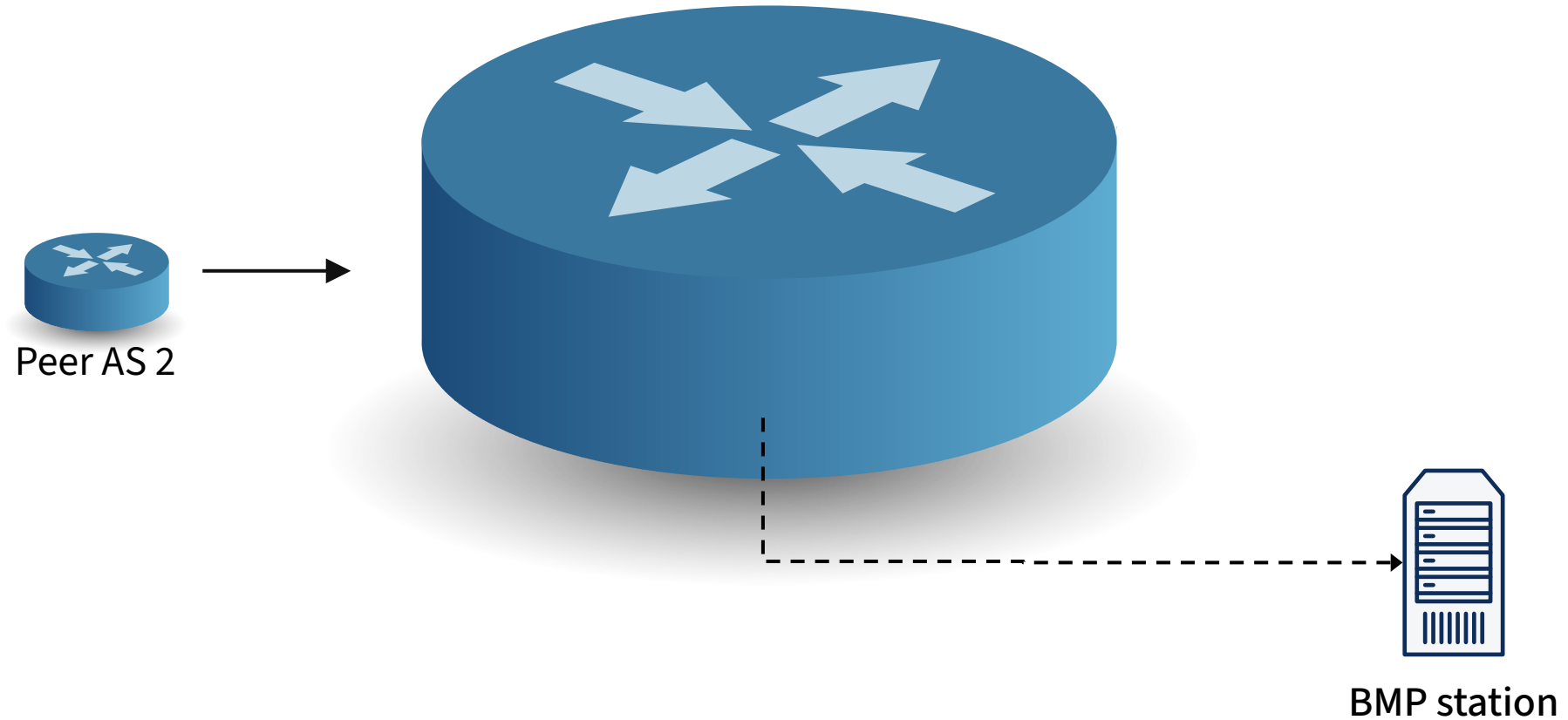
BMP, oversimplified

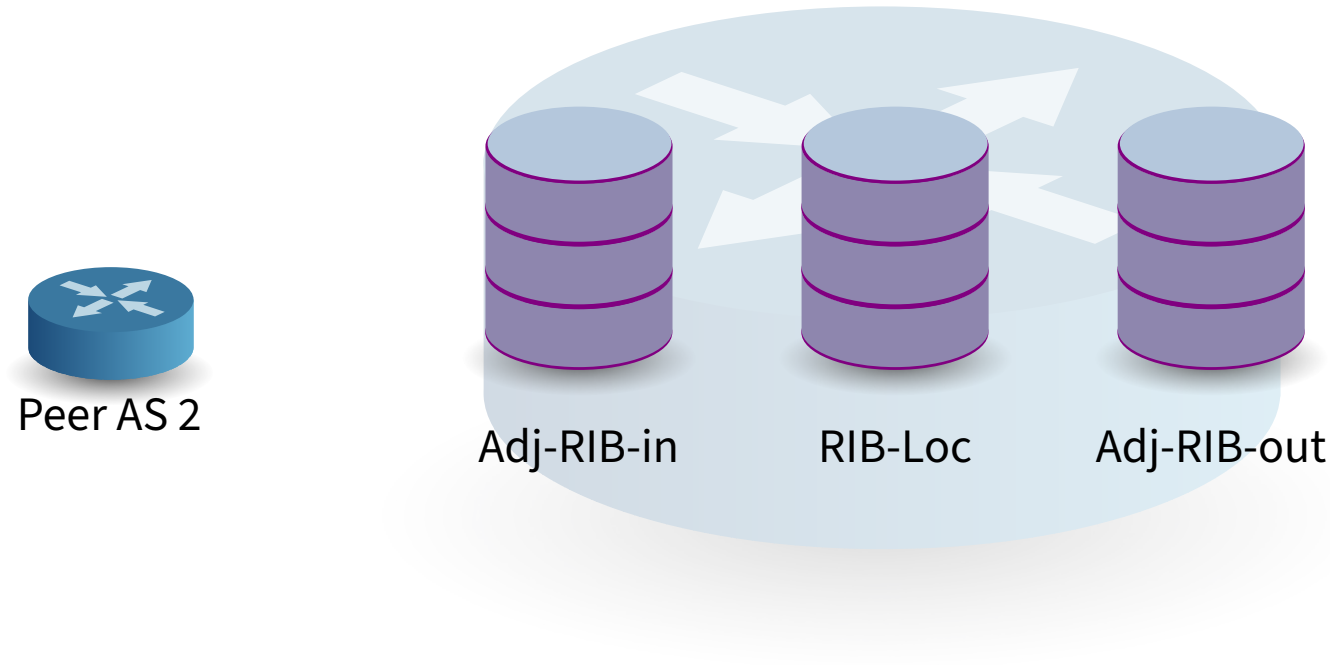


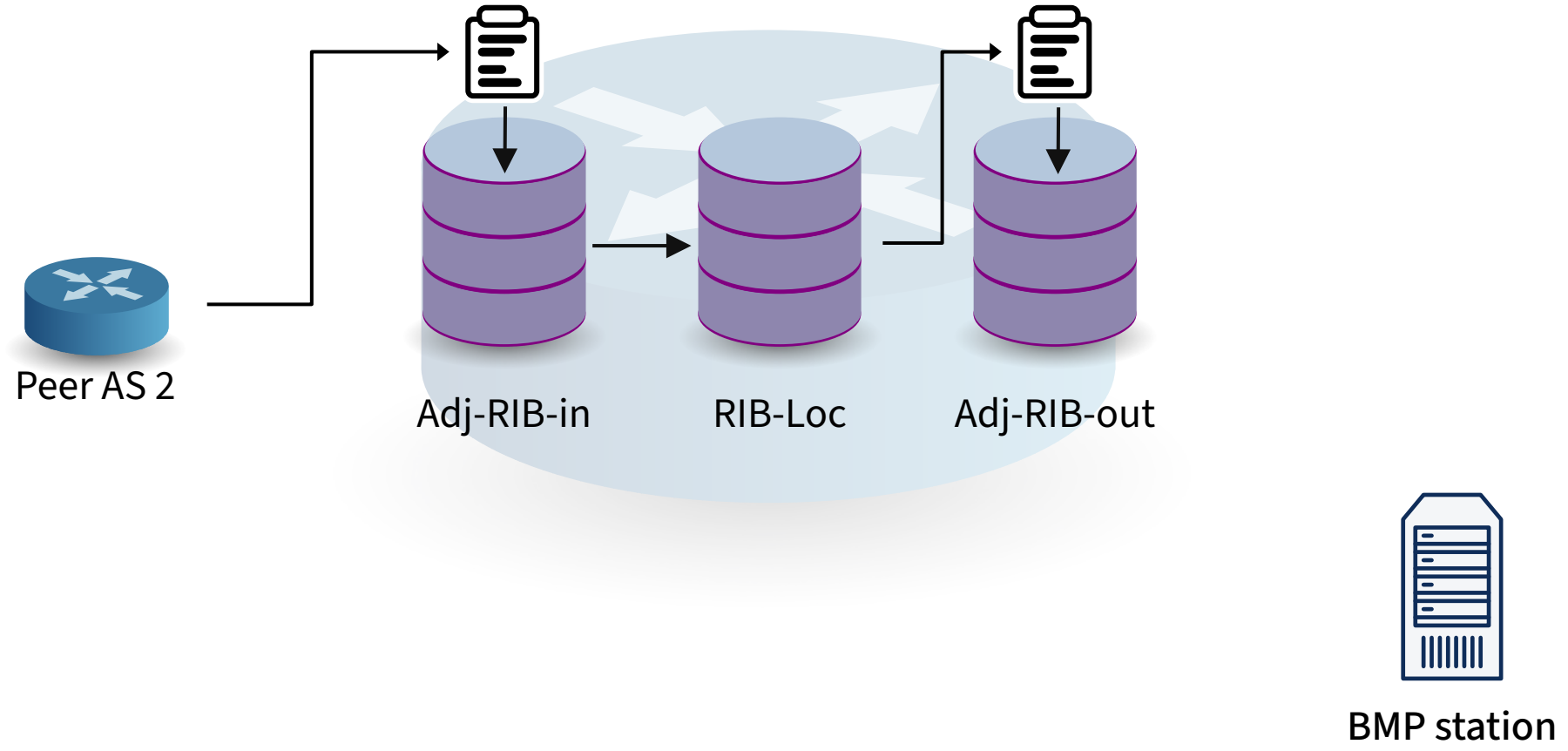
BMP station

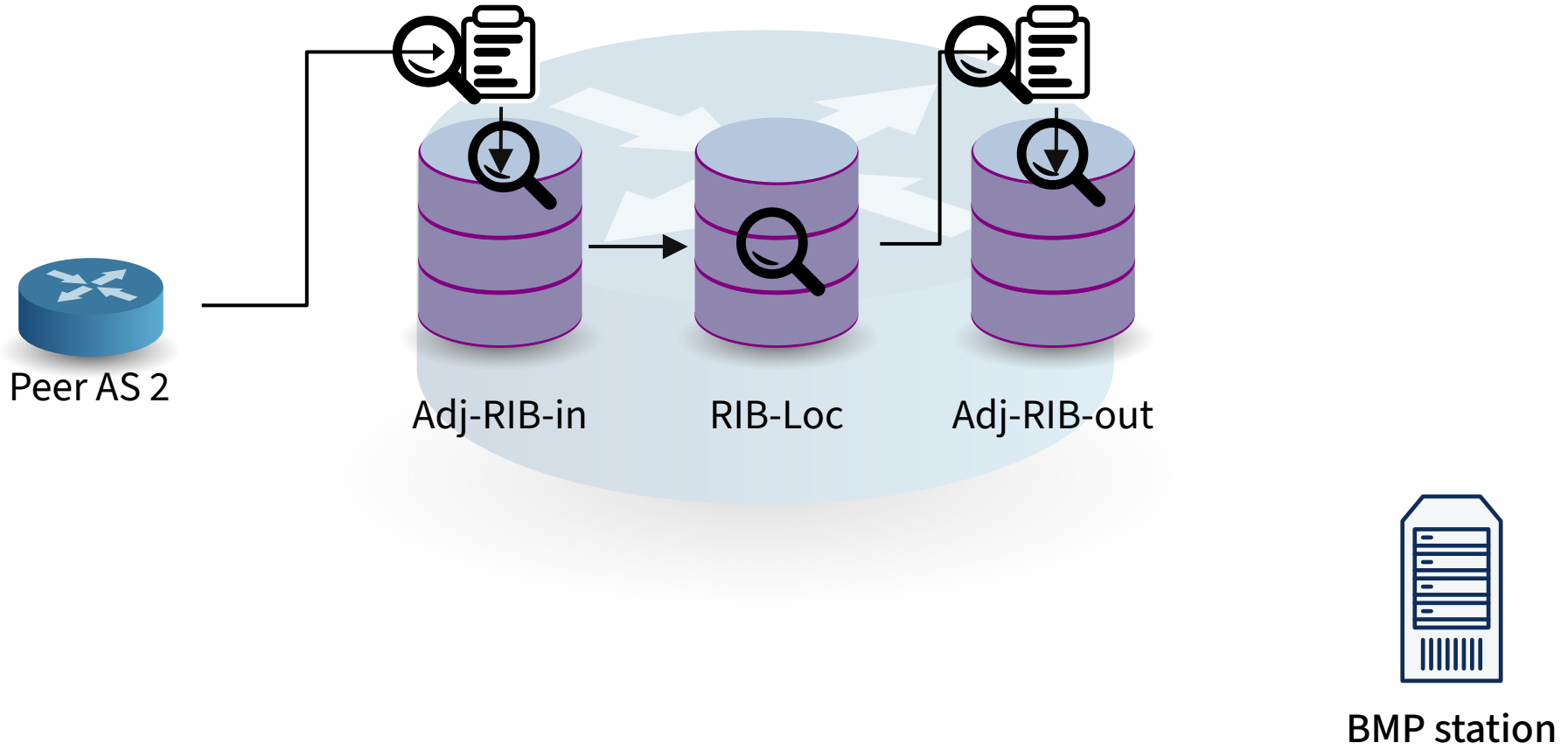
BMP, oversimplified

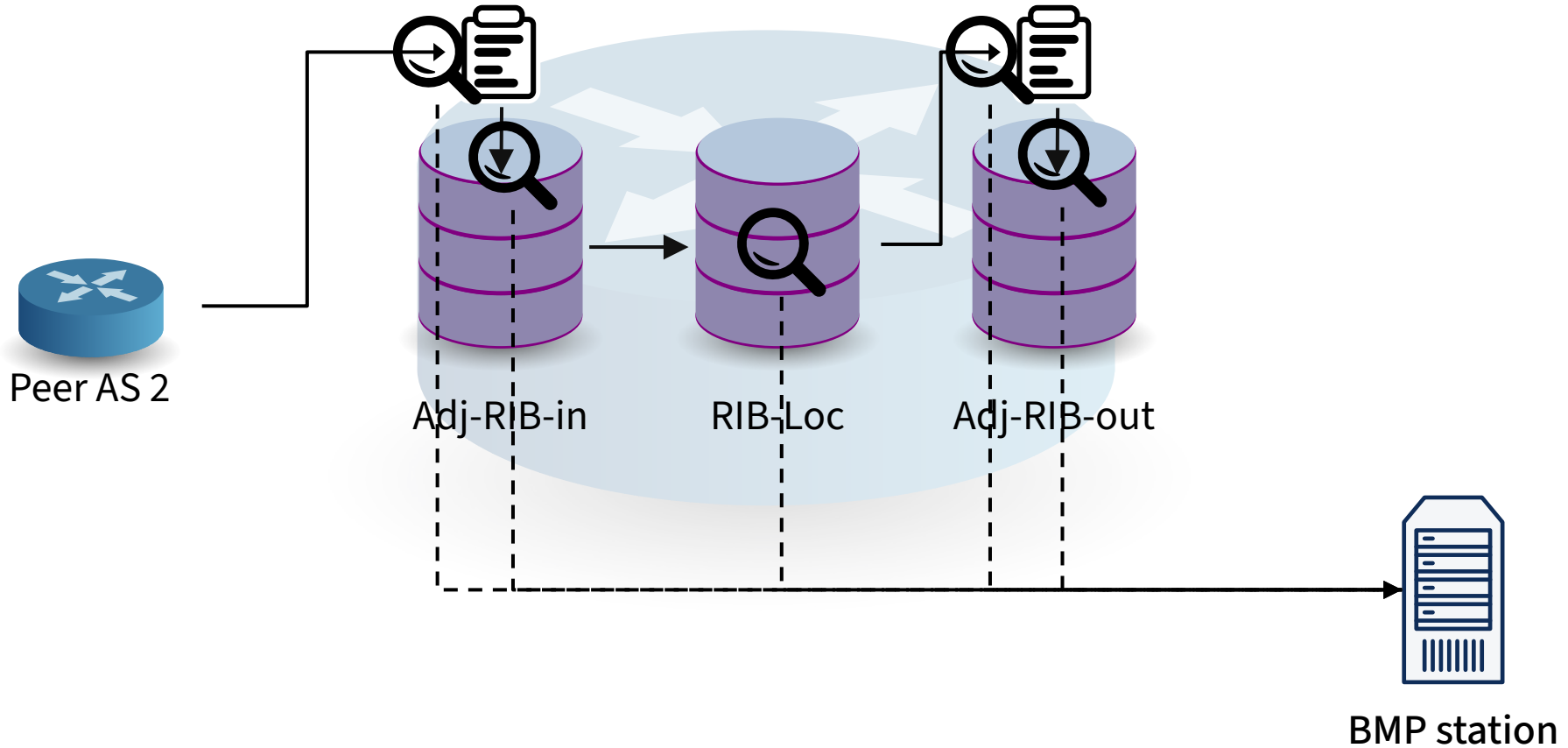


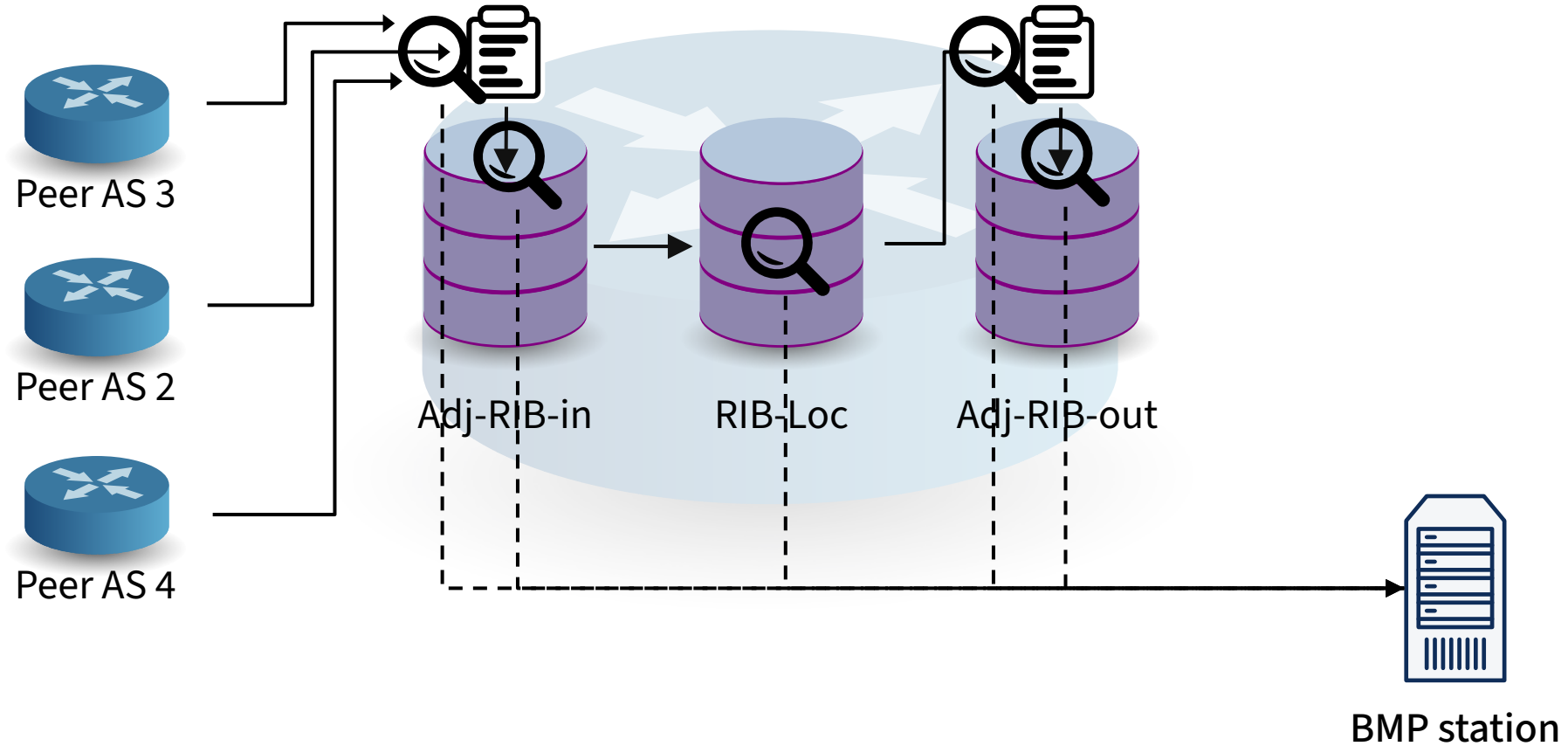

Peer AS 2
BMP station

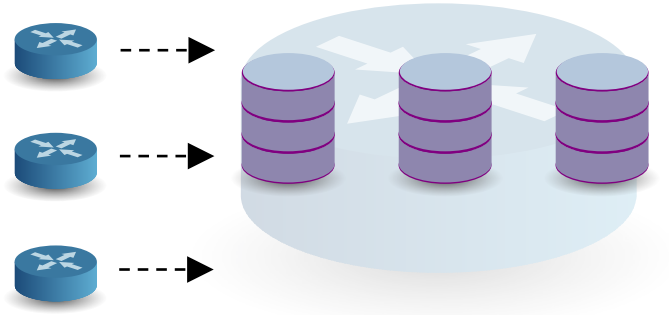

Peer AS 2
Adj-RIB-in
RIB-Loc
Adj-RIB-out
BMP station


Peer AS 2
Adj-RIB-in
RIB-Loc
Adj-RIB-out
BMP station

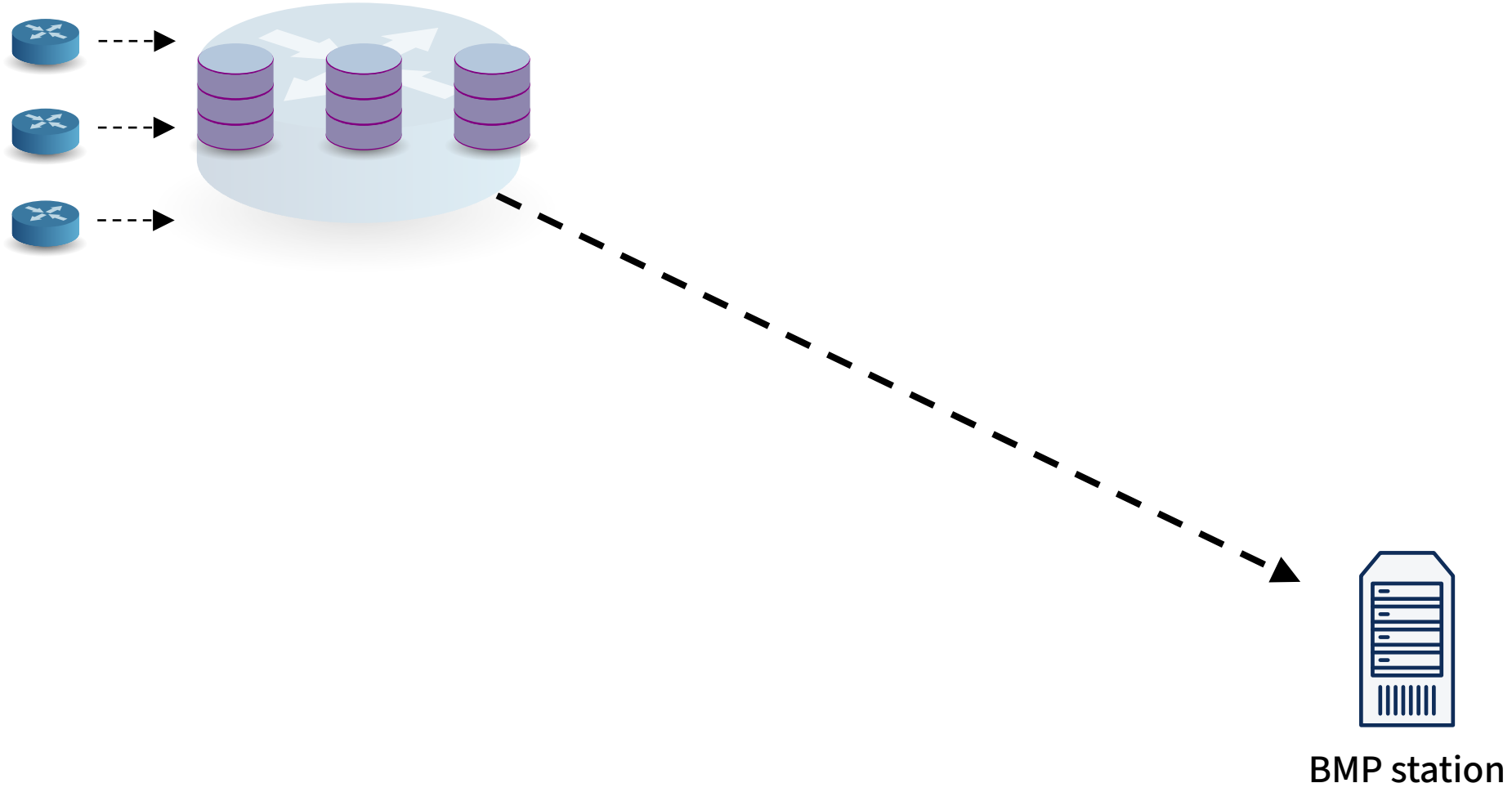

Peer AS 2
Adj-RIB-in
RIB-Loc
Adj-RIB-out
BMP station

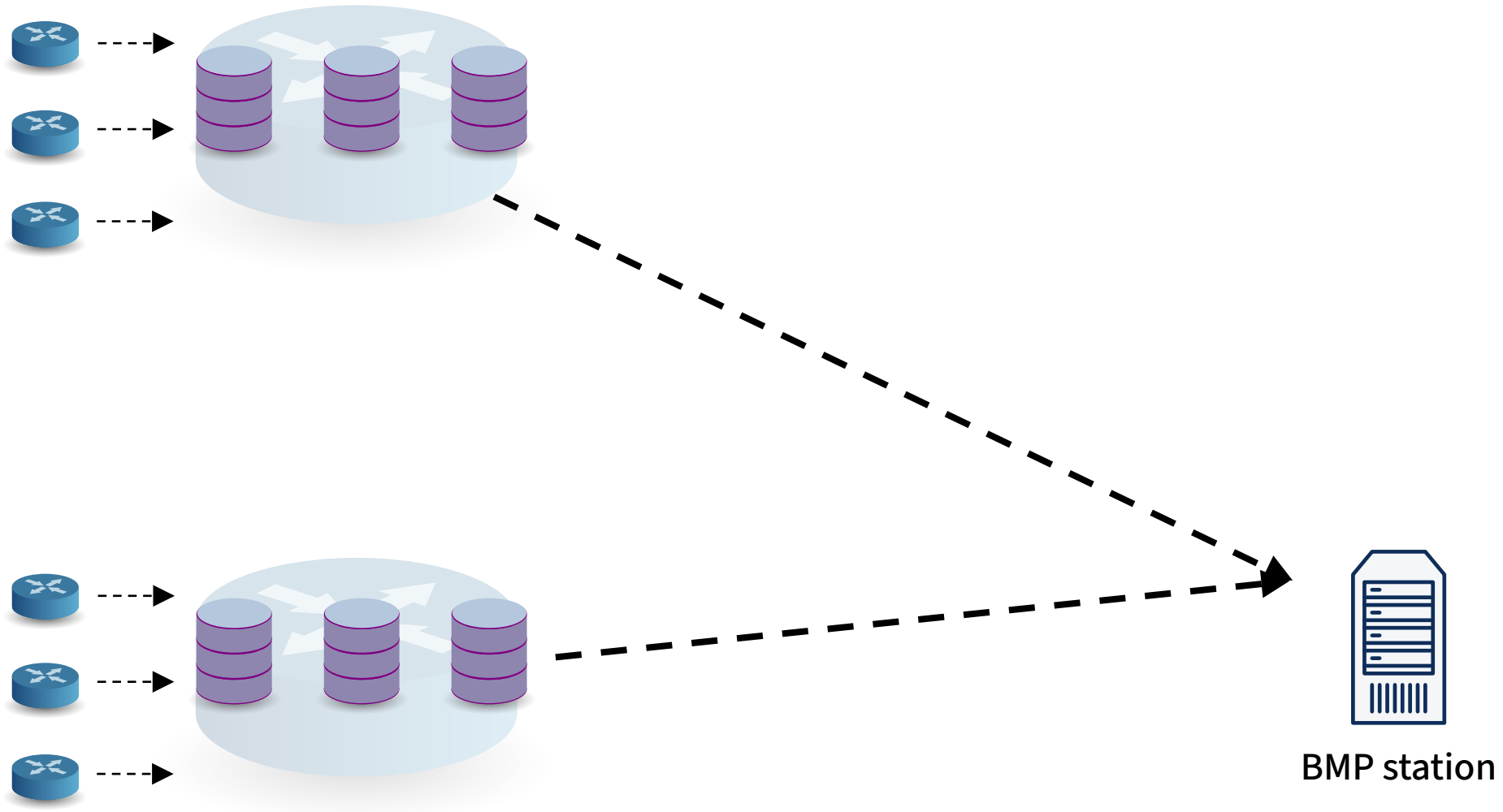

Peer AS 2
Adj-RIB-in
RIB-Loc
Adj-RIB-out
BMP station

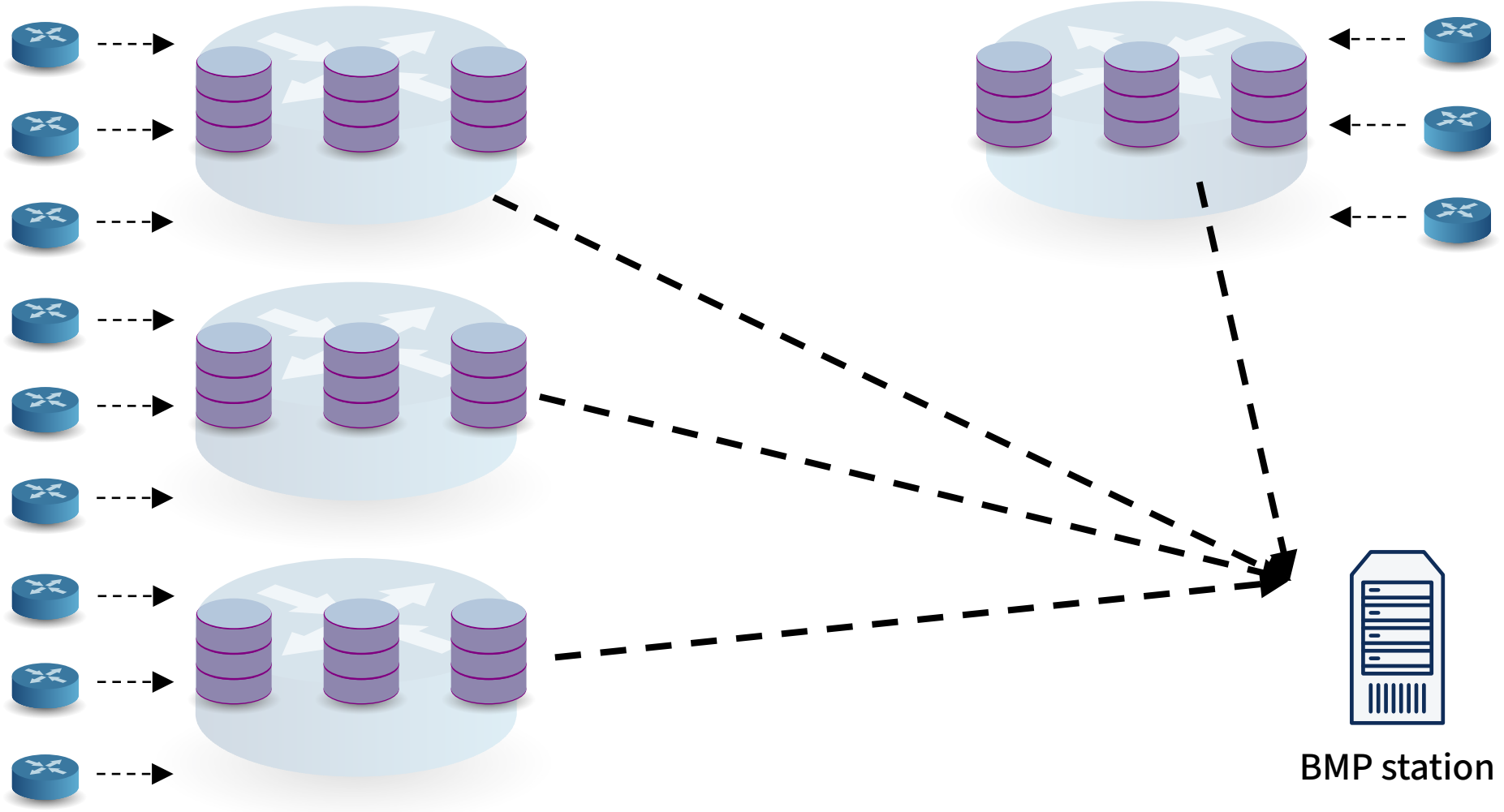

Peer AS 3
Peer AS 2
Peer AS 4
Adj-RIB-in
RIB-Loc
Adj-RIB-out
BMP station

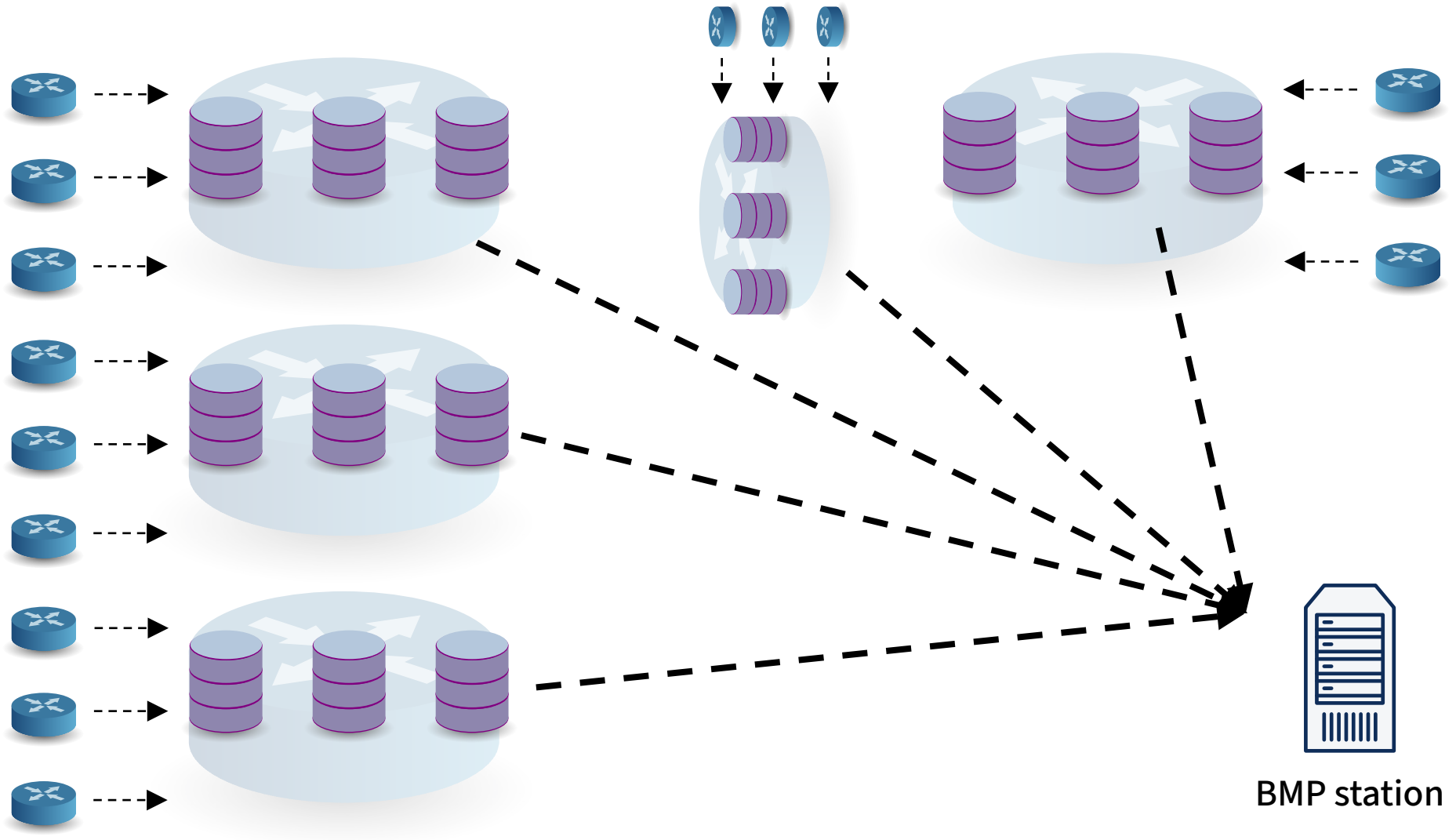


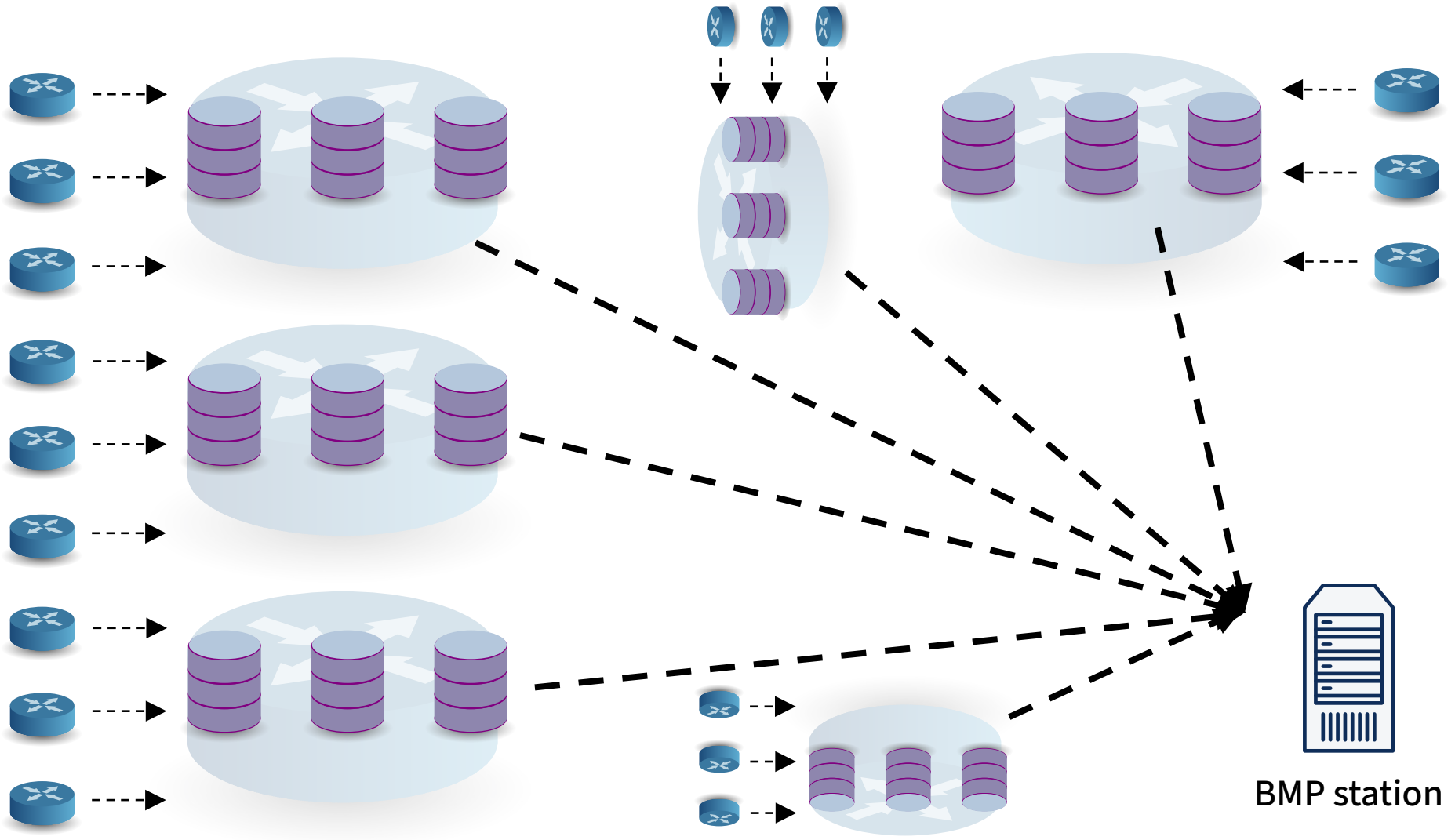
BMP station


BMP station


BMP station


BMP station

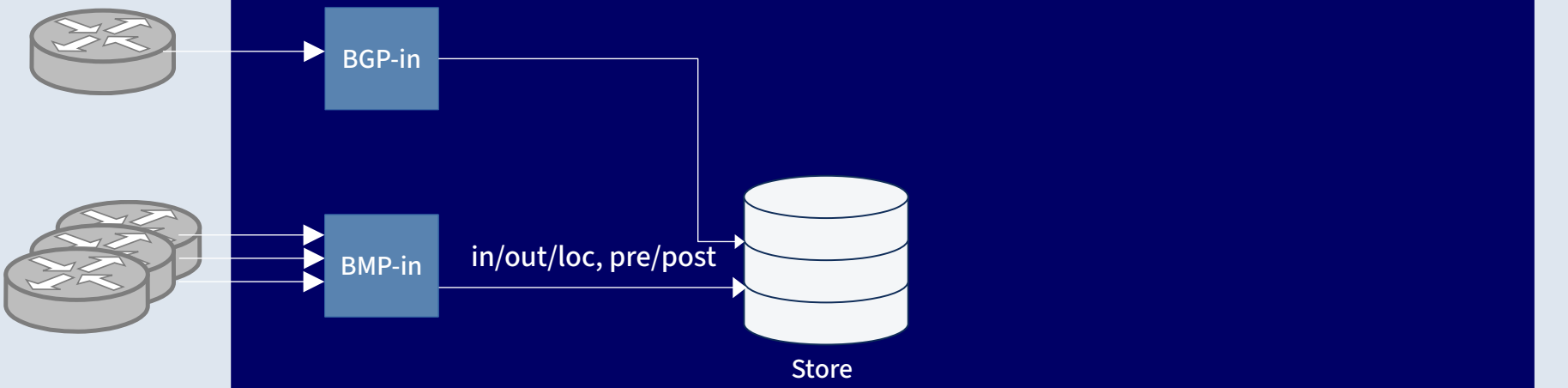

BMP station

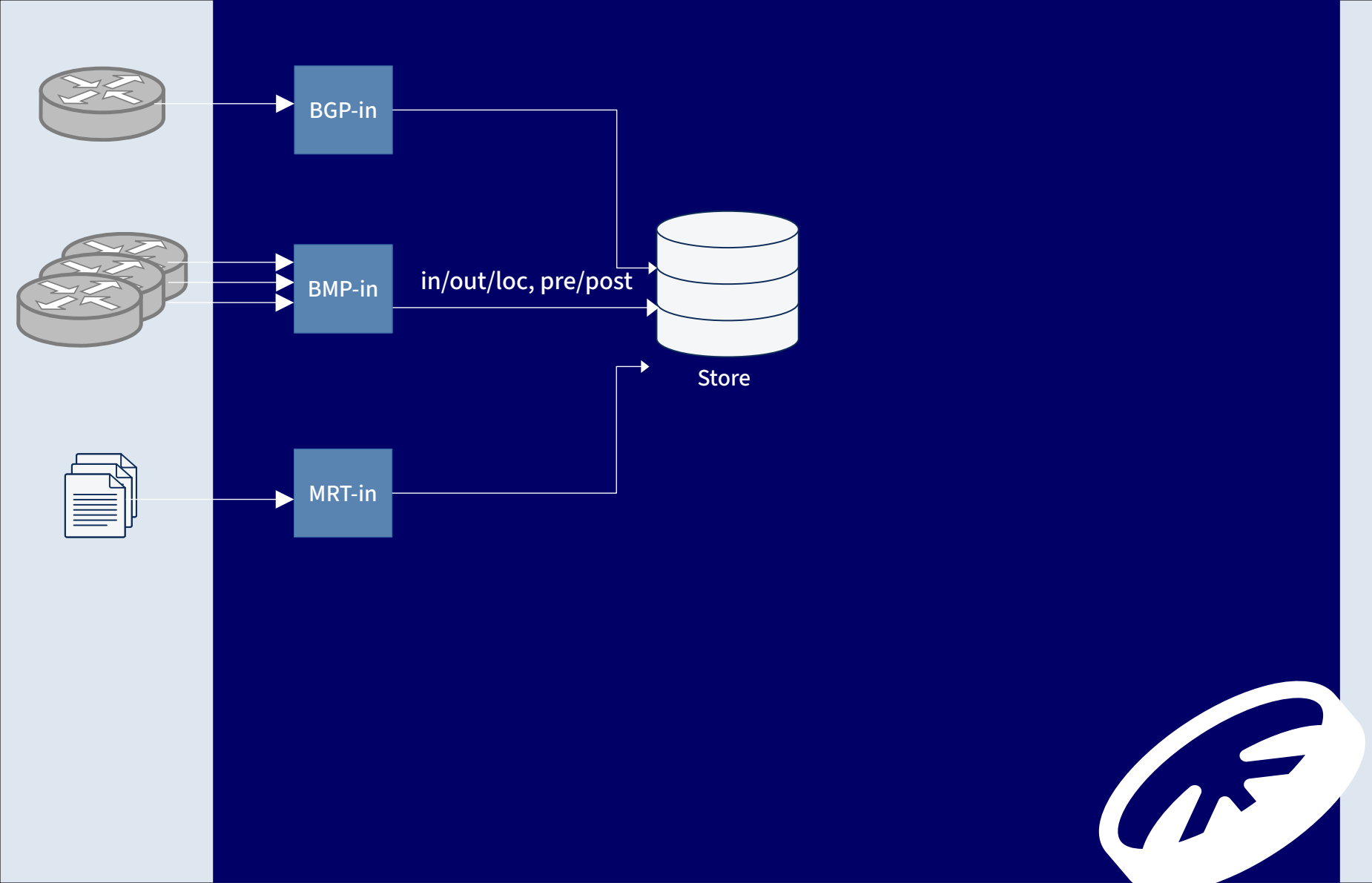

BMP station

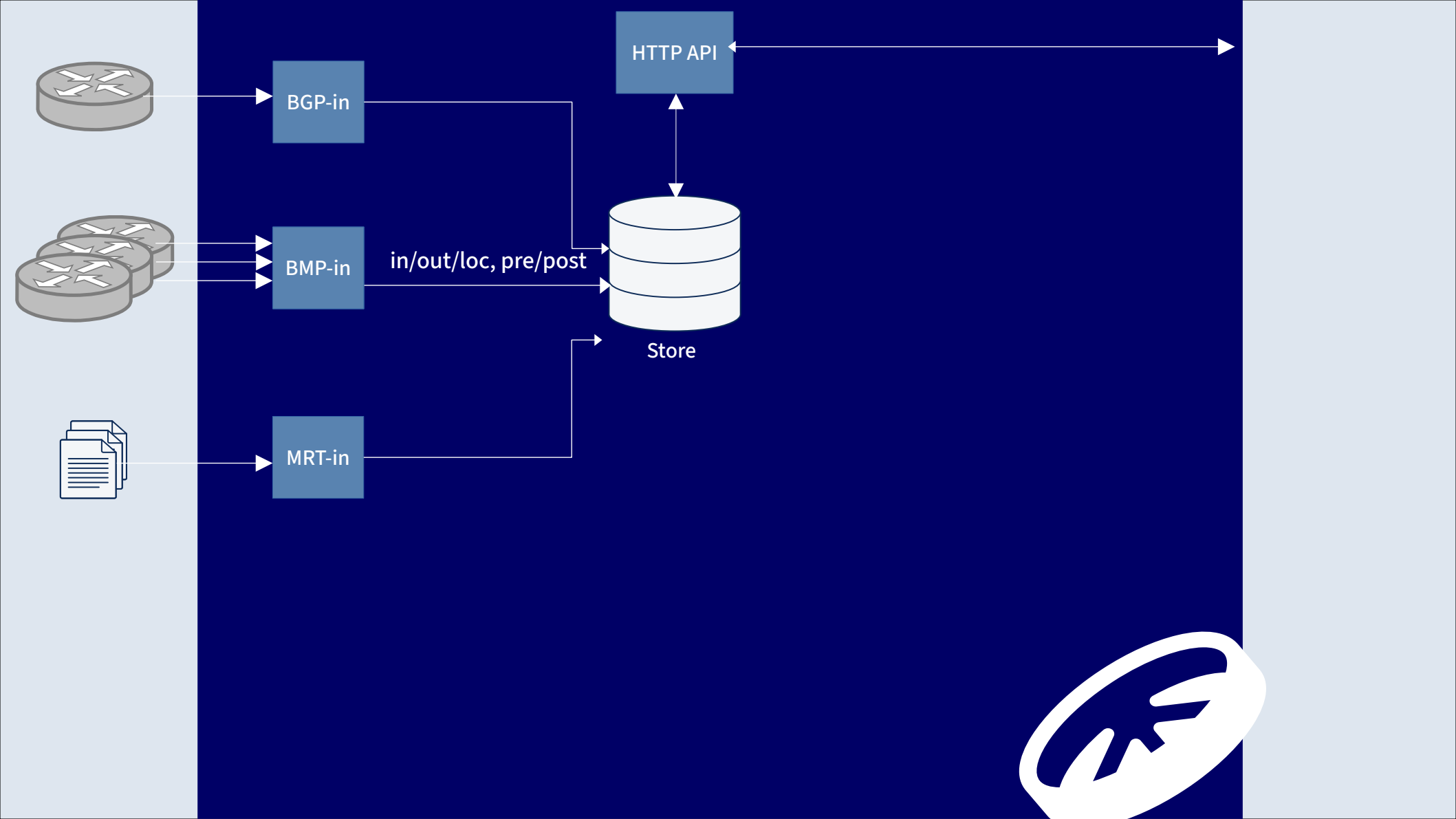

Store

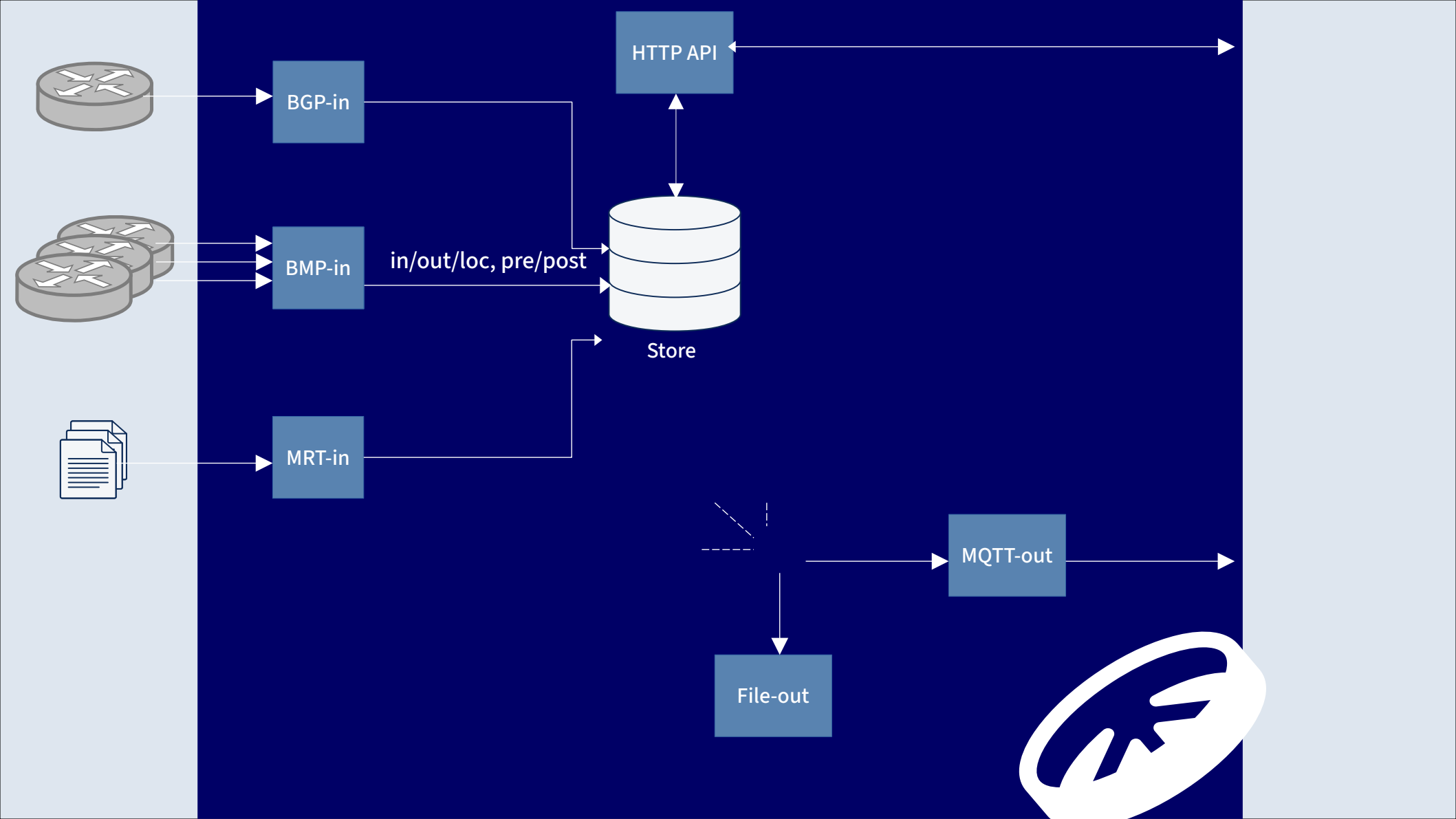

BMP-in
Store


BMP-in
in/out/loc, pre/post
Store


BGP-in
BMP-in
in/out/loc, pre/post
Store

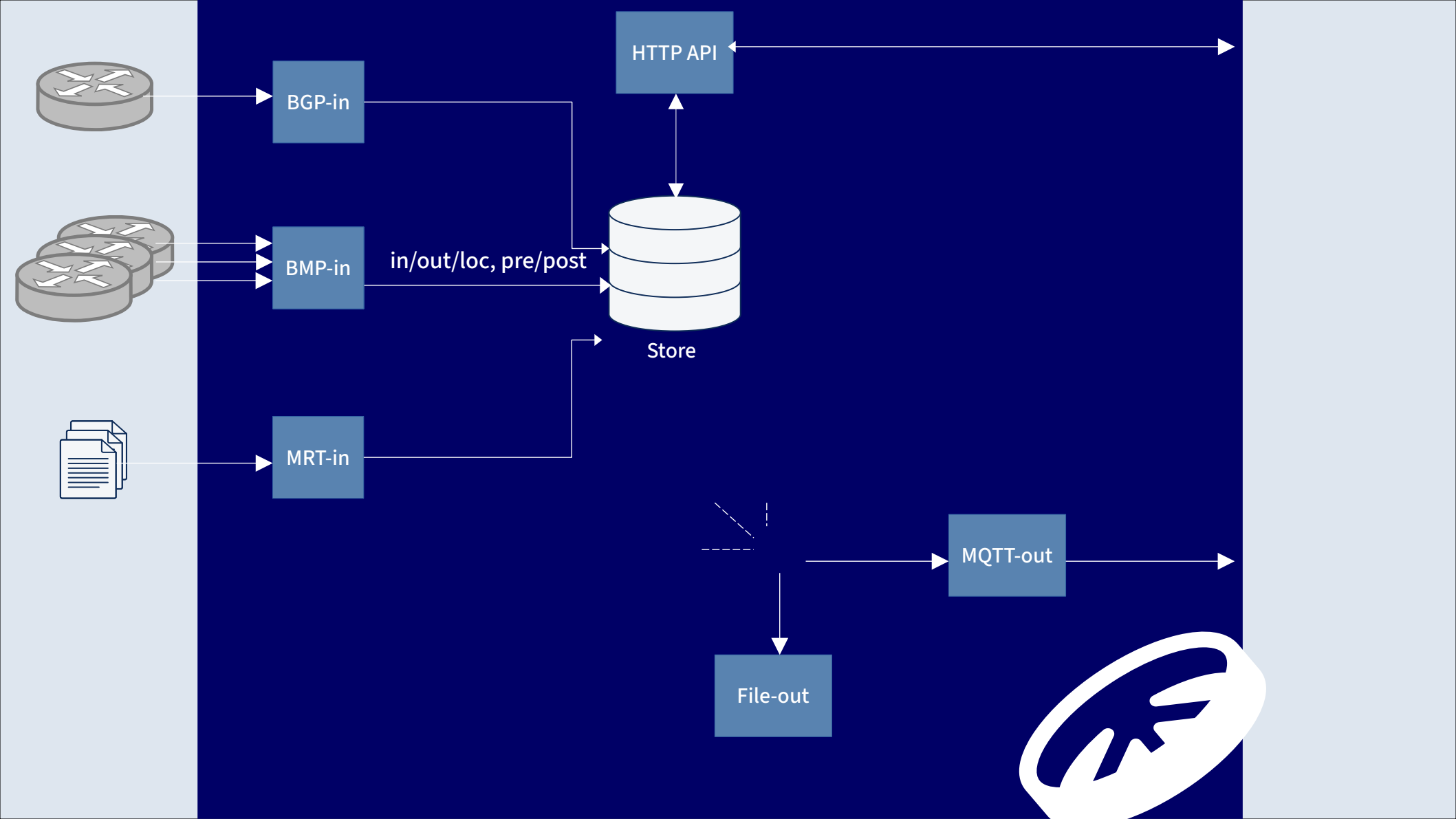

BGP-in
BMP-in
in/out/loc, pre/post
Store
MRT-in

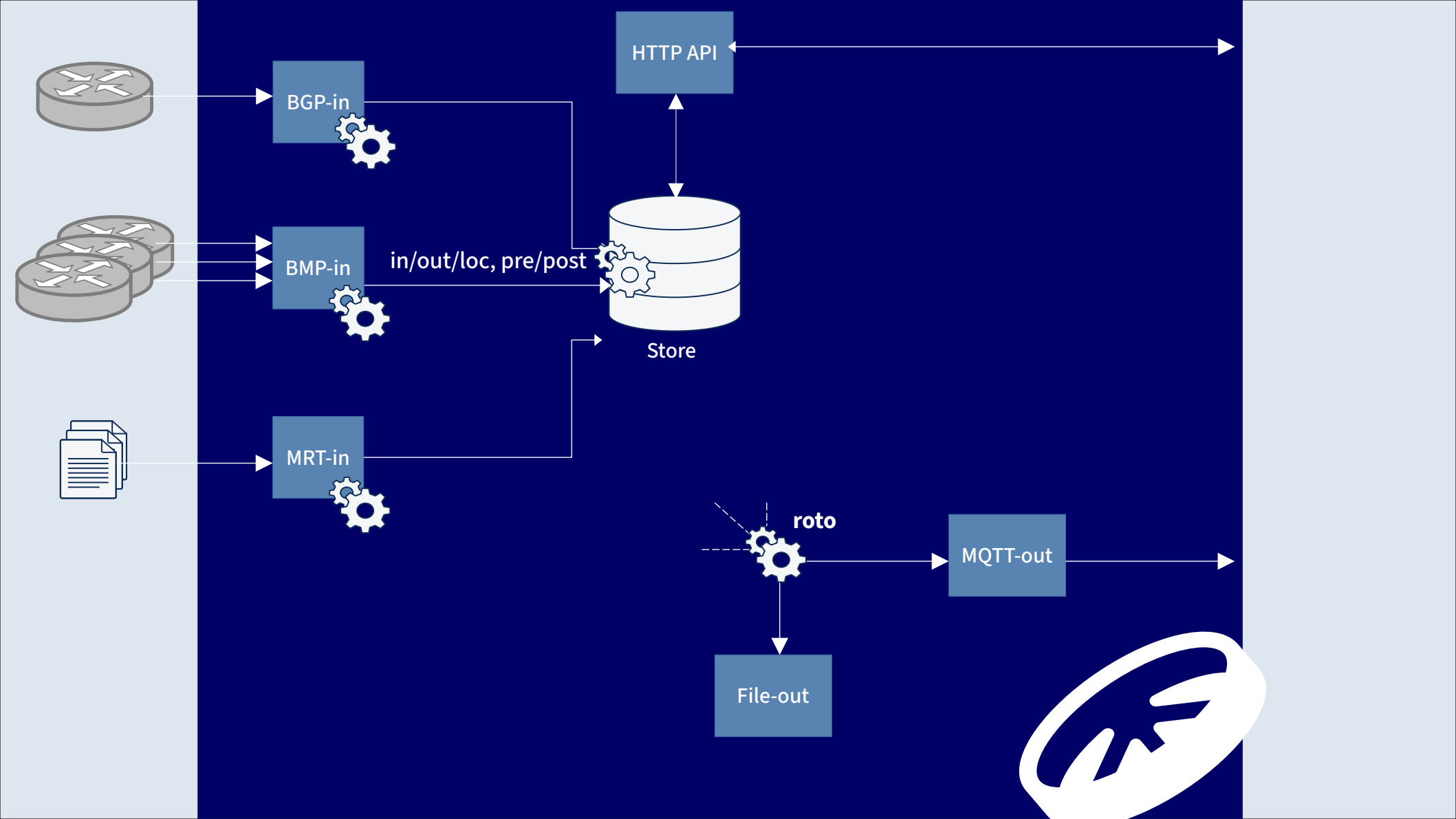

HTTP API
BGP-in
BMP-in
in/out/loc, pre/post
Store
MRT-in


HTTP API
BGP-in
BMP-in
in/out/loc, pre/post
Store
MRT-in
MQTT-out
File-out

Rotonda ...

- ingests routing data, and *keeps state;*
- is a specialized database;
- is one single, self-contained Rust application;
- aims at high volume, high performance, high flexibility.


HTTP API
BGP-in
BMP-in
in/out/loc, pre/post
Store
MRT-in
MQTT-out
File-out


HTTP API
BGP-in
BMP-in
in/out/loc, pre/post
Store
MRT-in
roto
MQTT-out
File-out

Introducing Roto

Roto: high-level feel

Syntax highlighting!

```
filter bmp_in(
    bmp_msg: BmpMsg,
    ingress_info: IngressInfo,
) {
    let my_asn = AS211321;

    if bmp_msg.is_ibgp(my_asn) {
        reject
    } else {
        accept
    }
}
```

Roto: high-level feel, strongly typed

Type checking and useful error messages!

```
filter bmp_in(
    bmp_msg: BmpMsg,
    ingress_info: IngressInfo,
) {
    let my_asn = 185.49.140.0/23; # ← oops

    if bmp_msg.is_ibgp(my_asn) {
        reject
    } else {
        accept
    }
}
```

Roto: high-level feel, strongly typed

Type checking and useful error messages!

```
filter bmp_in(
    bmp_msg: BmpMsg,
    ingress_info: IngressInfo,
```

```
Unable to load main Roto script: Error: Type error: mismatched types
    [ /home/luuk/code/rotonda/cleanup-branch/etc/filters.roto:51:24 ]

 51         if bmp_msg.is_ibgp(my_asn) {
                               ^^^^^^ expected `Asn`, found `Prefix`
```

```
}
```

Roto: high-level feel, strongly typed, compiled

Compiled!

```
filter bmp_in(
    bmp_msg: BmpMsg,
    ingress_info: IngressInfo,
) {
    let my_asn = AS211321;

    if bmp_msg.is_ibgp(my_asn) {
        reject
    } else {
        accept
    }
}
```

Roto: high-level feel, strongly typed, compiled

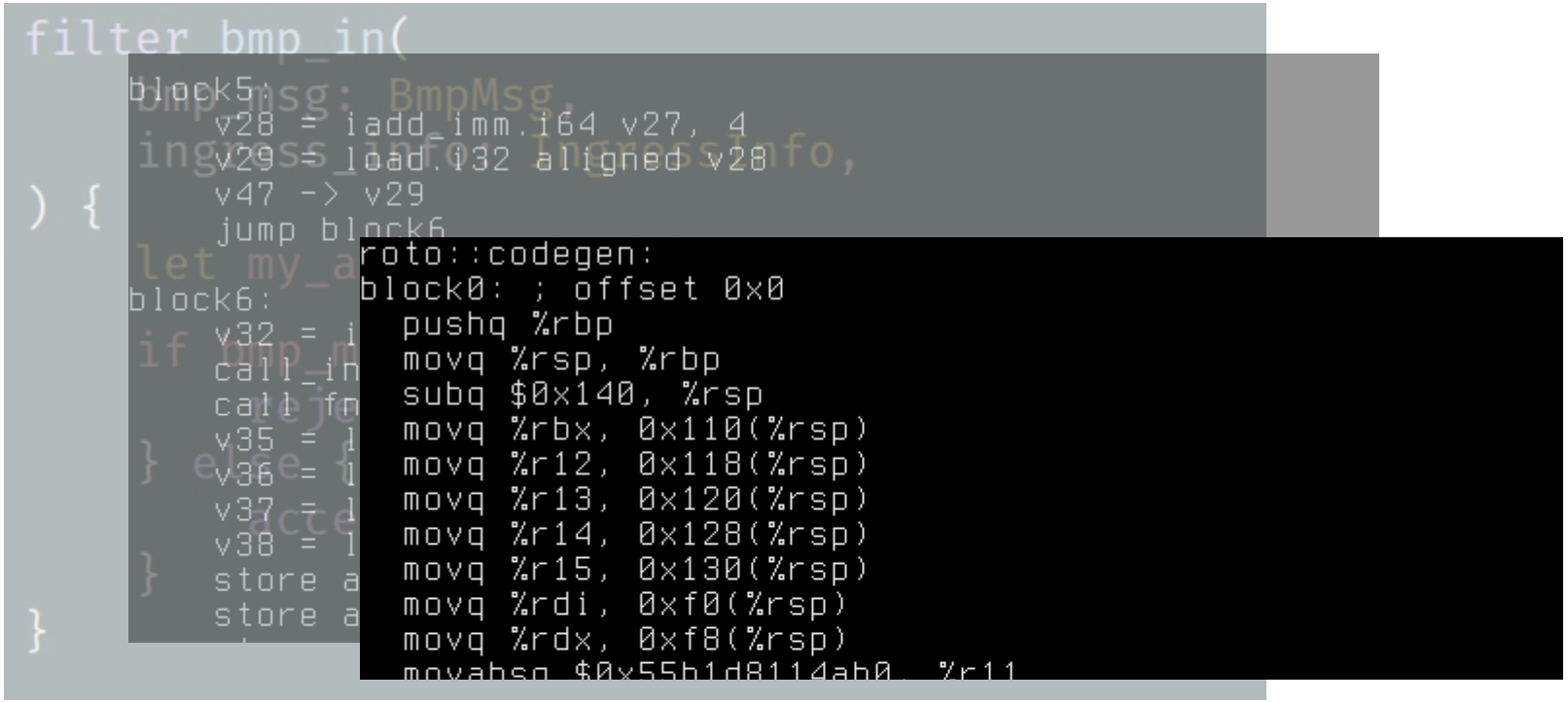
Compiled! From Roto to Cranelift IR,

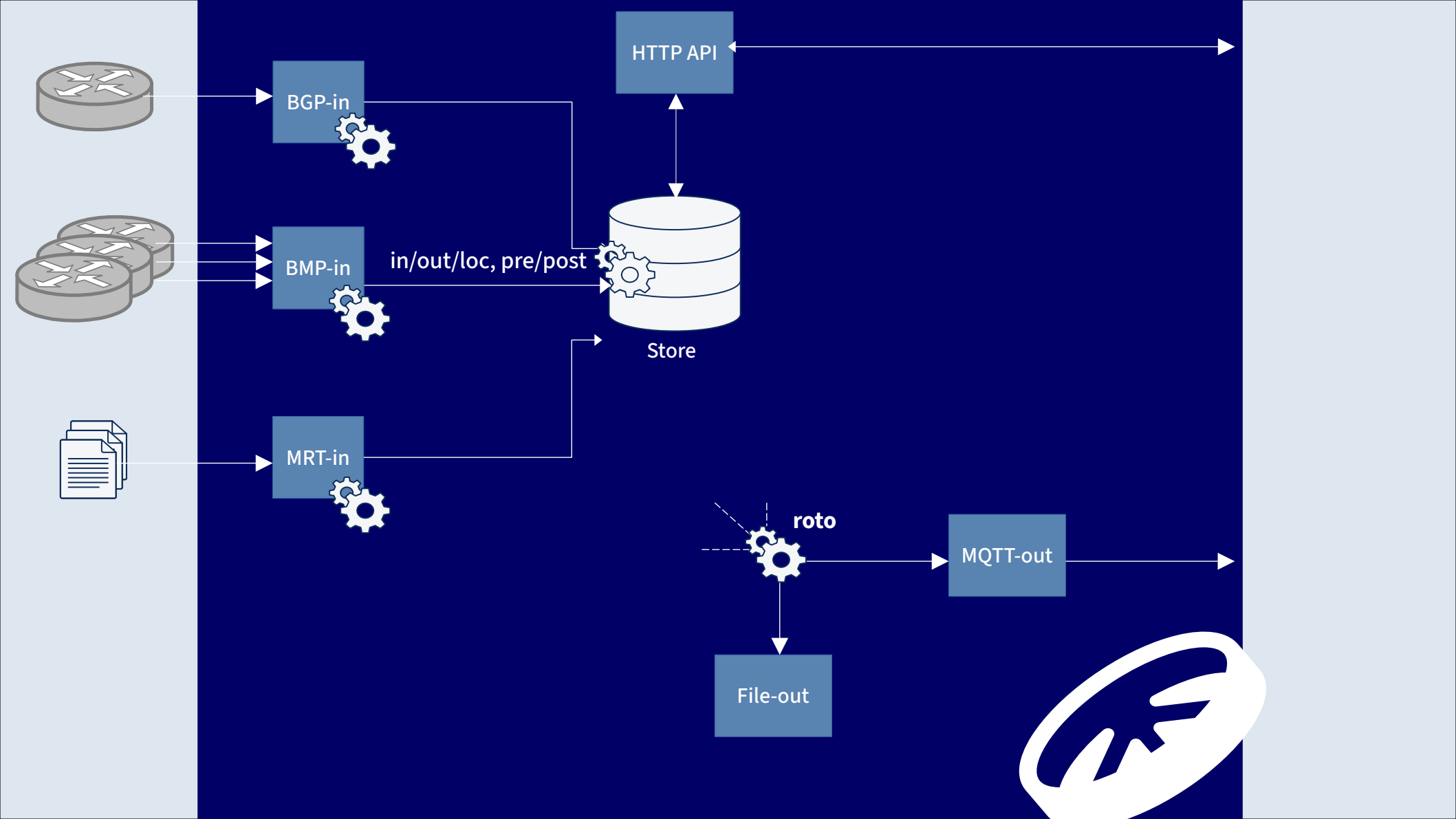
```
filter bmp_in(
) {
}
```

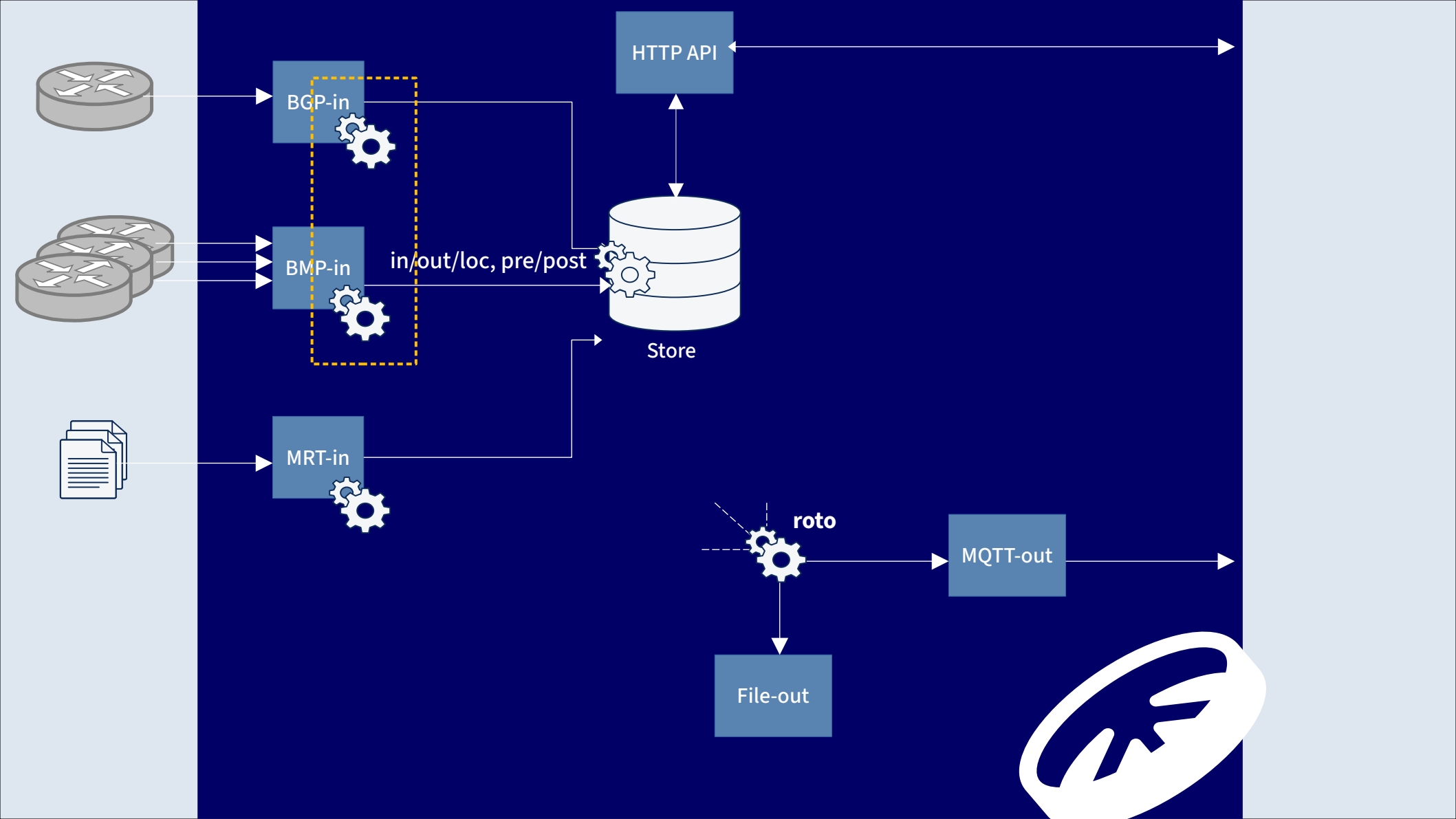
```
block5:
    v28 = iadd_imm.i64 v27, 4
    v29 = load.i32 aligned v28
    v47 -> v29
    jump block6

block6:
    v32 = iconst.i64 0x558f_2cfd_0ab0
    call_indirect sig1, v32(v31, v30)  ; v32 = 0x558f_2cfd_
    call fn1(v33, v30)
    v35 = load.i64 aligned v33
    v36 = load.i64 aligned v33+8
    v37 = load.i64 aligned v33+16
    v38 = load.i64 aligned v33+24
    store aligned v35, v34
    store aligned v36, v34+8
```

Roto: high-level feel, strongly typed, compiled

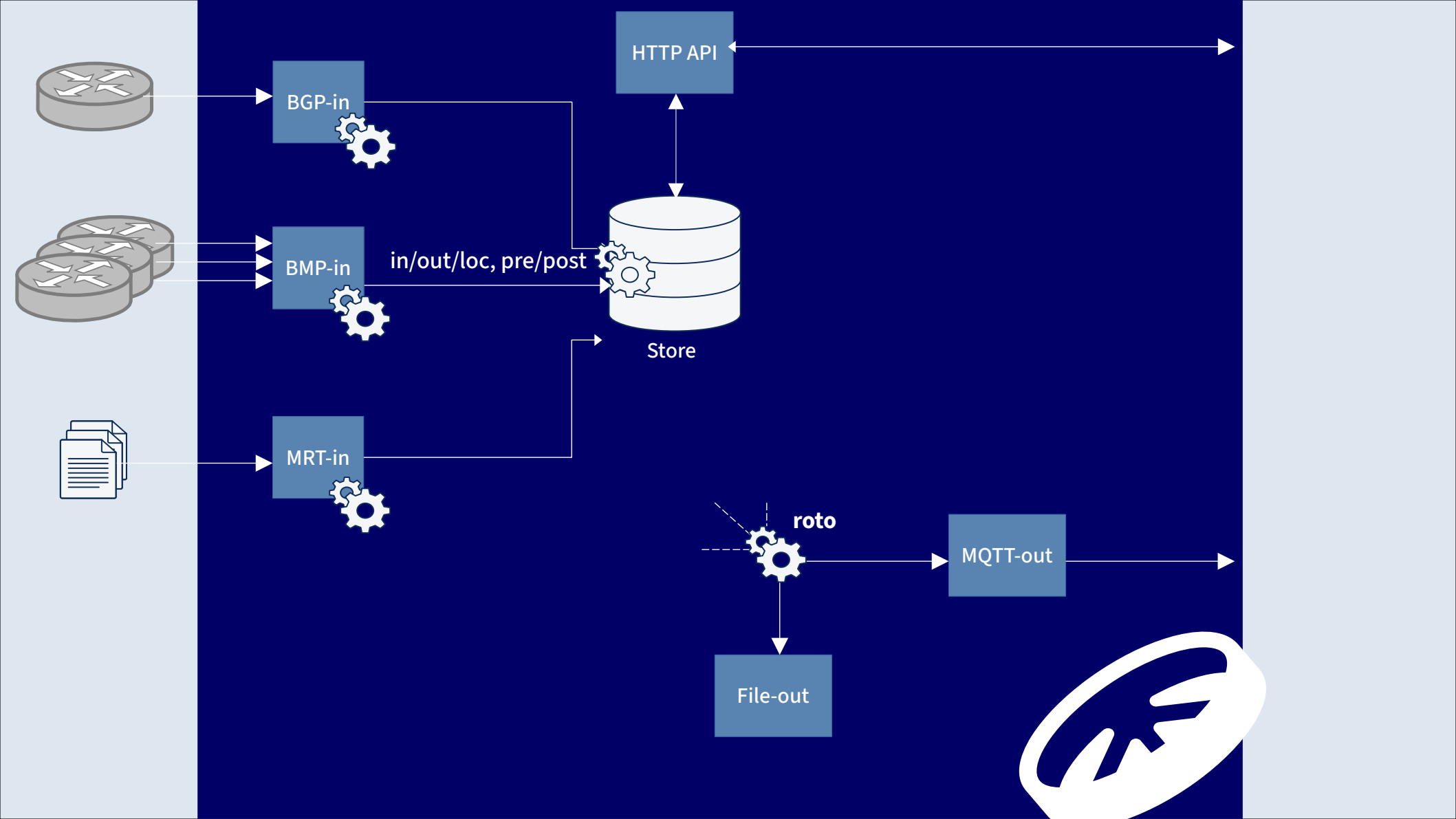
Compiled! From Roto to Cranelift IR, to actual x86_64 (or ARM64)!

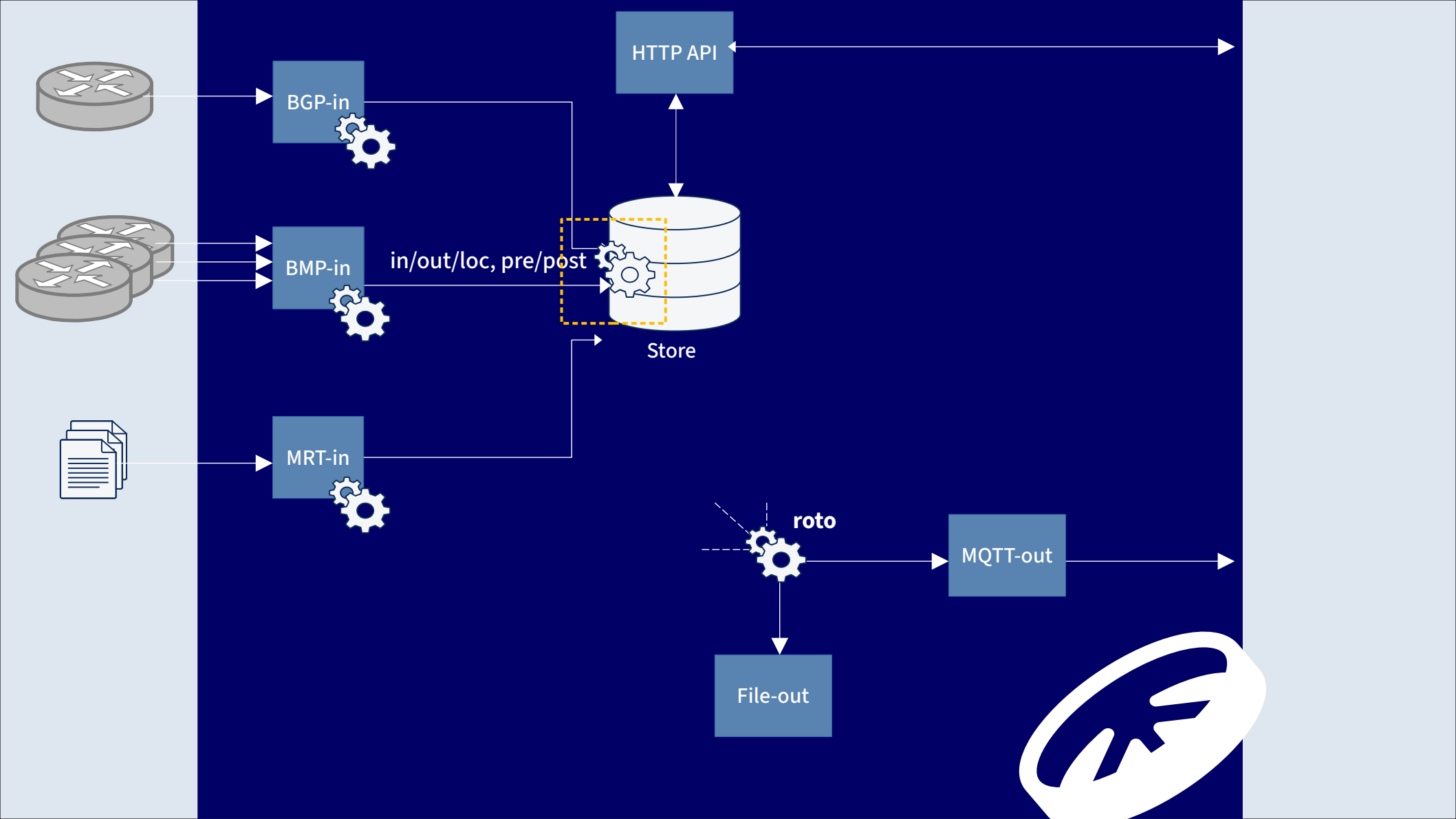



HTTP API
BGP-in
BMP-in
in/out/loc, pre/post
Store
MRT-in
roto
MQTT-out
File-out


HTTP API
BGP-in
BMP-in
in/out/loc, pre/post
Store
MRT-in
roto
MQTT-out
File-out

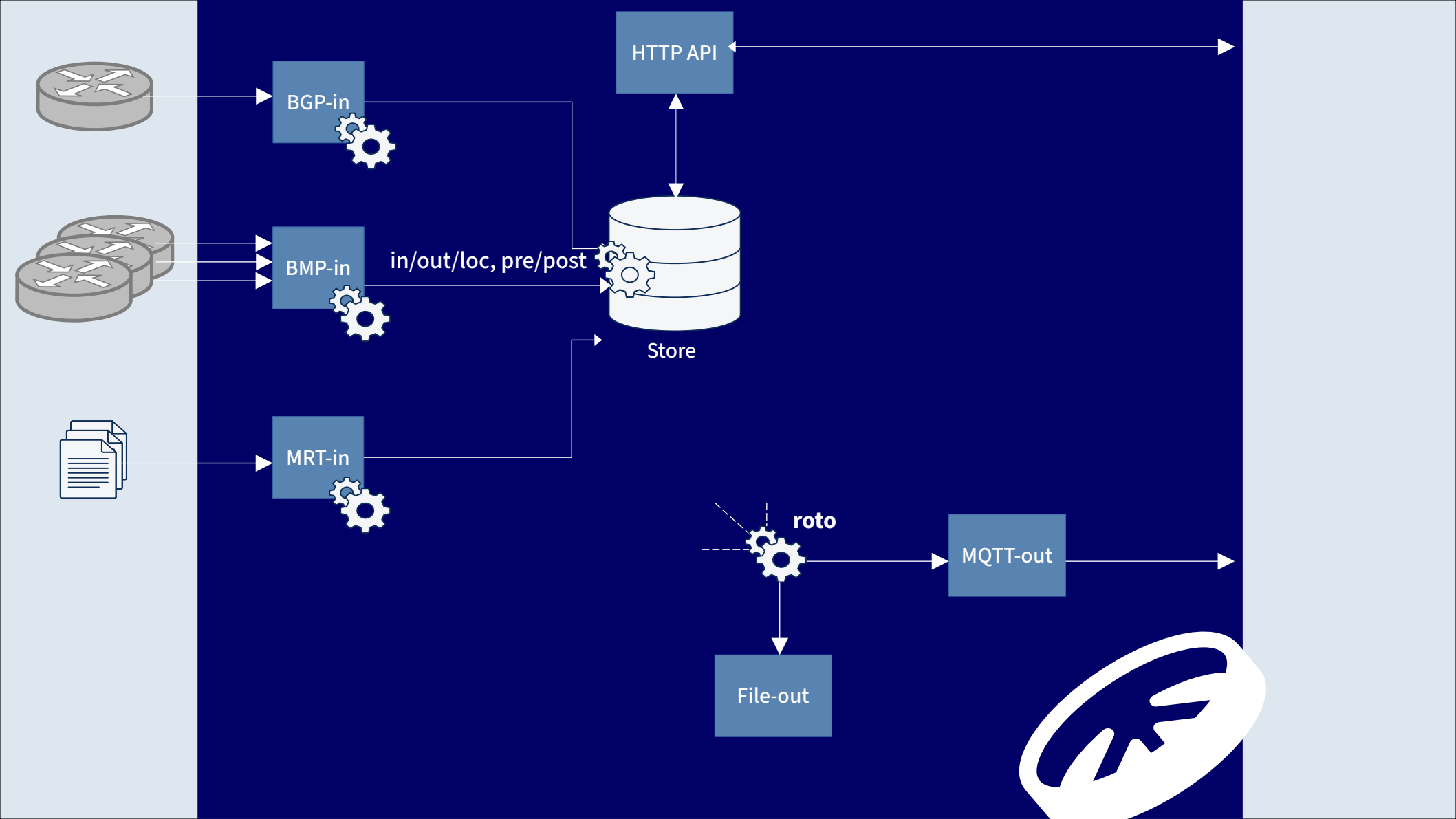
```
filter bmp_in(
    bmp_msg: BmpMsg,
    ingress_info: IngressInfo,
) {

    if ingress_info.peer_address() == 1.2.3.4 {
        return reject;
    }

    if ingress_info.peer_asn() == AS211321 {
        return reject;
    }

    if bmp_msg.aspath_contains(AS211321) {
        return reject;
    }

    accept
}
```

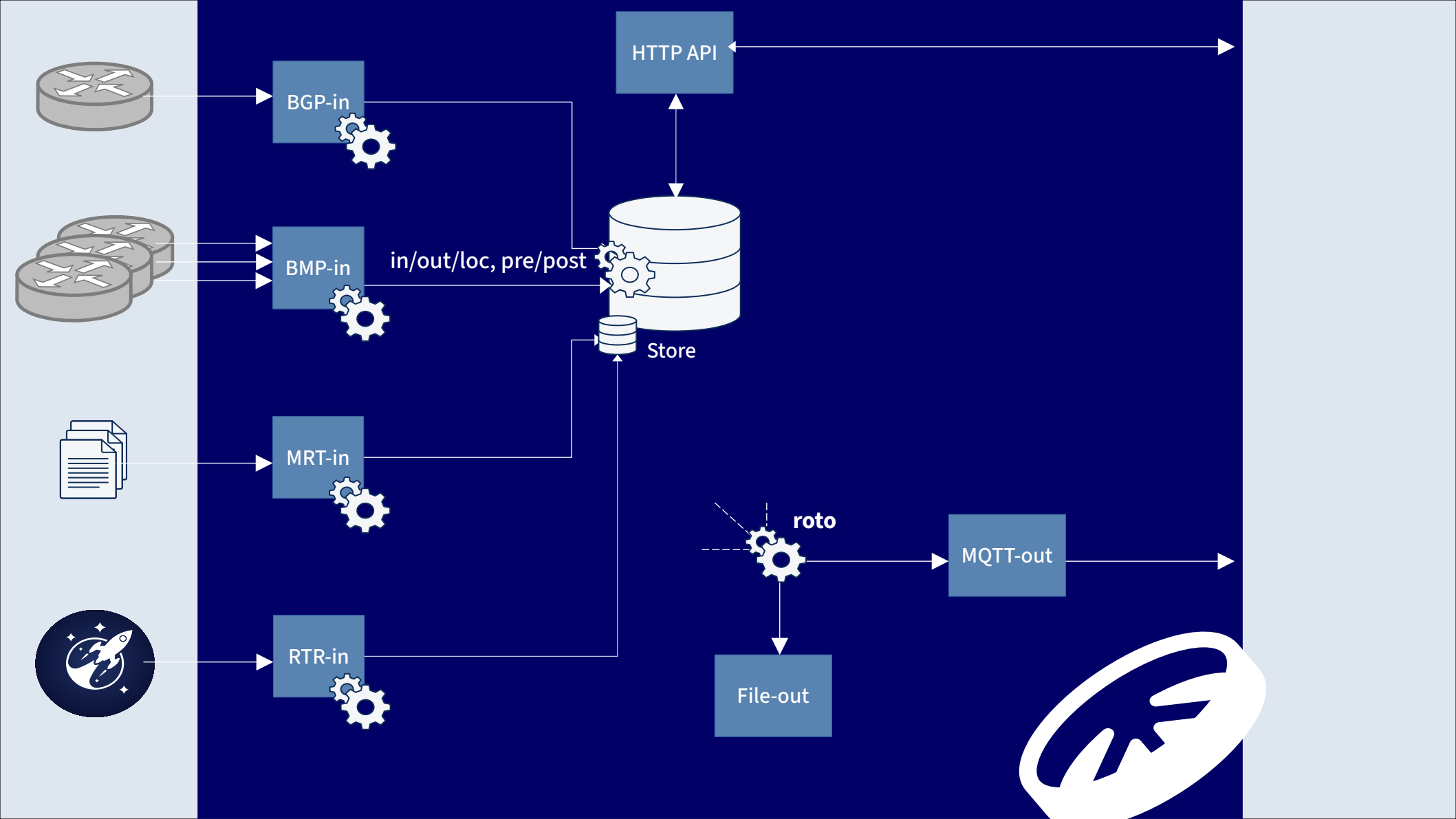

HTTP API
BGP-in
BMP-in
in/out/loc, pre/post
Store
MRT-in
roto
MQTT-out
File-out

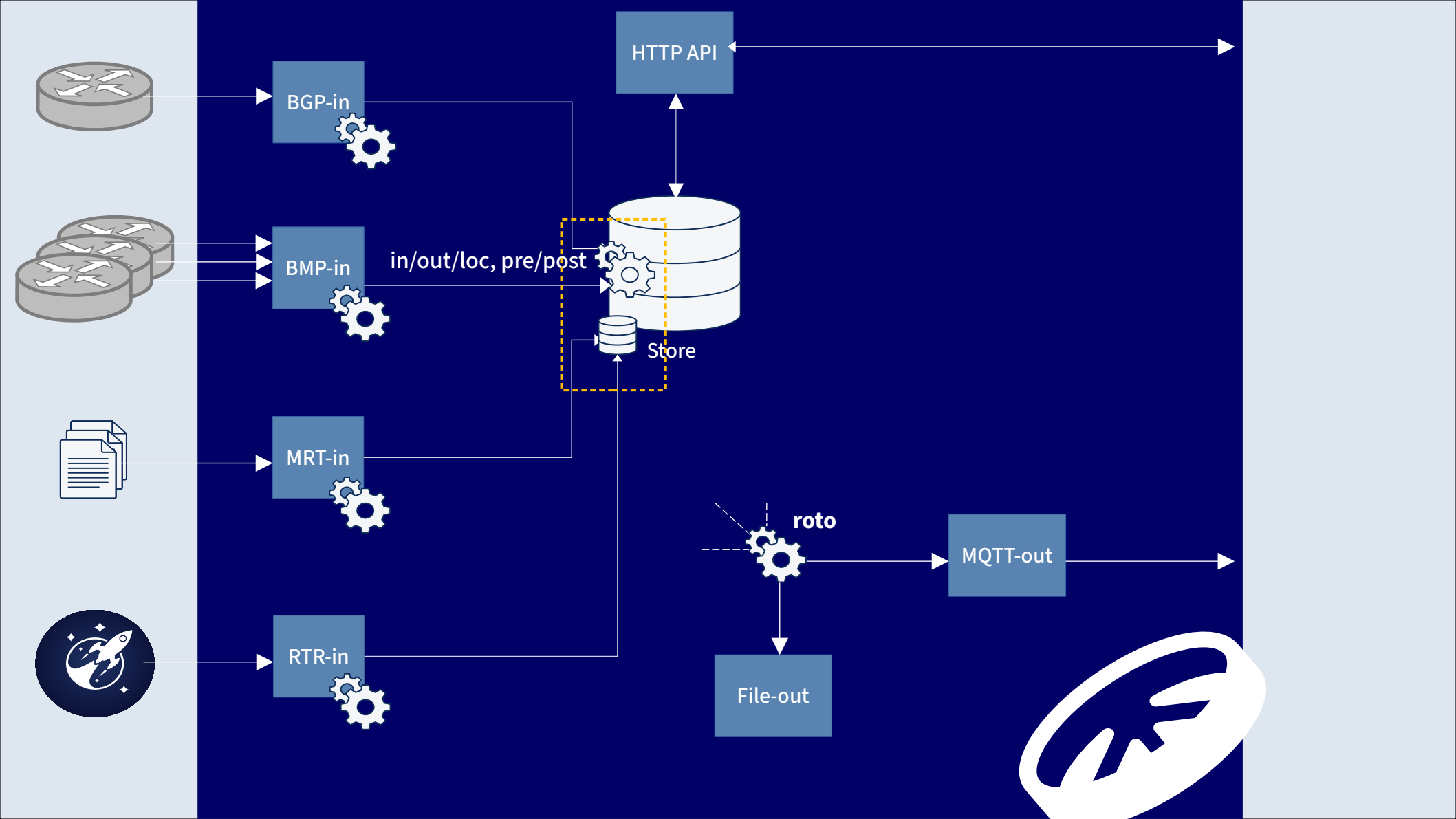

HTTP API
BGP-in
BMP-in
in/out/loc, pre/post
Store
MRT-in
roto
MQTT-out
File-out

```
filter rib_in_pre(
    route: Route,
    ingress_info: IngressInfo,
) {

    if prefix_lists.contains("my_prefixes", route.prefix()) {
        output.log_prefix(route.prefix());
    }




    accept
}
```

```
filter rib_in_pre(
    route: Route,
    ingress_info: IngressInfo,
) {

    if prefix_lists.contains("my_prefixes", route.prefix()) {
        output.log_prefix(route.prefix());
    }



    accept
}
```

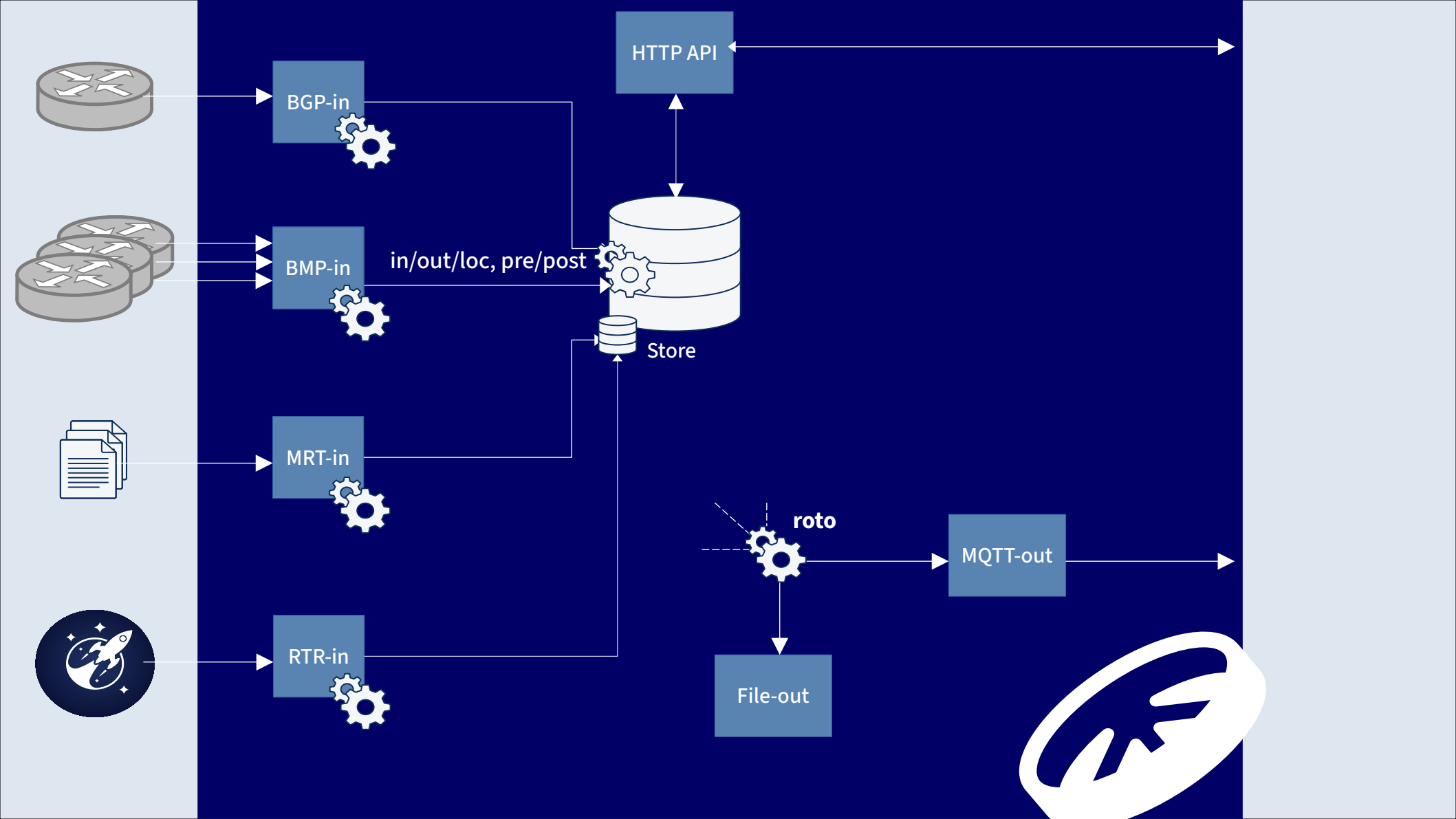

HTTP API
BGP-in
BMP-in
in/out/loc, pre/post
Store
MRT-in
roto
MQTT-out
File-out

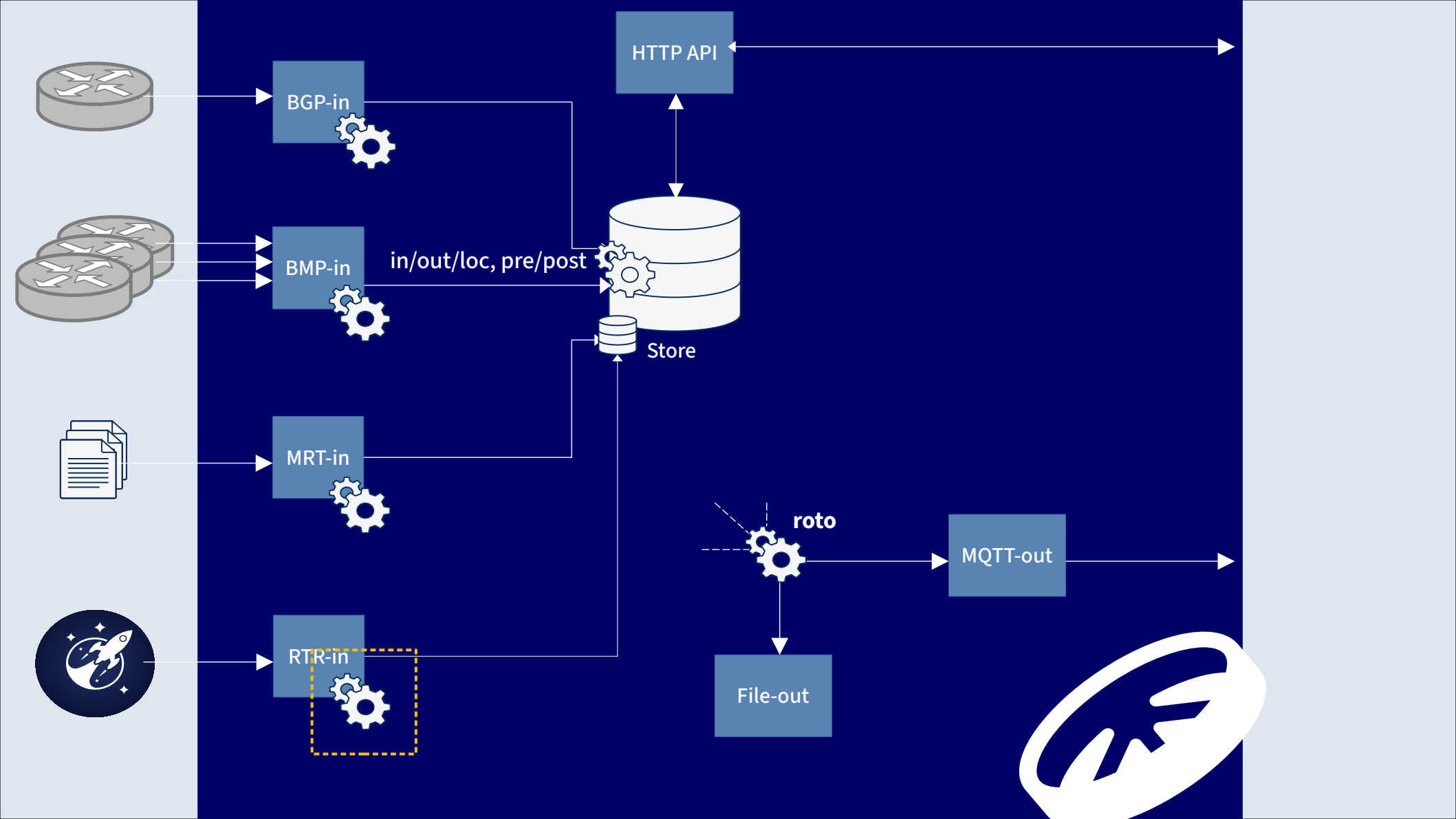

HTTP API
BGP-in
BMP-in
in/out/loc, pre/post
Store
MRT-in
roto
MQTT-out
RTR-in
File-out


HTTP API
BGP-in
BMP-in
in/out/loc, pre/post
Store
MRT-in
RTR-in
roto
MQTT-out
File-out

```
filter rib_in_pre(
    route: Route,
    ingress_info: IngressInfo,
) {

    if prefix_lists.contains("my_prefixes", route.prefix()) {
        output.log_prefix(route.prefix());
    }




    accept
}
```

```
filter rib_in_pre(
    route: Route,
    ingress_info: IngressInfo,
) {

    if prefix_lists.contains("my_prefixes", route.prefix()) {
        output.log_prefix(route.prefix());
    }

    # Only has effect when an RTR unit is configured in rotonda.conf
    rpki.check_rov(route);

    accept
}
```


HTTP API
BGP-in
BMP-in
in/out/loc, pre/post
Store
MRT-in
roto
MQTT-out
RTR-in
File-out


HTTP API
BGP-in
BMP-in
in/out/loc, pre/post
Store
MRT-in
roto
MQTT-out
RTR-in
File-out

Act on updates coming in from the RPKI

```
filter vrp_update(vrp_update: VrpUpdate) {

    if asn_lists.contains("my_asns", vrp_update.asn()) {
        output.entry().timestamped_custom("[RTR] (my asns): " + vrp_update.fmt());
        output.write_entry();
    }

    if prefix_lists.covers("my_prefixes", vrp_update.prefix()) {
        output.timestamped_print("[RTR] (my prefixes): " + vrp_update.fmt())
    }

    accept
}
```

Act on how an RPKI update affects your routes

```
fn rib_in_rov_status_update(rov_update: RovStatusUpdate) {

    if rov_update.has_changed() {
        # Only log whenever something changed to Invalid:
        if rov_update.current_status().is_invalid() {
            output.timestamped_print("[RTR]" + rov_update.fmt())
        }
    }
    # When it concerns any of 'my_prefixes', always log:
    if prefix_lists.covers("my_prefixes", rov_update.prefix()) {
        output.timestamped_print("[RTR] (my prefixes): " + rov_update.fmt())
    }
}
```

Act on how an RPKI update affects your routes

```
fn rib_in_rov_status_update(rov_update: RovStatusUpdate) {

    if rov_update.has_changed() {
        # Only log whenever something changed to Invalid:
        if rov_update.current_status().is_invalid() {
            output.timestamped_print("[RTR]" + rov_update.fmt())
        }
    }
    # When it concerns any of 'my_prefixes', always log:
    if prefix_lists.covers("my_prefixes", rov_update.prefix()) {
        output.timestamped_print("[RTR] (my prefixes): " + rov_update.fmt())
    }
}
```

Act on how an RPKI update affects your routes

```
fn rib_in_rov_status_update(rov_update: RovStatusUpdate) {

    if rov_update.has_changed() {
        # Only log whenever something changed to Invalid:
        if rov_update.current_status().is_invalid() {
            output.timestamped_print("[RTR]" + rov_update.fmt())
        }
    }
    # When it concerns any of 'my_prefixes', always log:
    if prefix_lists.covers("my_prefixes", rov_update.prefix()) {
        output.timestamped_print("[RTR] (my prefixes): " + rov_update.fmt())
    }
}
```

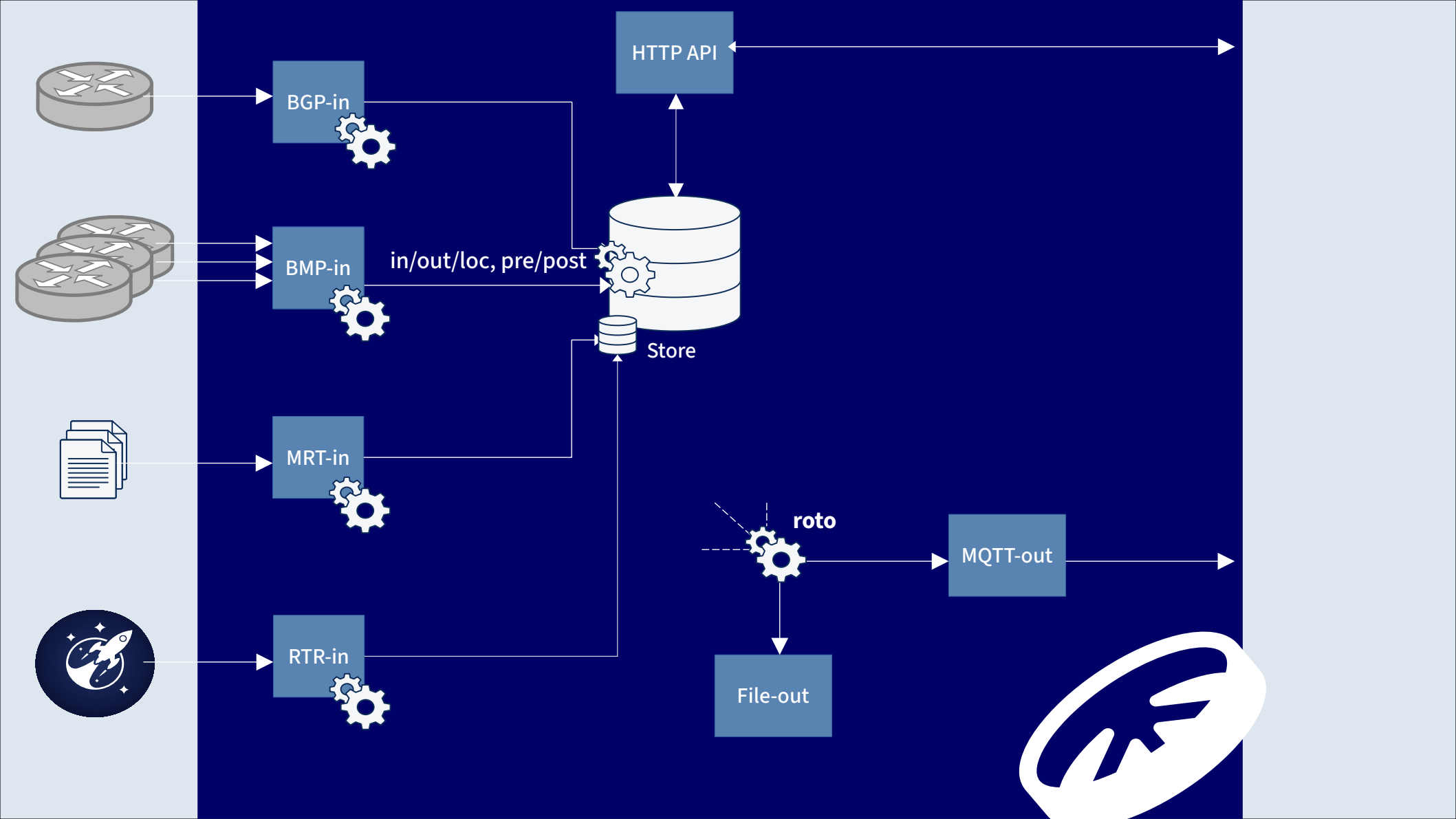
Act on how an RPKI update affects your routes

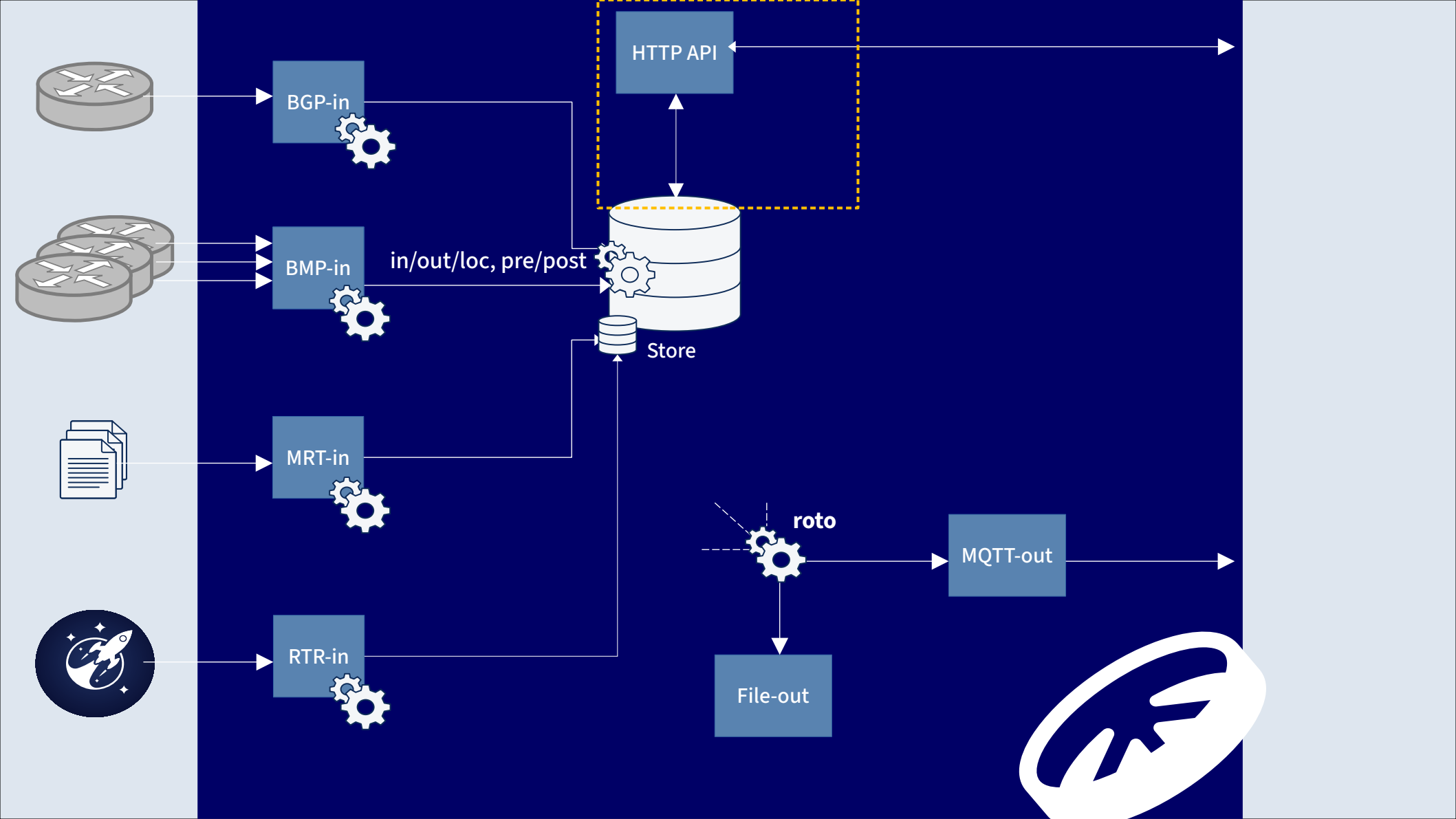
```
fn rib_in_rov_status_update(rov_update: RovStatusUpdate) {

    if rov_update.has_changed() {
        # Only log whenever something changed to Invalid:
        if rov_update.current_status().is_invalid() {
            output.timestamped_print("[RTR] " + rov_update.fmt())
        }
    }
    # When it concerns any of 'my_prefixes', always log:
    if prefix_lists.covers("my_prefixes", rov_update.prefix()) {
        output.timestamped_print("[RTR] (my prefixes): " + rov_update.fmt())
    }
}
```

Act on how an RPKI update affects your routes

```
fn rib_in_rov_status_update(rov_update: RovStatusUpdate) {

    if rov_update.has_changed() {
        # Only log whenever something changed to Invalid:
        if rov_update.current_status().is_invalid() {
```

```
rotonda::units::rib_unit::unit: got RTR Serial update
rotonda::units::rib_unit::unit: pushed VRP for 40.223.13.0/24-24 from AS214025
rotonda::units::rib_unit::unit: removed VRP for 40.223.12.0/22-24 from AS834
rotonda::units::rib_unit::unit: pushed VRP for 66.93.41.0/24-24 from AS214025
rotonda::units::rib_unit::unit: pushed VRP for 66.212.31.0/24-24 from AS214025
[2025-09-27T07:51:22Z] [RTR][NotFound] -> [Invalid] 66.212.31.0/24 originated by AS834, learned from AS200242
[2025-09-27T07:51:22Z] [RTR][NotFound] -> [Invalid] 66.212.31.0/24 originated by AS834, learned from AS213151
[2025-09-27T07:51:22Z] [RTR][NotFound] -> [Invalid] 66.212.31.0/24 originated by AS834, learned from AS3214
[2025-09-27T07:51:22Z] [RTR][NotFound] -> [Invalid] 66.212.31.0/24 originated by AS834, learned from AS8888
[2025-09-27T07:51:22Z] [RTR][NotFound] -> [Invalid] 66.212.31.0/24 originated by AS834, learned from AS60150
[2025-09-27T07:51:22Z] [RTR][NotFound] -> [Invalid] 66.212.31.0/24 originated by AS834, learned from AS51019
[2025-09-27T07:51:22Z] [RTR][NotFound] -> [Invalid] 66.212.31.0/24 originated by AS834, learned from AS52025
```


HTTP API
BGP-in
BMP-in
in/out/loc, pre/post
Store
MRT-in
roto
MQTT-out
RTR-in
File-out

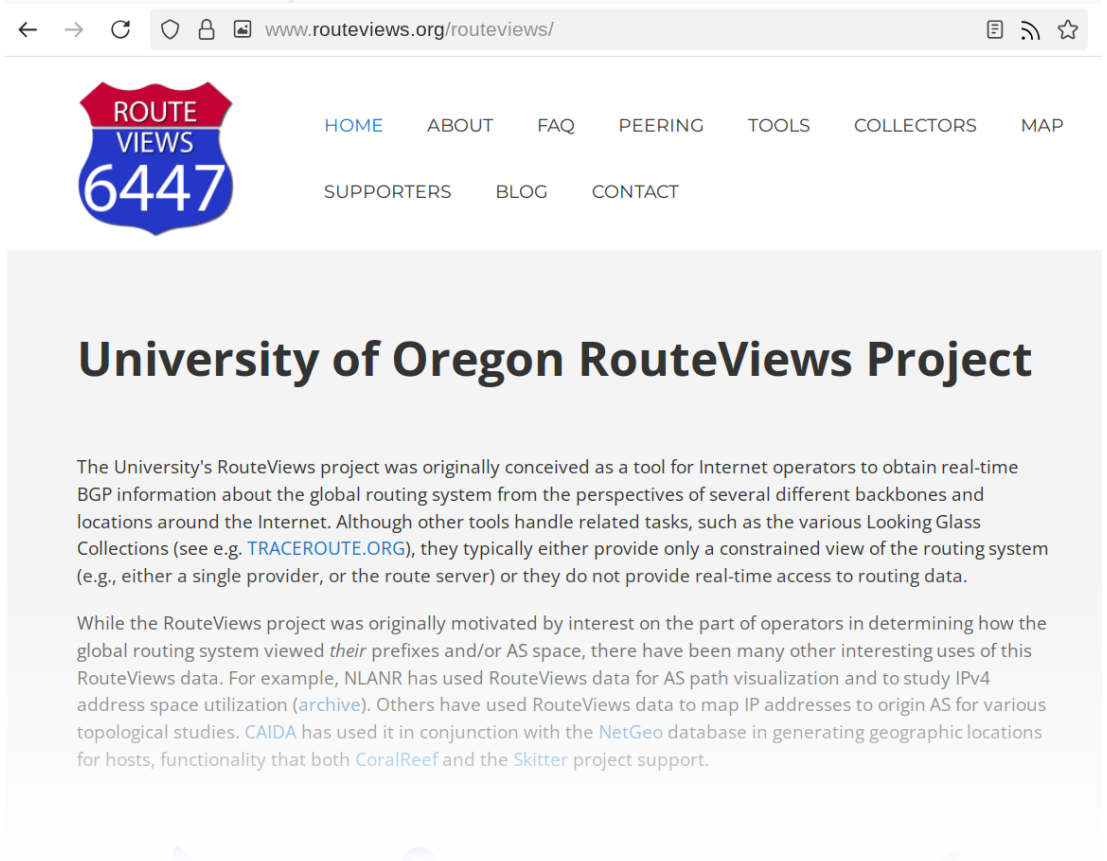

HTTP API
BGP-in
BMP-in
in/out/loc, pre/post
Store
MRT-in
RTR-in
roto
MQTT-out
File-out

Getting JSON out of Rotonda

- `/api/v1/ingresses[?filter[type]=bgp|bmp|...]`
  Query for *ingresses* and their session information

- `/api/v1/ribs/ipv4unicast/routes`
  `/api/v1/ribs/ipv4unicast/routes/185.49.140.0/23`
  Query for stored routes per address family

Thank you RouteViews!




```
"data": {
  "nlri": "185.49.140.0/23",
  "routes": [
    {
      "status": "active",
      "ingress": {
        "id": 38,
        "ingress_type": "bgpViaBmp",
        "parent_ingress": 2,
        "remote_addr": "185.1.167.88",
        "remote_asn": 51019,
        "rib_type": "AdjRibIn",
        "peer_rib_type": "inPost",
        "peer_type": "GlobalInstance"
      },
      "rpki": {
        "rov": "valid"
      },
      "pathAttributes": [
        {
          "asPath": [
            "AS6447",
            "AS51019",
            "AS34927",
            "AS50673",
            "AS8587"
          ]
        },
        {
          "standardCommunities": [
            "AS34927:930",
            "AS51019:1101"
          ]
```


```
"data": {
  "nlri": "185.49.140.0/23",
  "routes": [
    {
      "status": "active",
      "ingress": {
        "id": 38,
        "ingress_type": "bgpViaBmp",
        "parent_ingress": 2,
        "remote_addr": "185.1.167.88",
        "remote_asn": 51019,
        "rib_type": "AdjRibIn",
        "peer_rib_type": "inPost",
        "peer_type": "GlobalInstance"
      },
      "rpki": {
        "rov": "valid"
      },
      "pathAttributes": [
        {
          "asPath": [
            "AS6447",
            "AS51019",
            "AS34927",
            "AS50673",
            "AS8587"
          ]
        },
        {
          "standardCommunities": [
            "AS34927:930",
            "AS51019:1101"
          ]
```


```
"data": {
  "nlri": "185.49.140.0/23",
  "routes": [
    {
      "status": "active",
      "ingress": {
        "id": 38,
        "ingress_type": "bgpViaBmp",
        "parent_ingress": 2,
        "remote_addr": "185.1.167.88",
        "remote_asn": 51019,
        "rib_type": "AdjRibIn",
        "peer_rib_type": "inPost",
        "peer_type": "GlobalInstance"
      },
      "rpki": {
        "rov": "valid"
      },
      "pathAttributes": [
        {
          "asPath": [
            "AS6447",
            "AS51019",
            "AS34927",
            "AS50673",
            "AS8587"
          ]
        },
        {
          "standardCommunities": [
            "AS34927:930",
            "AS51019:1101"
          ]
```


```
"data": {
  "nlri": "185.49.140.0/23",
  "routes": [
    {
      "status": "active",
      "ingress": {
        "id": 38,
        "ingress_type": "bgpViaBmp",
        "parent_ingress": 2,
        "remote_addr": "185.1.167.88",
        "remote_asn": 51019,
        "rib_type": "AdjRibIn",
        "peer_rib_type": "inPost",
        "peer_type": "GlobalInstance"
      },
      "rpki": {
        "rov": "valid"
      },
      "pathAttributes": [
        {
          "asPath": [
            "AS6447",
            "AS51019",
            "AS34927",
            "AS50673",
            "AS8587"
          ]
        },
        {
          "standardCommunities": [
            "AS34927:930",
            "AS51019:1101"
          ]
```

Getting JSON out of Rotonda: simple filtering

- **/api/v1/ribs/ipv4unicast/routes/145.100.0.0/15**

```
?include=moreSpecifics,   # include, you know, more specifics

&filter[ribType]=inPre          # routes in AdjRibIn pre-policy
&filter[rovStatus]=valid        # && only ROV valid
&filter[originAsn]=1133         # && originated by AS1133
&filter[community]=AS1103:9876 # && and contains this community
```

Getting JSON out of Rotonda: simple filtering

- **/api/v1/ribs/ipv4unicast/routes/145.100.0.0/15**

```
?include=moreSpecifics,   # include, you know, more specifics

&filter[ribType]=inPre          # routes in AdjRibIn pre-policy
&filter[rovStatus]=valid        # && only ROV valid
&filter[originAsn]=1133         # && originated by AS1133
&filter[community]=AS1103:9876 # && and contains this community
```

All filters are boolean AND'd

(More filters, includes and other options are described in the docs)

Filtering JSON responses using Roto

- **`/api/v1/ribs/ipv4unicast/routes`**

```
?include=moreSpecifics,    # on /routes, this is default

&function[roto]=hidden_hop     # use a roto function to filter
```

Custom HTTP endpoint filter function

```
filter hidden_hop(attr: PathAttributes) {
    match attr.otc() {
        Some(otc) -> {
            match attr.aspath() {
                Some(aspath) -> {
                    if not aspath.contains(otc) {
                        accept
                    } else {
                        reject
                    }
                },
                None -> reject,
            }
        }
        None -> { reject }
    }
}
```

Custom HTTP endpoint filter function

```
filter hidden_hop(attr: PathAttributes) {
    match attr.otc() {
        Some(otc) -> {
            match attr.aspath() {
                Some(aspath) -> {
                    if not aspath.contains(otc)
                        accept
                    } else {
                        reject
                    }
                },
                None -> reject,
            }
        }
        None -> { reject }
    }
}
```


```
{
  "nlri": "185.134.65.0/24",
  "routes": [
    {
      "status": "active",
      "ingress": {
        "id": 37,
        "ingress_type": "bgpViaBmp",
        "parent_ingress": 2,
        "remote_addr": "185.1.167.195",
        "remote_asn": 60150,
        "rib_type": "AdjRibIn",
        "peer_rib_type": "inPost",
        "peer_type": "GlobalInstance"
      },
      "rpki": {
        "rov": "notFound"
      },
      "pathAttributes": [
        {
          "asPath": [
            "AS6447",
            "AS60150",
            "AS5405",
            "AS31477"
          ]
        },
        {
          "otc": "AS6777"
        }
      ]
    }
```

Custom HTTP endpoint filter function

```
filter hidden_hop(attr: PathAttributes) {
    match attr.otc() {
        Some(otc) -> {
            match attr.aspath() {
                Some(aspath) -> {
                    if not aspath.contains(otc)
                        accept
                    } else {
                        reject
                    }
                },
                None -> reject,
            }
        }
        None -> { reject }
    }
}
```


```
{
  "nlri": "185.134.65.0/24",
  "routes": [
    {
      "status": "active",
      "ingress": {
        "id": 37,
        "ingress_type": "bgpViaBmp",
        "parent_ingress": 2,
        "remote_addr": "185.1.167.195",
        "remote_asn": 60150,
        "rib_type": "AdjRibIn",
        "peer_rib_type": "inPost",
        "peer_type": "GlobalInstance"
      },
      "rpki": {
        "rov": "notFound"
      },
      "pathAttributes": [
        {
          "asPath": [
            "AS6447",
            "AS60150",
            "AS5405",
            "AS31477"
          ]
        },
        {
          "otc": "AS6777"
        }
      ]
    }
```

/metrics

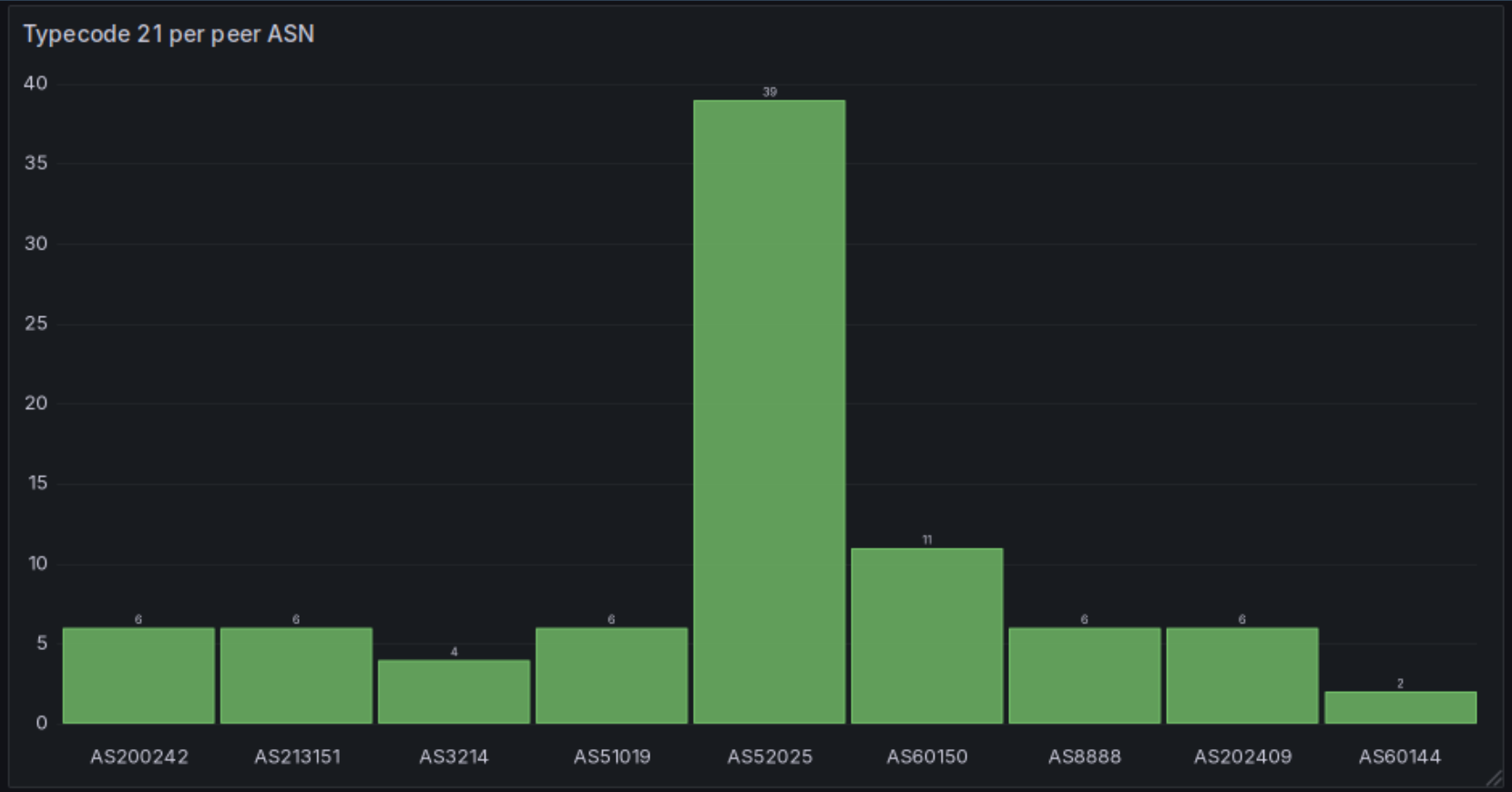
- Prometheus style metrics, enabling for monitoring and alerting
- Built-in Rotonda-related metrics

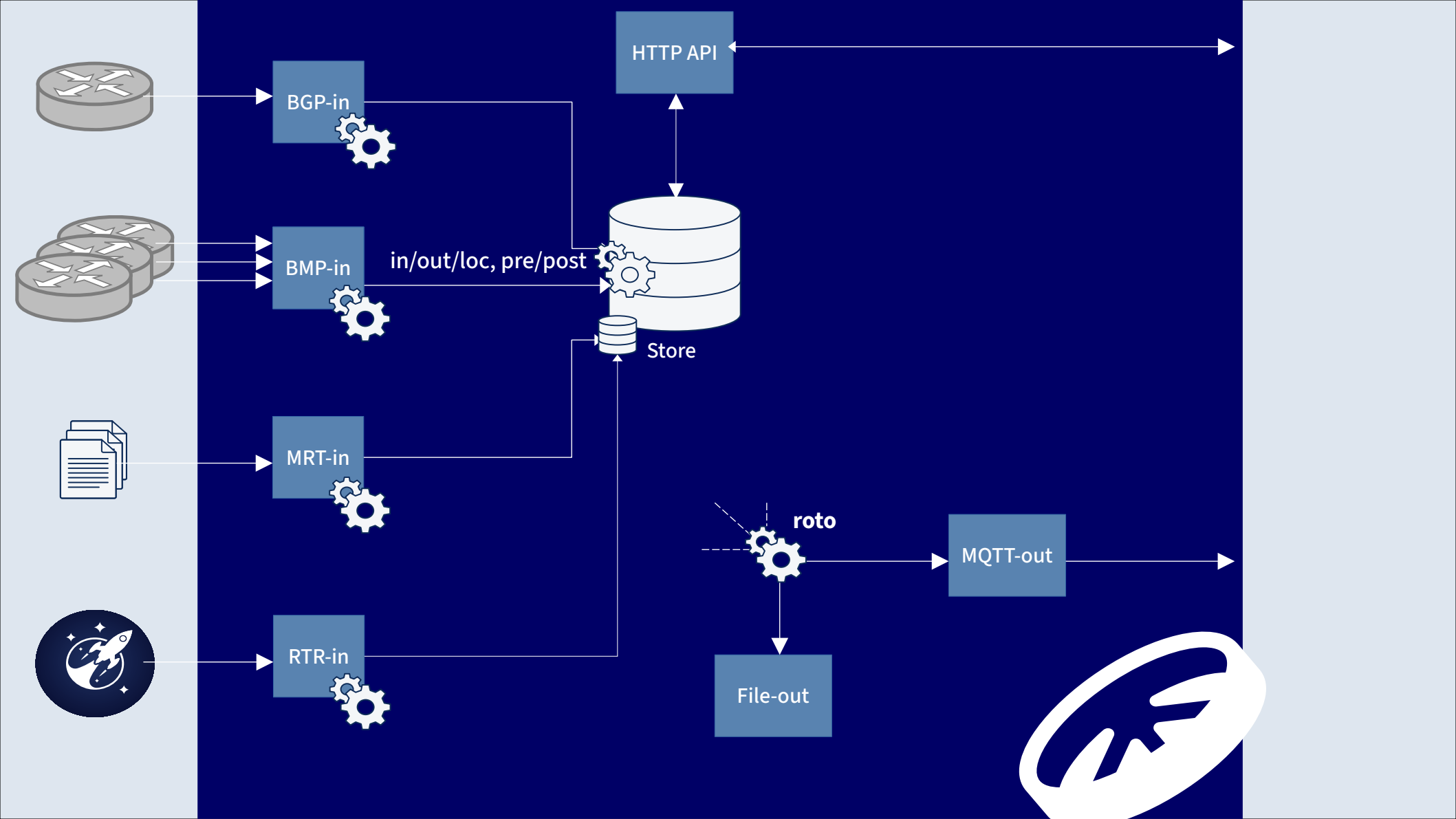
- and of course ...

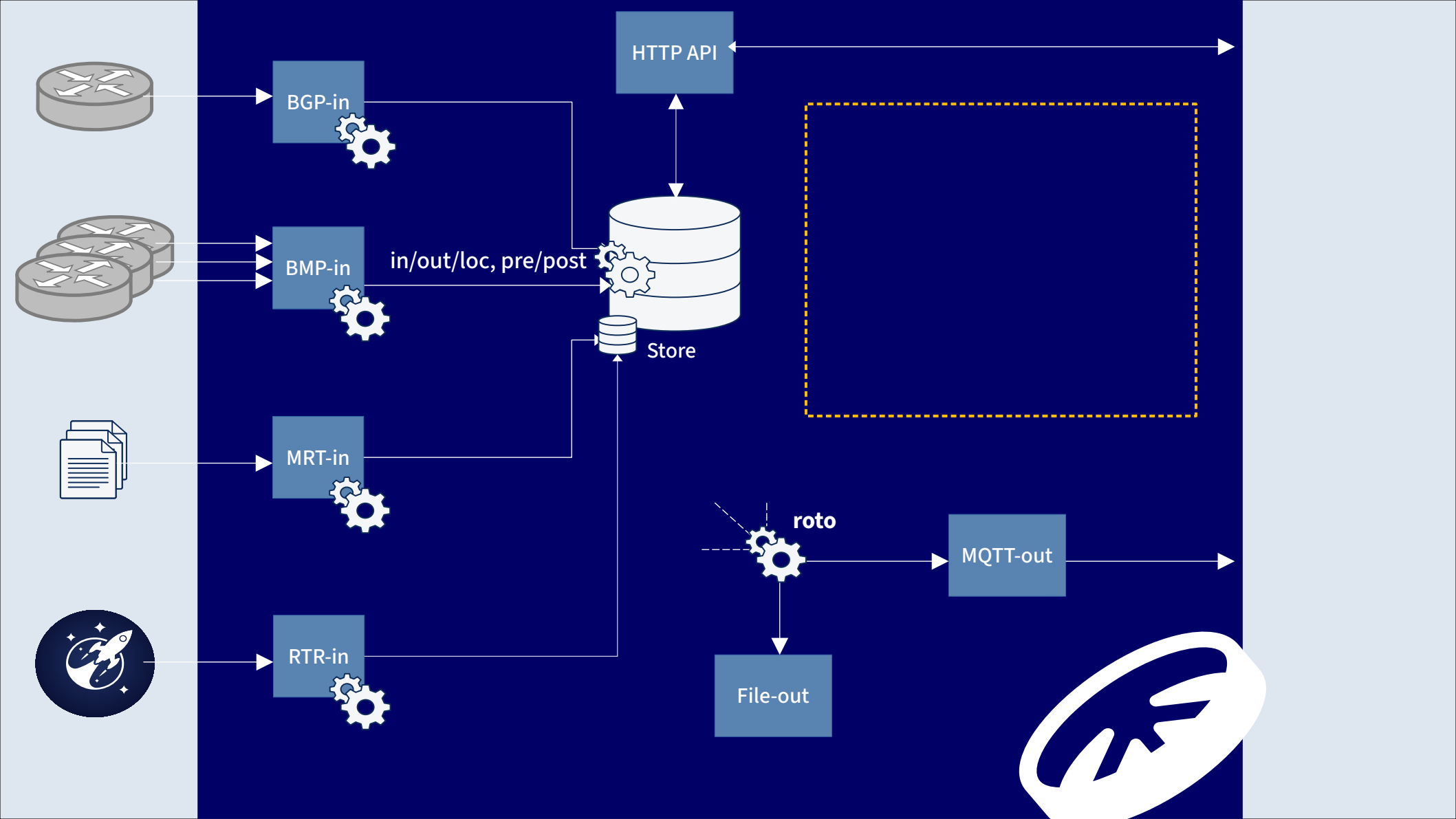
User-defined metrics via Roto!

```
filter bmp_in(
    bmp_msg: BmpMsg,
    ingress_info: IngressInfo,
) {
    # Keep track of a specific deprecated (and possibly problematic) attribute.
    # This will show up in the /metrics endpoint as
    #
    #   'roto_user_defined_unwanted_attribute'
    #
    if bmp_msg.has_attribute(21) {
        metrics.increase_counter(
            f"unwanted_attribute{{peer_asn=\"{ingress_info.peer_asn()}\"}}",
            1
        );
    }

    accept
}
```

Hooked up to Grafana, ready to alert




HTTP API
BGP-in
BMP-in
in/out/loc, pre/post
Store
MRT-in
roto
MQTT-out
RTR-in
File-out


HTTP API
BGP-in
BMP-in
in/out/loc, pre/post
Store
MRT-in
RTR-in
roto
MQTT-out
File-out

Concluding, Rotonda today

- Ingest and store high volumes of routing info in memory
- IPv6/IPv4 unicast (and multicast)
- Interaction via HTTP JSON API

Rotonda $tomorrow

- **Non-unicast address families**
  - e.g. EVPN, FlowSpec, (BGP-LS?)
  - ADDPATH
- **Expose ASPA to Roto**
- **Persistent storage to disk**
- **Interaction via CLI**
- **YANG-based configuration**

Give it a go now

- Freely available open source software
- Take a .deb, .rpm or docker image, or compile using cargo
- https://rotonda.docs.nlnetlabs.nl
- https://github.com/NLnetLabs/rotonda/

and help us out!

- **We are not operators**
  - we don’t run a network
  - we don’t have any routing hardware
- **Publicly available non-unicast (test)data is hard to come by**
  - .pcaps welcome!
- **BMP export implementations all have their surprises**
  - Moreover, BMP as a protocol is still evolving. Any input/data is useful for us.

On protocol evolvement

- draft-lucente-grow-bmp-offline (“BMP Snapshots”)
- draft-petrie-grow-mrt-bmp

Collecting and analyzing routing information with ROTONDA

RIPE91 Bucharest, Romania

{luuk,jasper}@nlnetlabs.nl

NLNETLABS