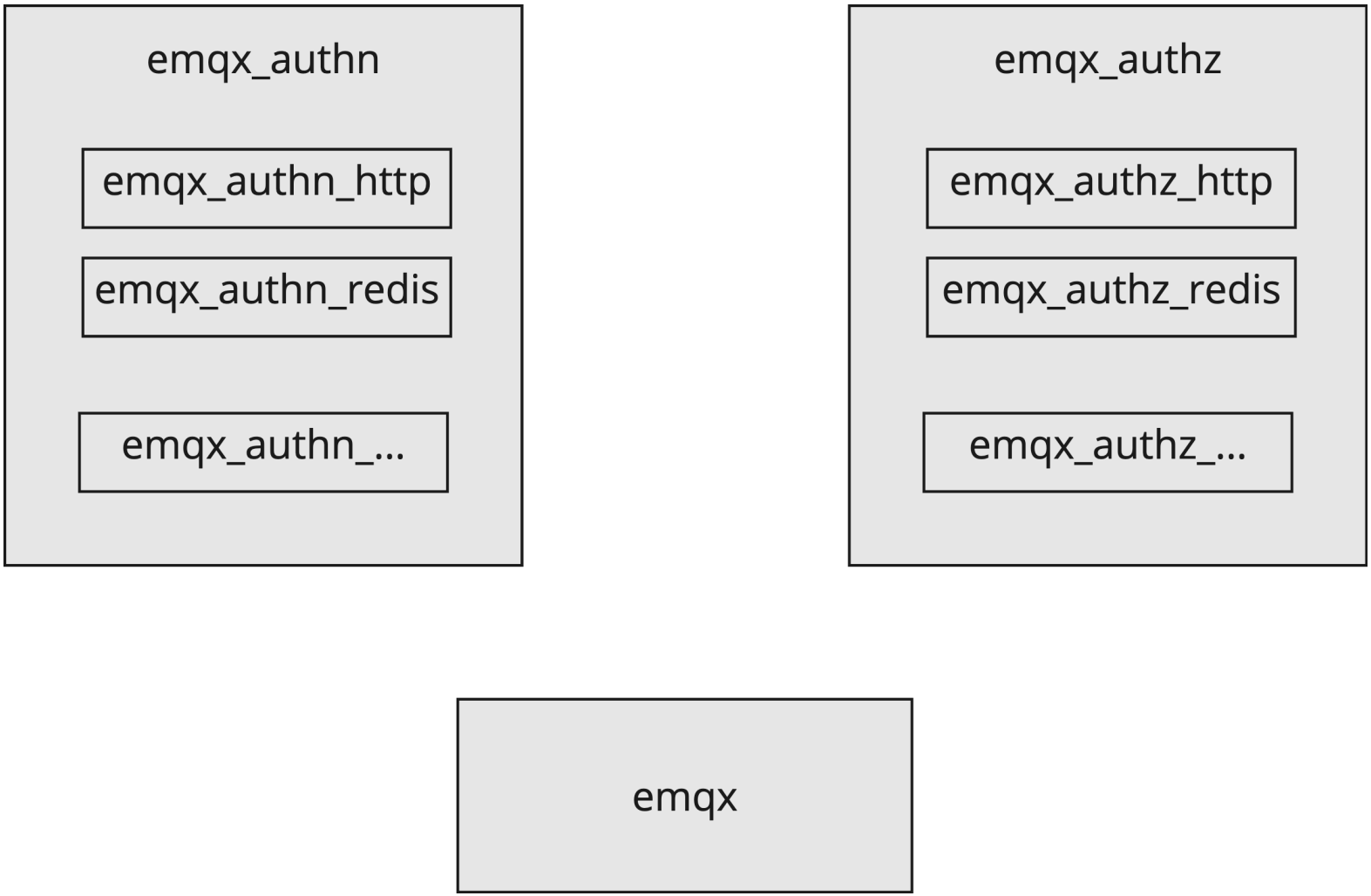
Auth Application Refactoring, Stage 2

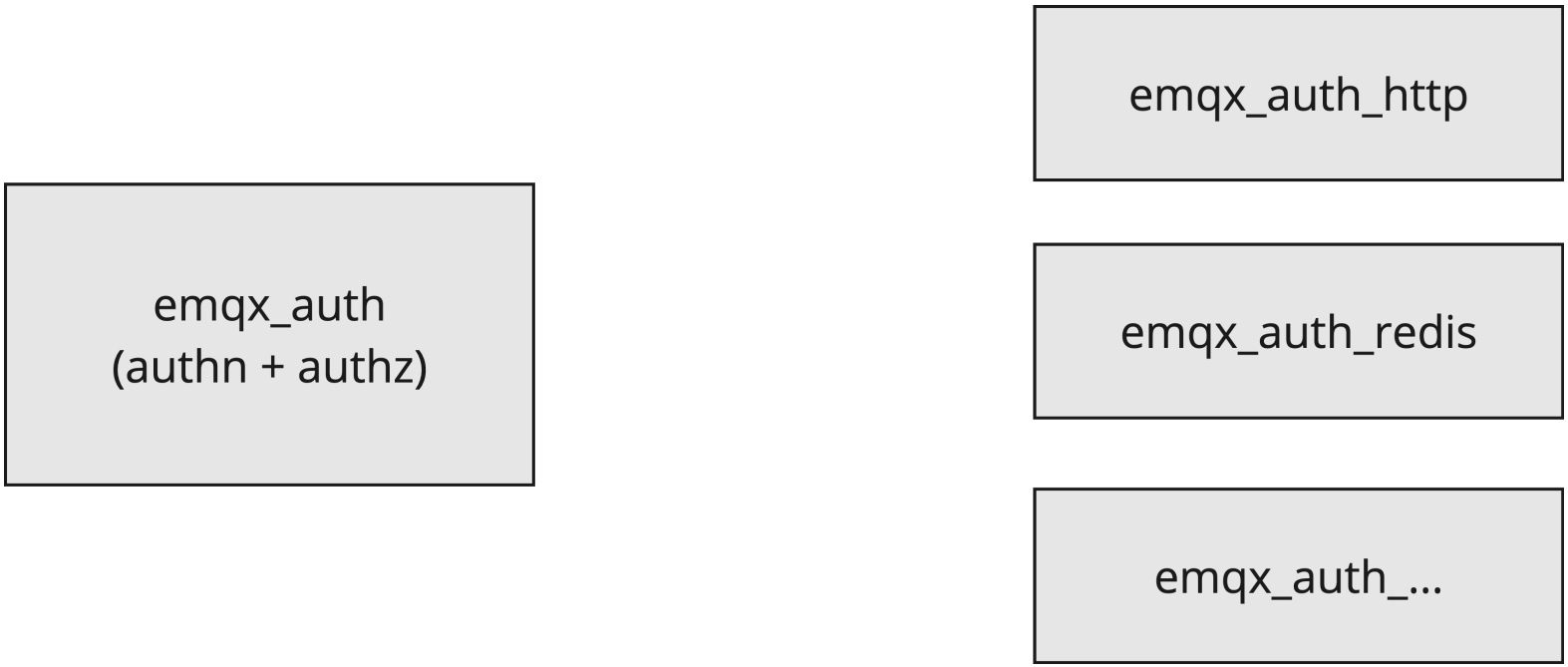
Ilya Averyanov

2023

Current State



Desired State

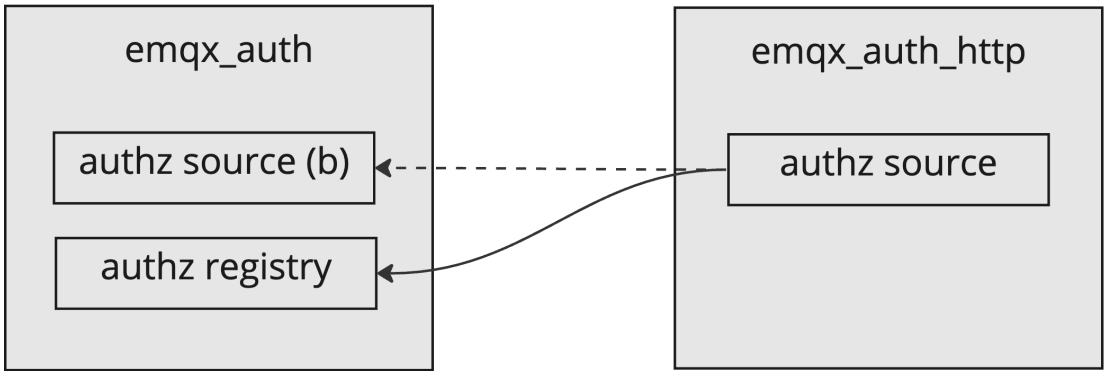


Auth Refactoring
Purposes

- Better separation of concerns
- Better application infrastructure (less dependencies, easier hot code reloading, less cumbersome CI)
- Better extensibility and pluggability

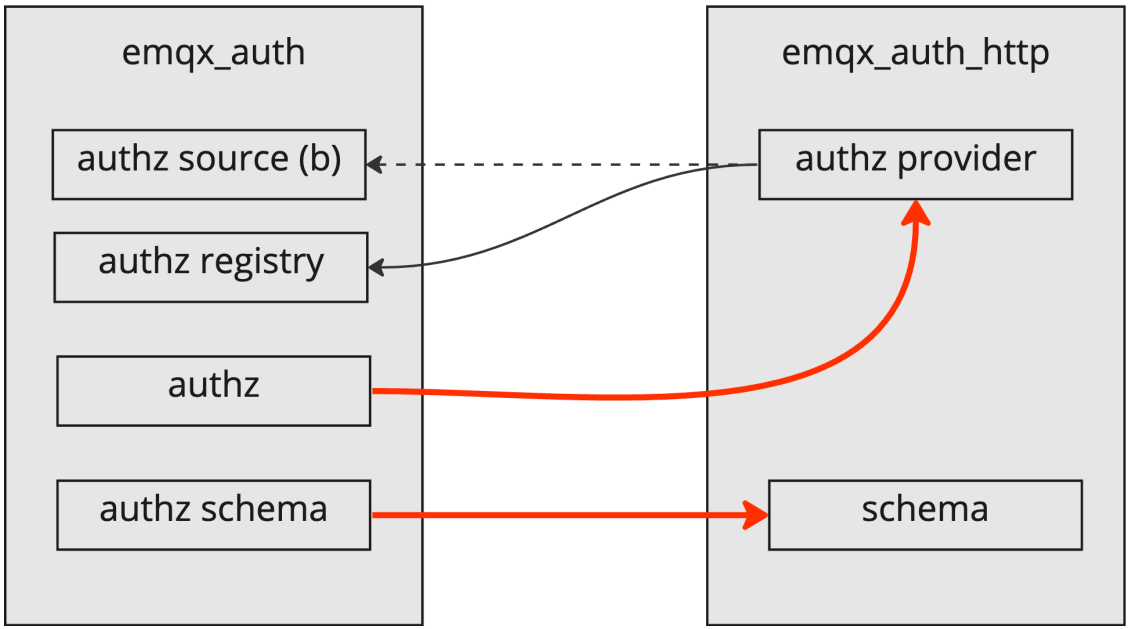
Auth Refactoring

Naive



Auth Refactoring

Actual



Auth Refactoring
Challenges

- Leaking (ad hoc) transformations of authn provider/authz source data
- Opposite directions of dependencies in schema layer and in provider/source layer
- Delayed initialization
- Lack of top level "profile" IoC container

Auth Refactoring
Ad Hoc Transformations

We extend authn/authz behaviours to make transformations opaque

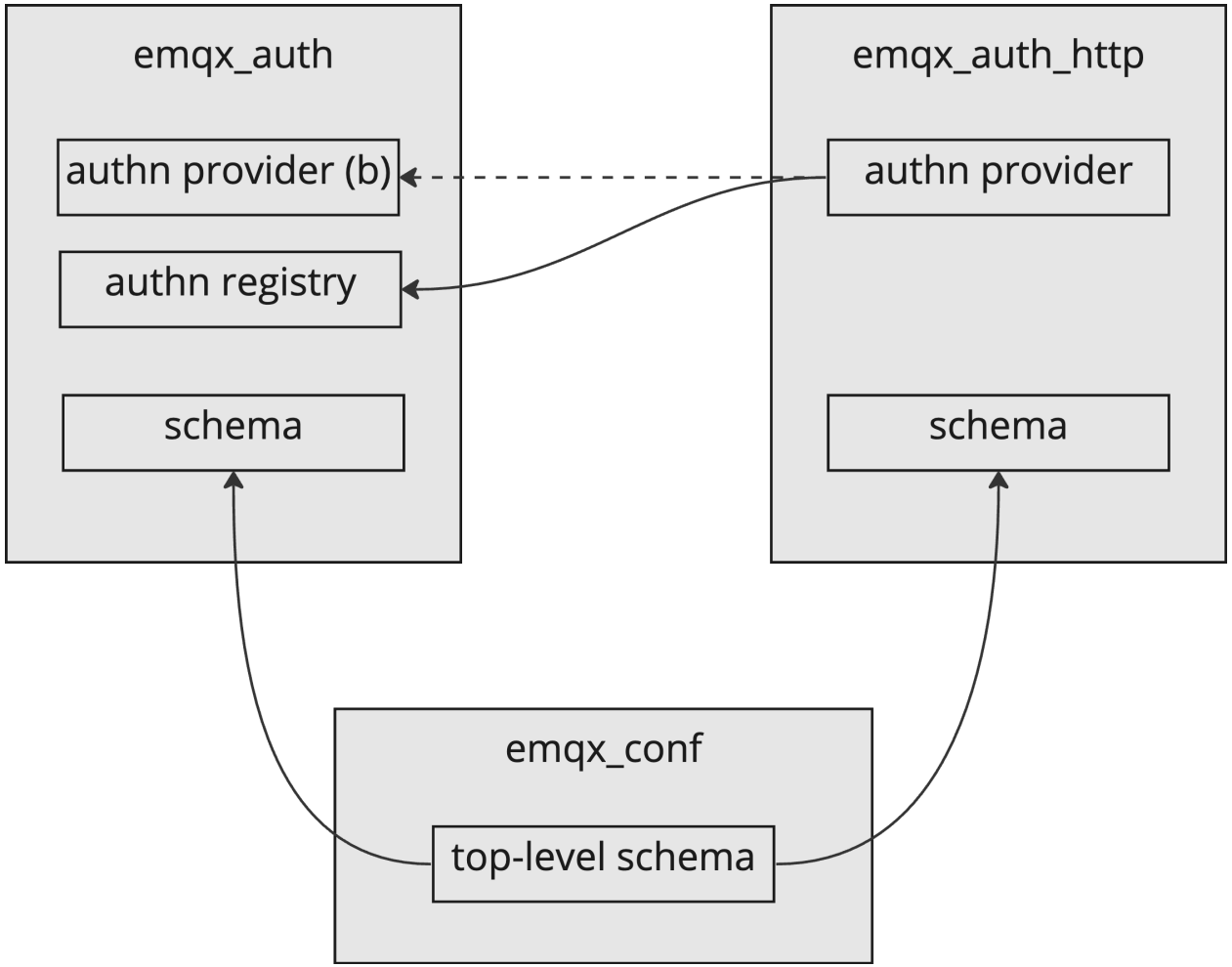
```
%% Some new authz source callbacks

-callback write_files(raw_source()) ->
    raw_source() | no_return().
-callback read_files(raw_source()) ->
    raw_source() | no_return().
-callback merge_defaults(raw_source()) ->
    raw_source().
```

Auth Refactoring
Delayed Initialization

- Authn/authz install forbidding hooks on start.
- Authn/authz do not instantiate providers/sources on start if there are any nontrivial ones.
- On source/privider registration we check if all configured providers/sources are available.
- If yes, we instantiate them and install actual checking hooks.

Auth Refactoring
Circular Dependencies



Auth Refactoring
Circular Dependencies

- We have inevitable circular dependencies only in schema.
- We use `emqx_conf` (`emqx_conf_schema`) as the plumbing layer (actually, it already is).
- `emqx_conf` identifies actual provider/source schema modules.
- `emqx_conf` passes them to the full authn/authz schema modules.
- `emqx_conf` injects the resulting schema into ‘emqx‘ app.

Auth Refactoring

Custom Auth: With Schema Validation

- Implement an application that implements auth and auth schema behaviours, similar to the existing apps.
- Also integrate this into the build configs (`reboot_lists.eterm`, `mix.exs`, etc.).

Auth Refactoring

Custom Auth: With Loose Schema Validation

- **We** implement generic schema behaviours, which allows to provide an arbitrary option map as (a part of) config (TBD).
- **Customer** implements a full application or just a **plugin** that implements only source/provider behaviours.

Auth Refactoring
Missing features

- Further **local** unifications — they are intentionally left for later in order not to make the PR too big and containing mixed logic.
- Improving READMEs, this is also can be done continously.

Thank you!