Cilium + eBPF Day EUROPE

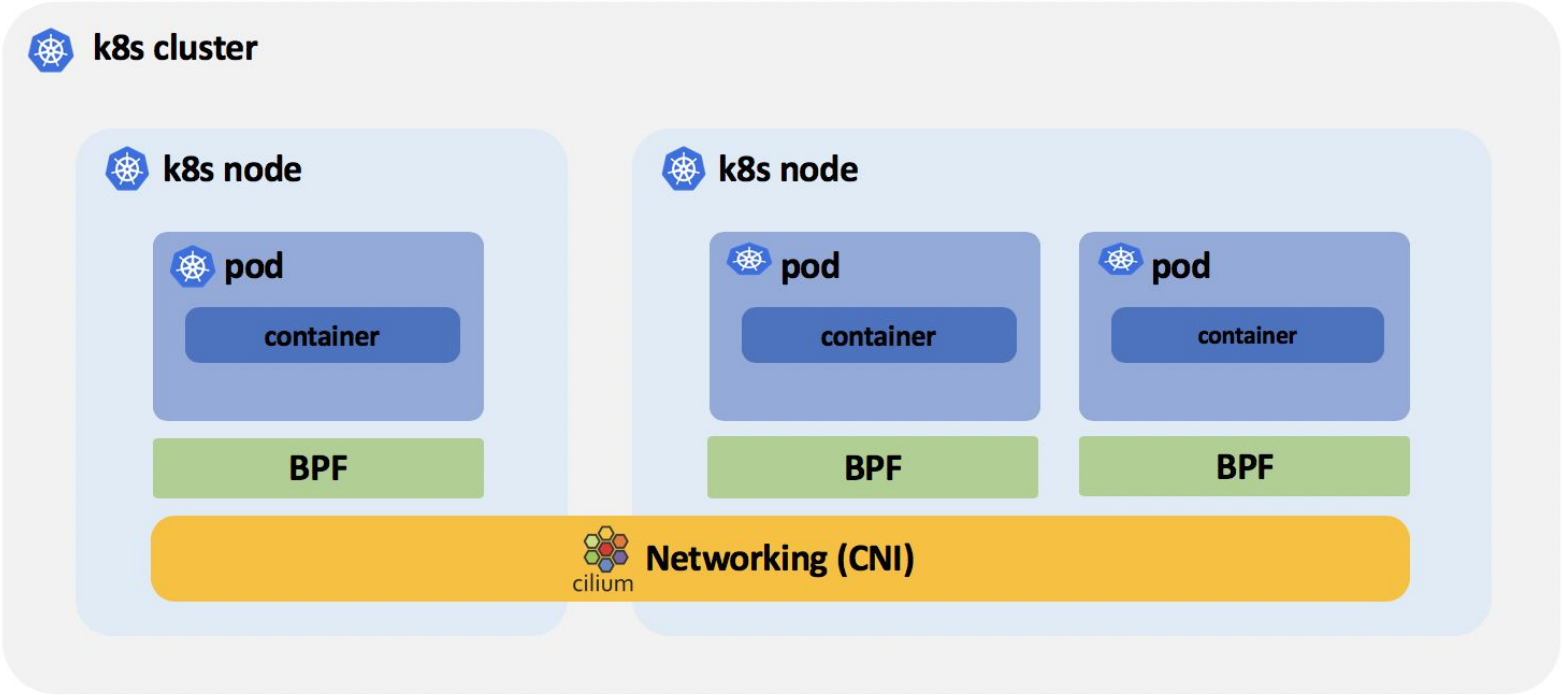
Deep Dive Into Cilium Resilient Architecture

Jussi Mäki

Martynas Pumputis

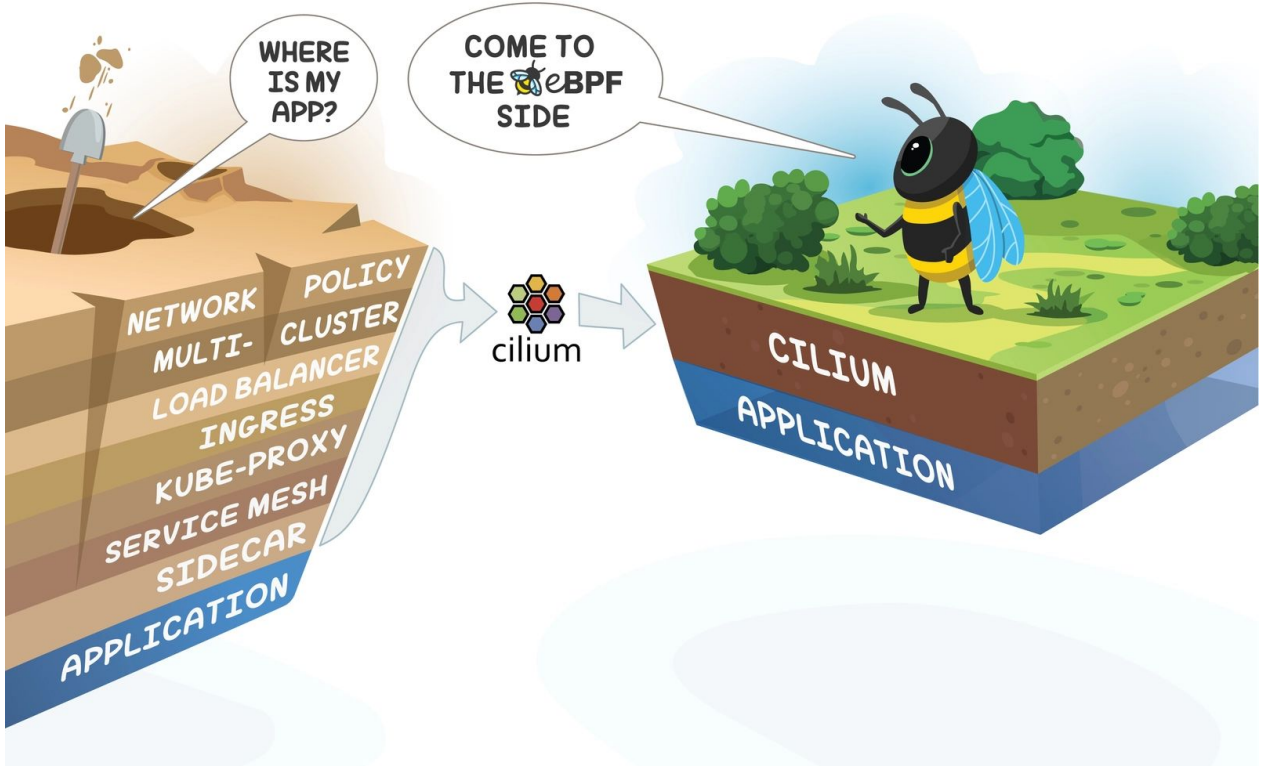
ISOVALENT

Cilium CNI

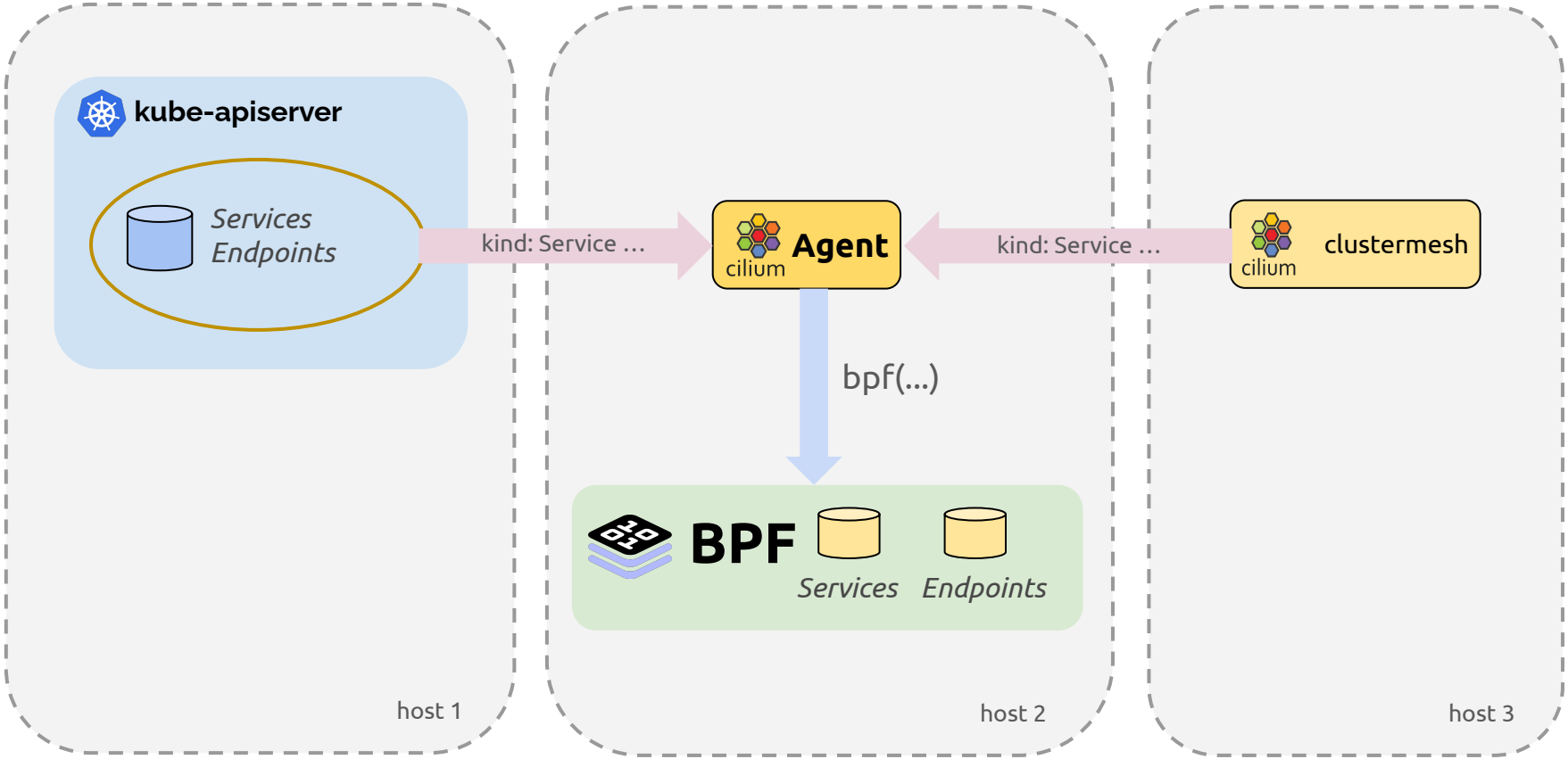

k8s cluster
k8s node
pod
container
BPF
k8s node
pod
container
BPF
pod
container
BPF
cilium
Networking (CNI)

… which liberated K8s from iptables

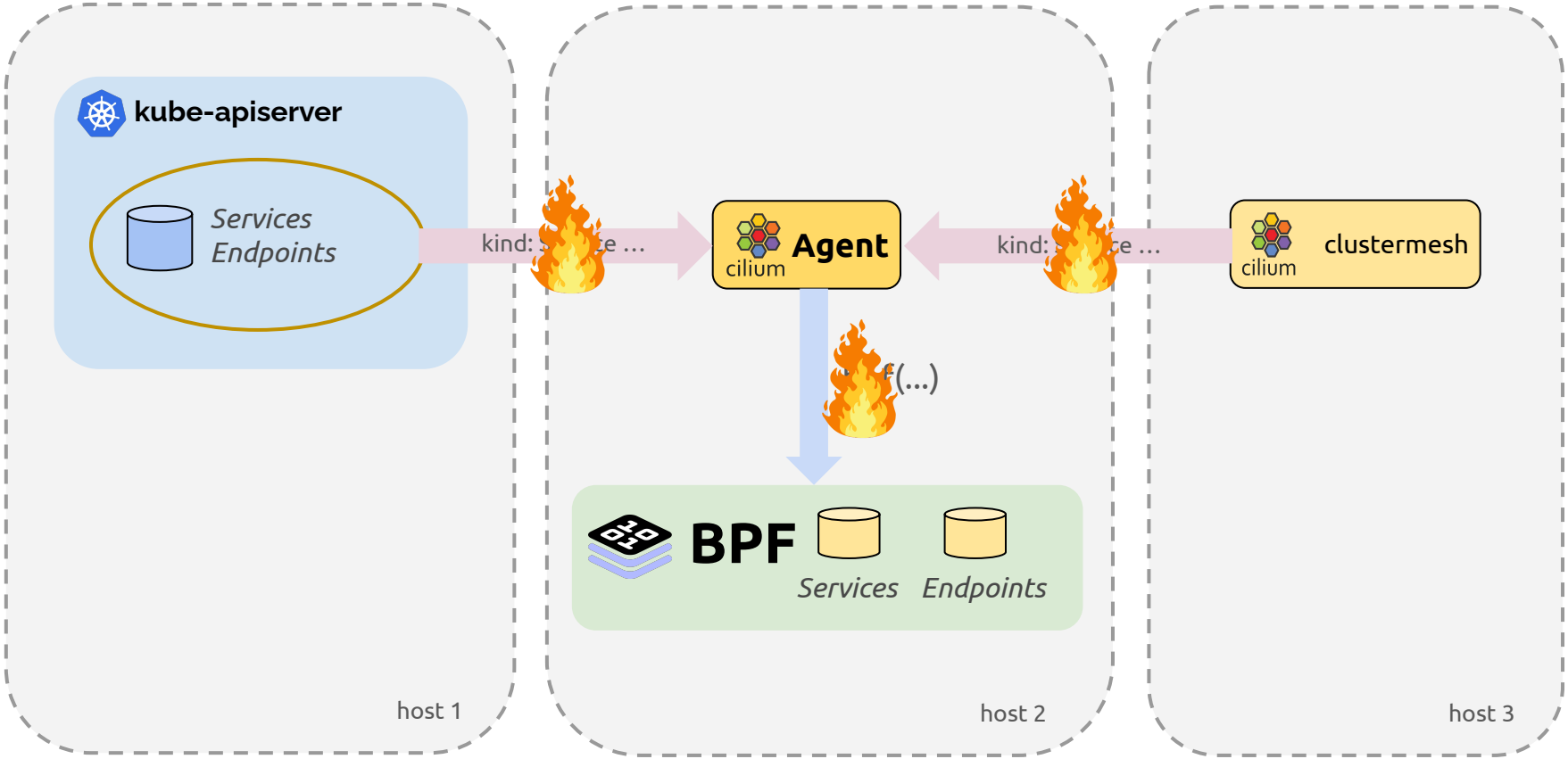
… and solved most of the K8s networking problems



Cilium under the hood

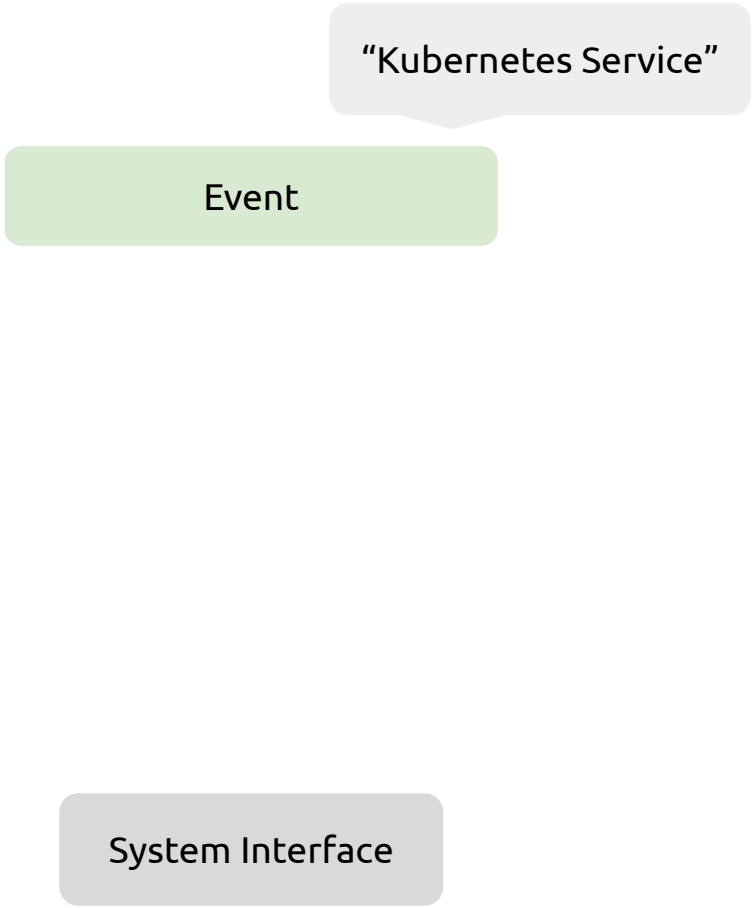


Cilium under the hood

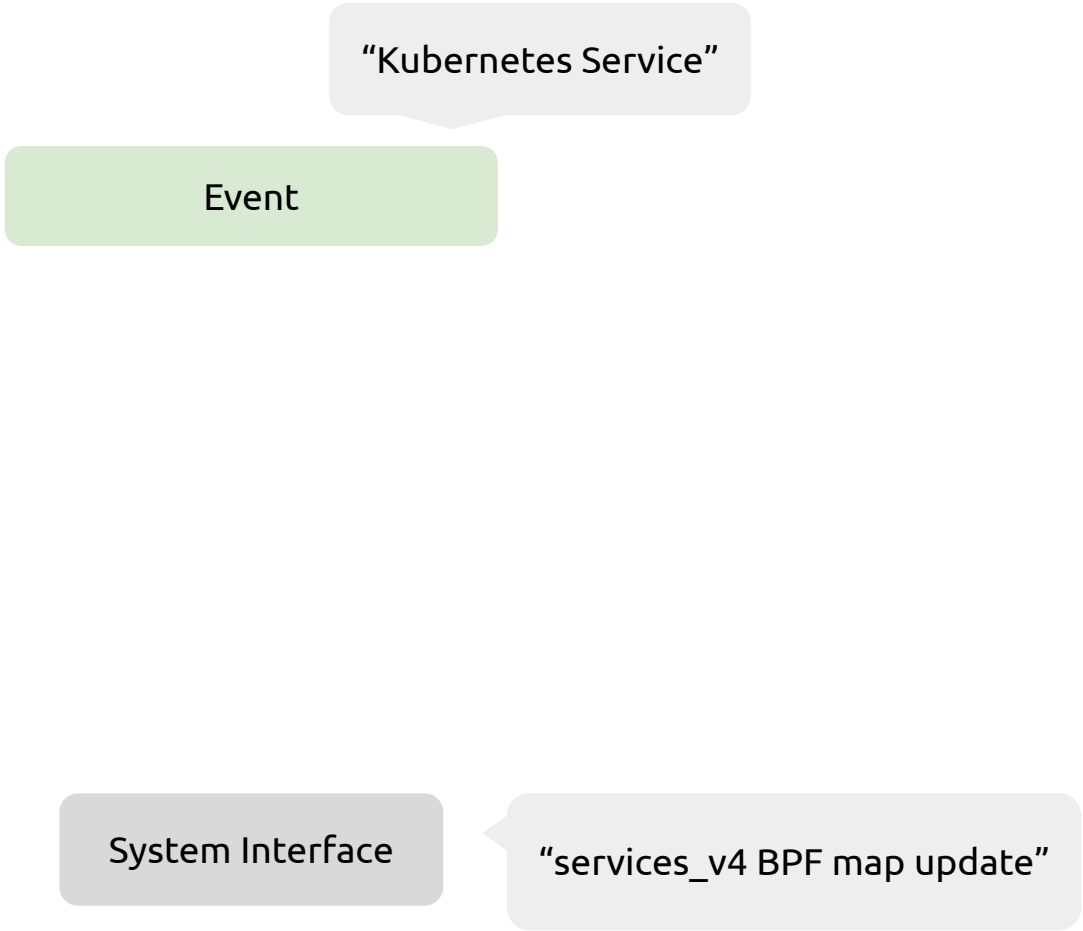


Reconciliation

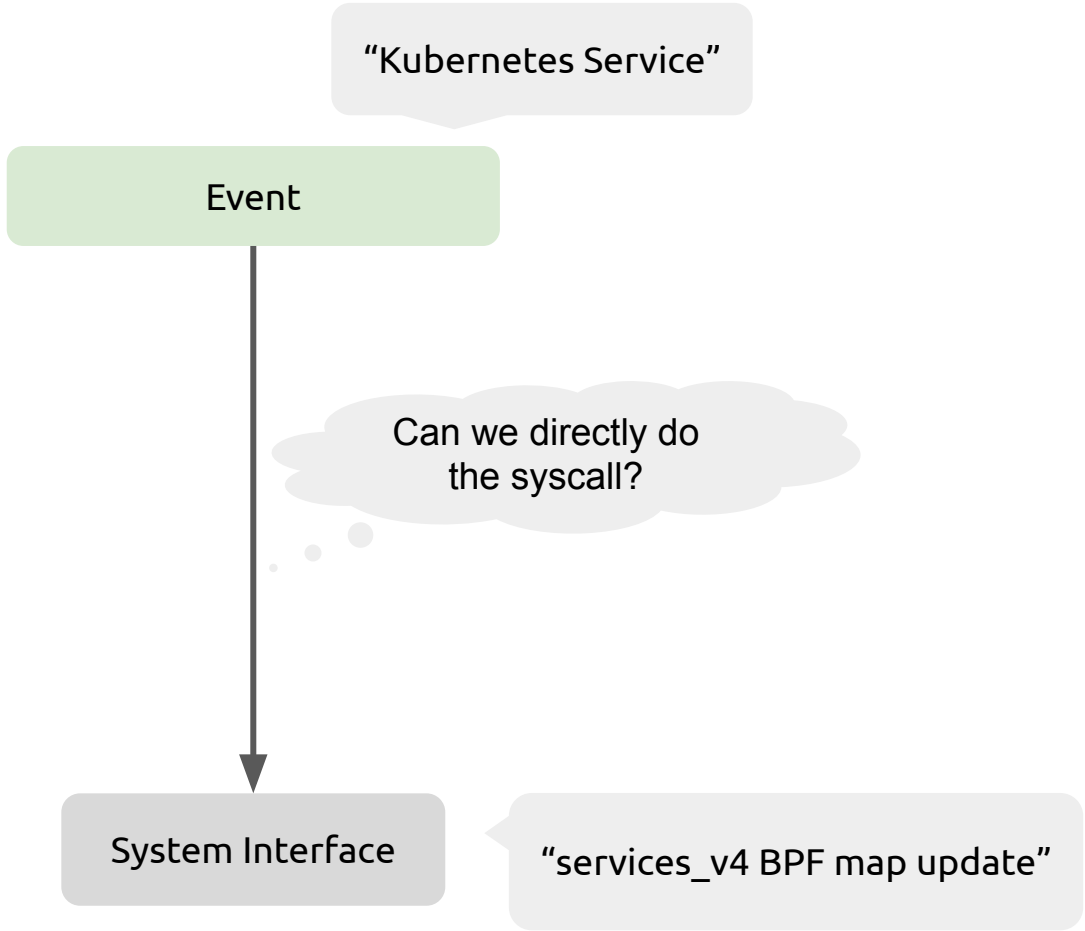
Reconciliation



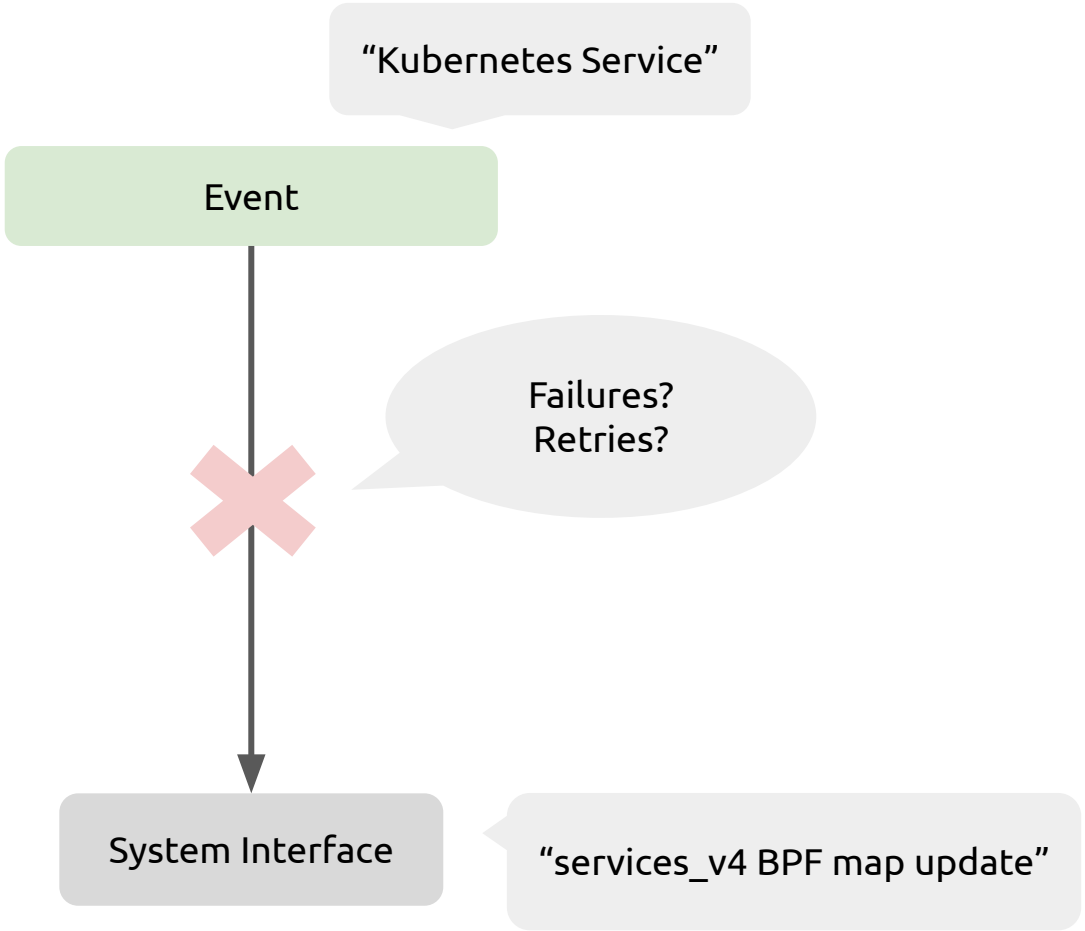
Reconciliation



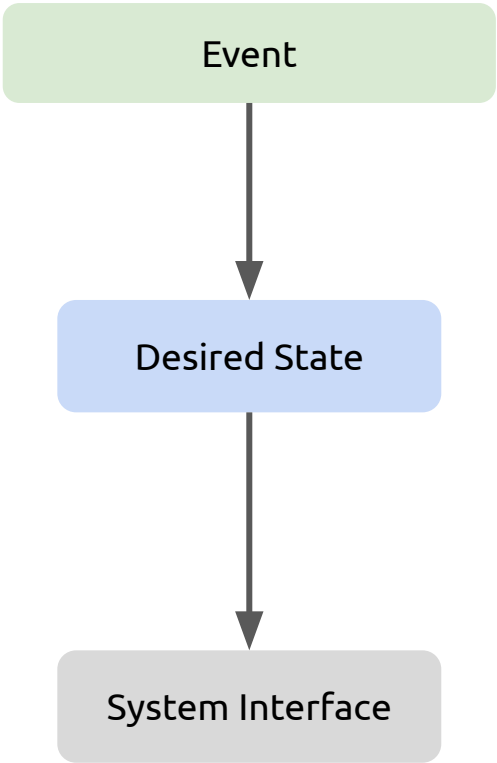
Reconciliation



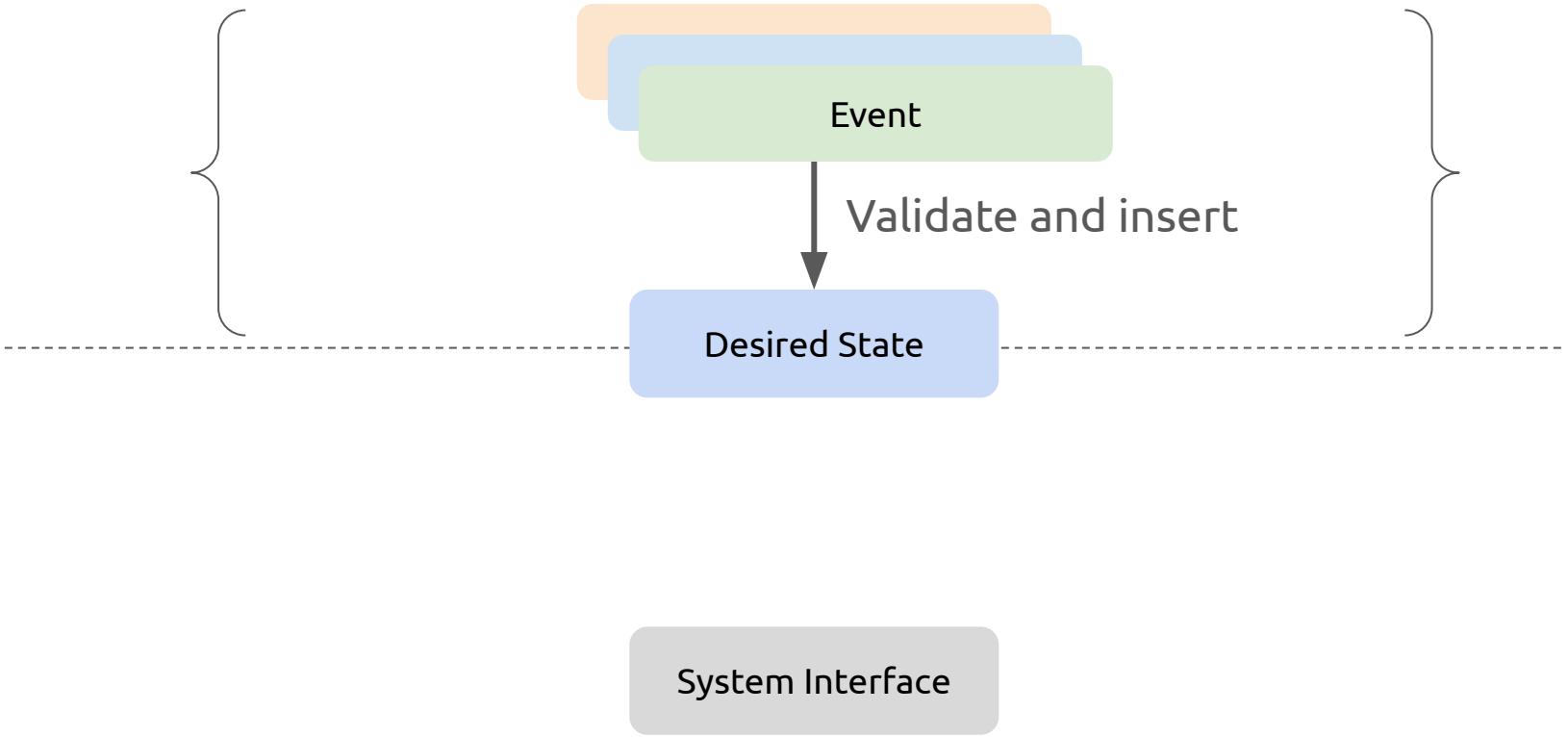
Reconciliation



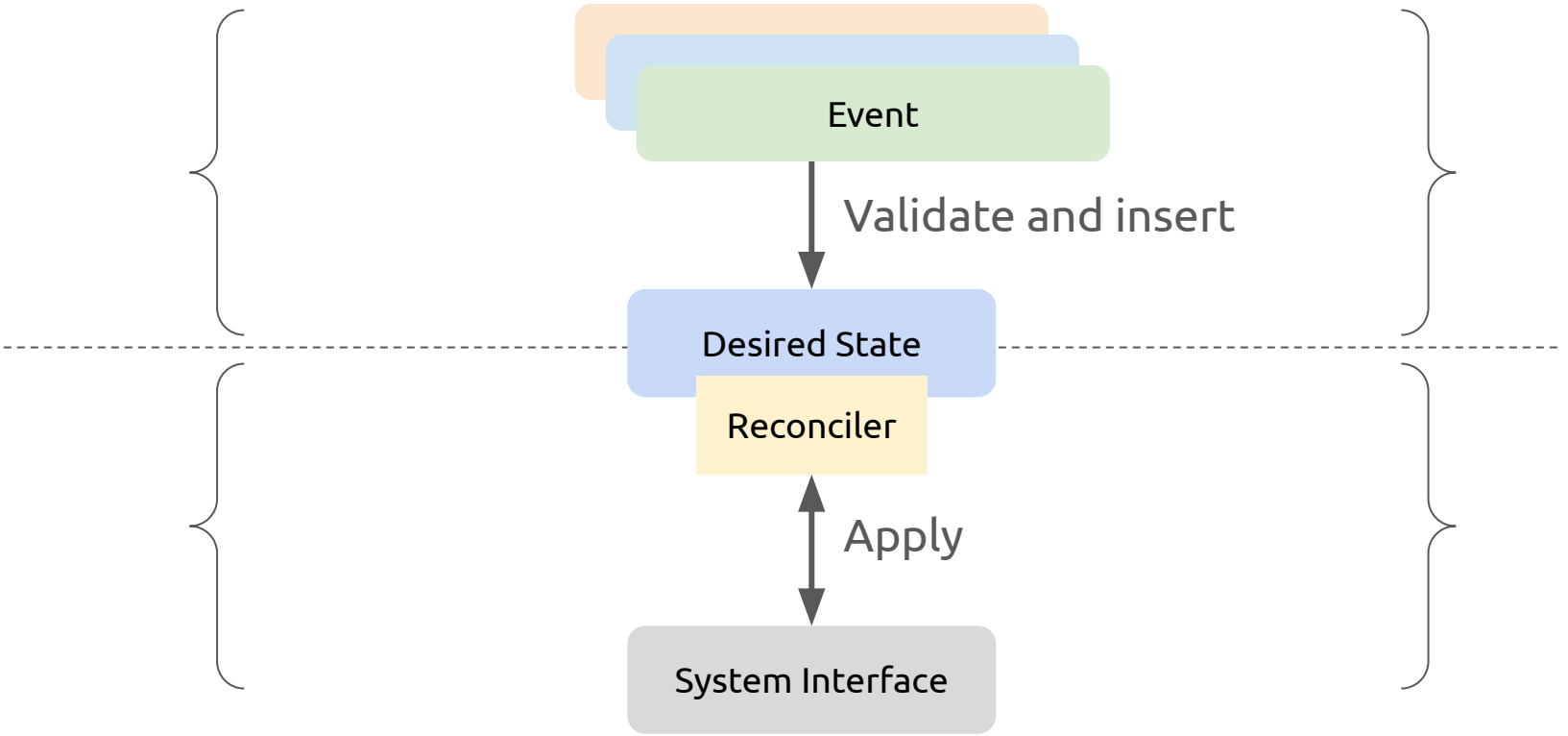
Reconciliation



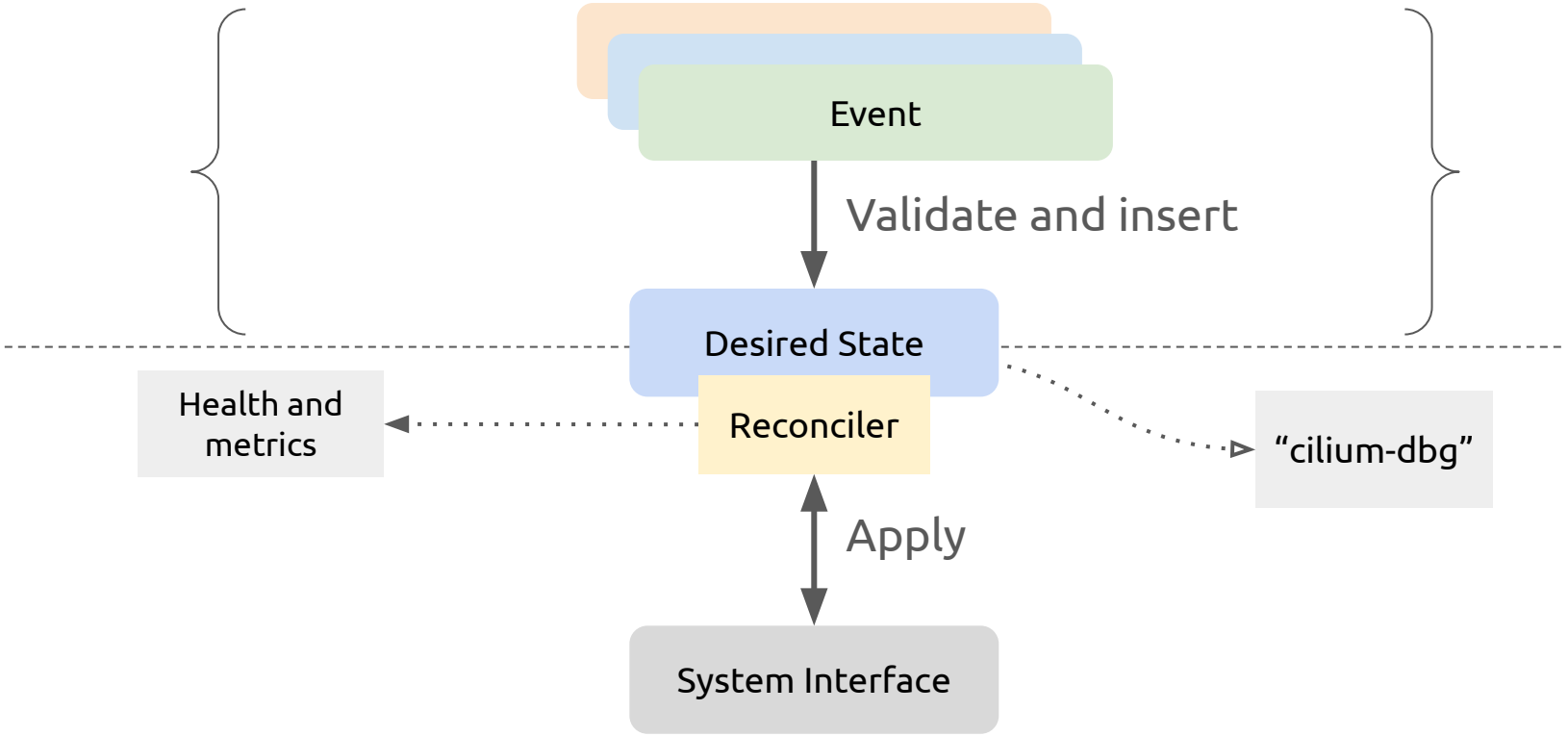
Reconciliation



Reconciliation



Reconciliation



StateDB: in-memory database for state

- In-memory transactional database

StateDB: in-memory database for state

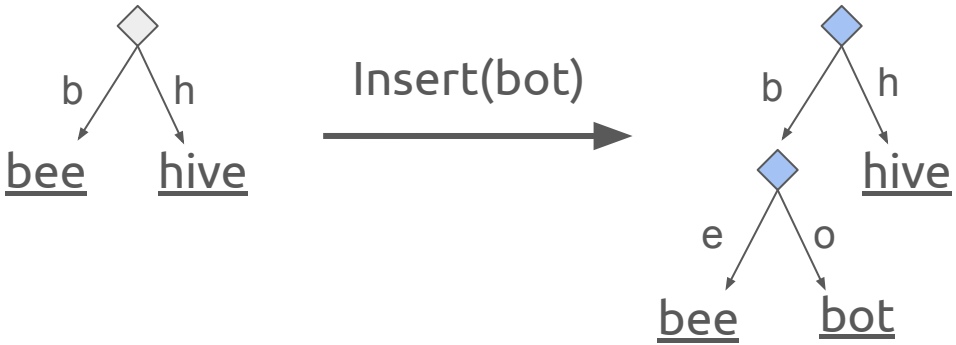
- In-memory transactional database
- **Immutable** revisioned objects indexed in immutable radix trees

StateDB: in-memory database for state

- In-memory transactional database
- **Immutable** revisioned objects indexed in immutable radix trees
- Channel-based change notification mechanism

StateDB: in-memory database for state

- In-memory transactional database
- **Immutable** revisioned objects indexed in immutable radix trees
- Channel-based change notification mechanism



StateDB: Creating a table

```
type Example struct { ID string }
var IDIndex = statedb.Index[*Example, string]{
    Name: “id”,
    FromObject: func(e *Example) index.KeySet {
        return index.NewKeySet(index.String(e.ID))
    },
    FromKey: index.String,
    Unique: true,
}
var MyExampleTable = statedb.NewTable(“examples”, IDIndex)
```

StateDB: Creating a table

```
type Example struct { ID string }
var IDIndex = statedb.Index[*Example, string]{
    Name: “id”,
    FromObject: func(e *Example) index.KeySet {
        return index.NewKeySet(index.String(e.ID))
    },
    FromKey: index.String,
    Unique: true,
}
var MyExampleTable = statedb.NewTable(“examples”, IDIndex)
```

StateDB: Creating a table

```
type Example struct { ID string }
var IDIndex = statedb.Index[*Example, string]{
    Name: “id”,
    FromObject: func(e *Example) index.KeySet {
        return index.NewKeySet(index.String(e.ID))
    },
    FromKey: index.String,
    Unique: true,
}
var MyExampleTable = statedb.NewTable(“examples”, IDIndex)
```

StateDB: Creating a table

```
type Example struct { ID string }
var IDIndex = statedb.Index[*Example, string]{
    Name: “id”,
    FromObject: func(e *Example) index.KeySet {
        return index.NewKeySet(index.String(e.ID))
    },
    FromKey: index.String,
    Unique: true,
}
var MyExampleTable = statedb.NewTable(“examples”, IDIndex)
```

StateDB: Querying and watching

```
type Service struct { Name, Namespace string; … }
var Services statedb.Table[*Service]
var NamespaceIndex statedb.Index[*Service, string]
var db *statedb.DB

txn := db.ReadTxn()
iter, watch := Services.Get(txn, NamespaceIndex.Query(“default”))
for svc, rev, ok := iter.Next(); ok; svc, rev, ok = iter.Next() {
  …
}
<-watch
```

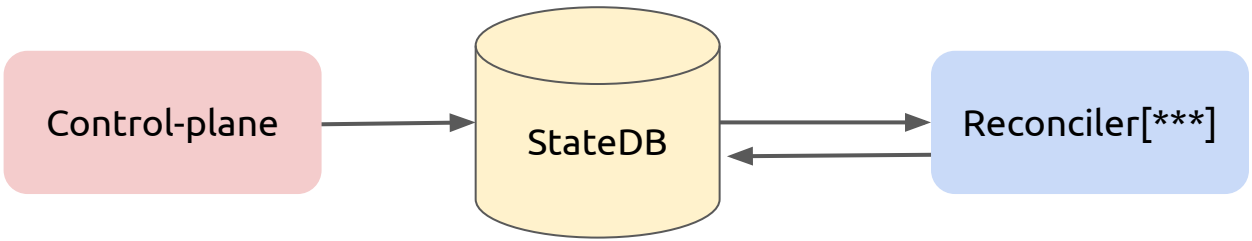
StateDB: Querying and watching

```
type Service struct { Name, Namespace string; … }
var Services statedb.Table[*Service]
var NamespaceIndex statedb.Index[*Service, string]
var db *statedb.DB

txn := db.ReadTxn()
iter, watch := Services.Get(txn, NamespaceIndex.Query(“default”))
for svc, rev, ok := iter.Next(); ok; svc, rev, ok = iter.Next() {
  …
}
<-watch
```

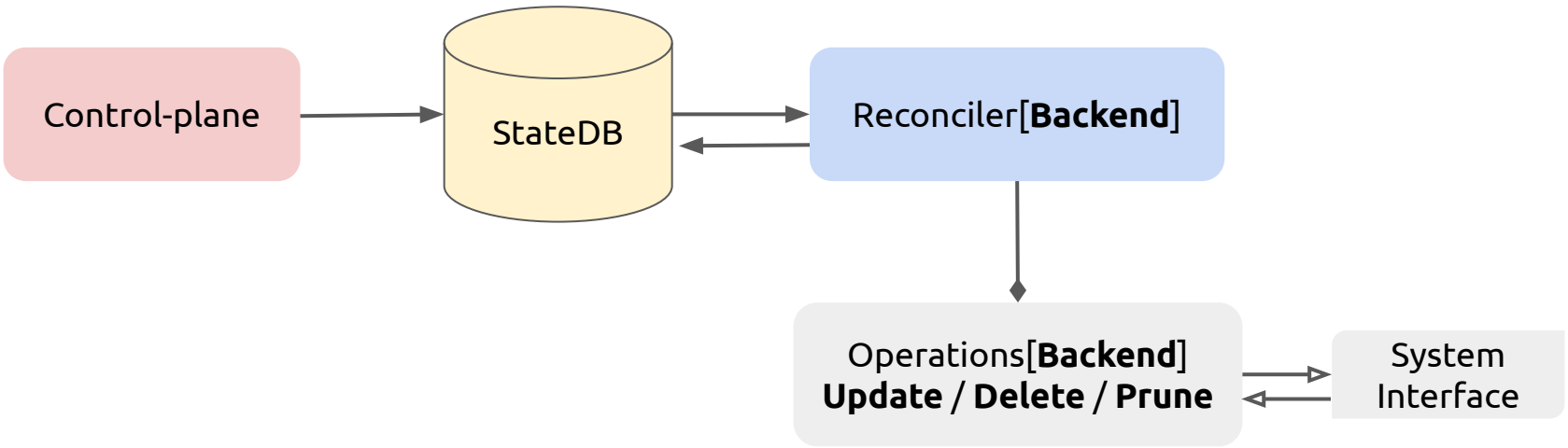
StateDB: Querying and watching

```
type Service struct { Name, Namespace string; … }
var Services statedb.Table[*Service]
var NamespaceIndex statedb.Index[*Service, string]
var db *statedb.DB

txn := db.ReadTxn()
iter, watch := Services.Get(txn, NamespaceIndex.Query(“default”))
for svc, rev, ok := iter.Next(); ok; svc, rev, ok = iter.Next() {
  …
}
<-watch // chan struct{}
```

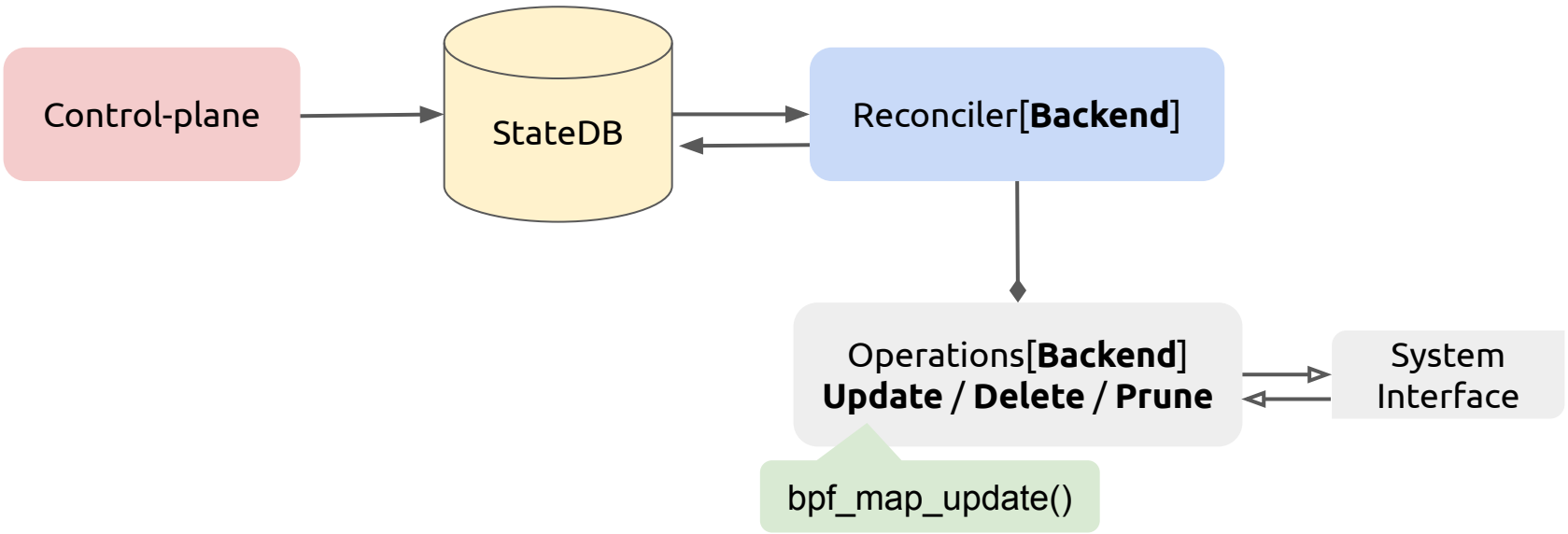
StateDB Reconciler



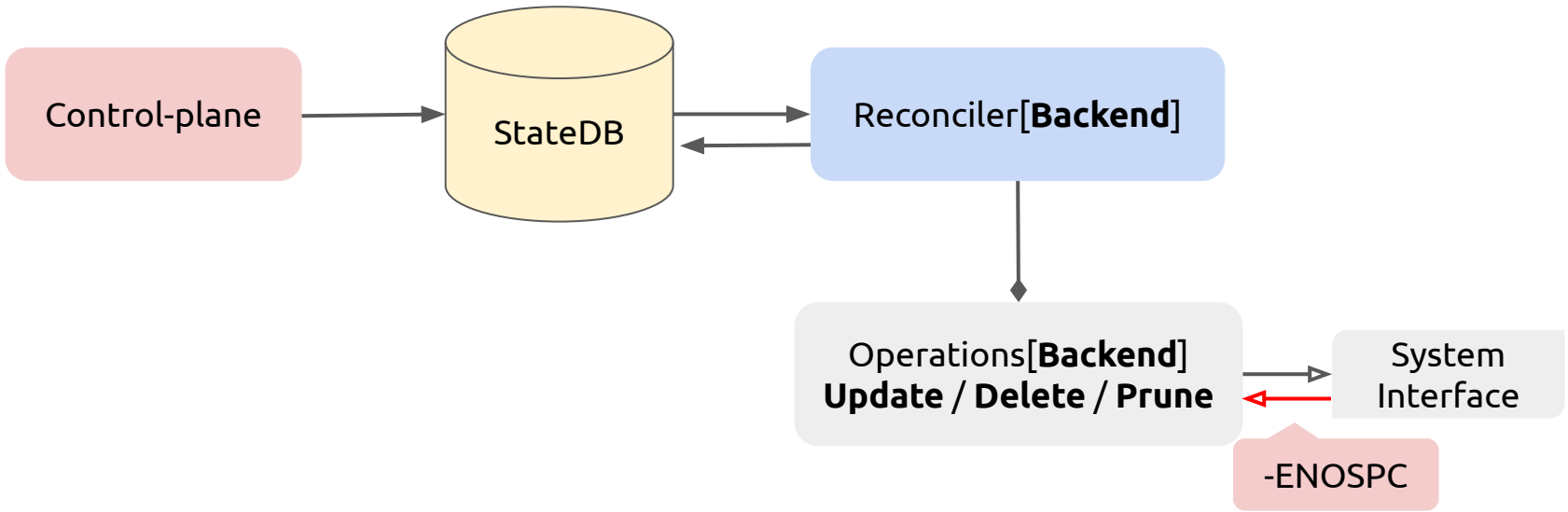
StateDB Reconciler



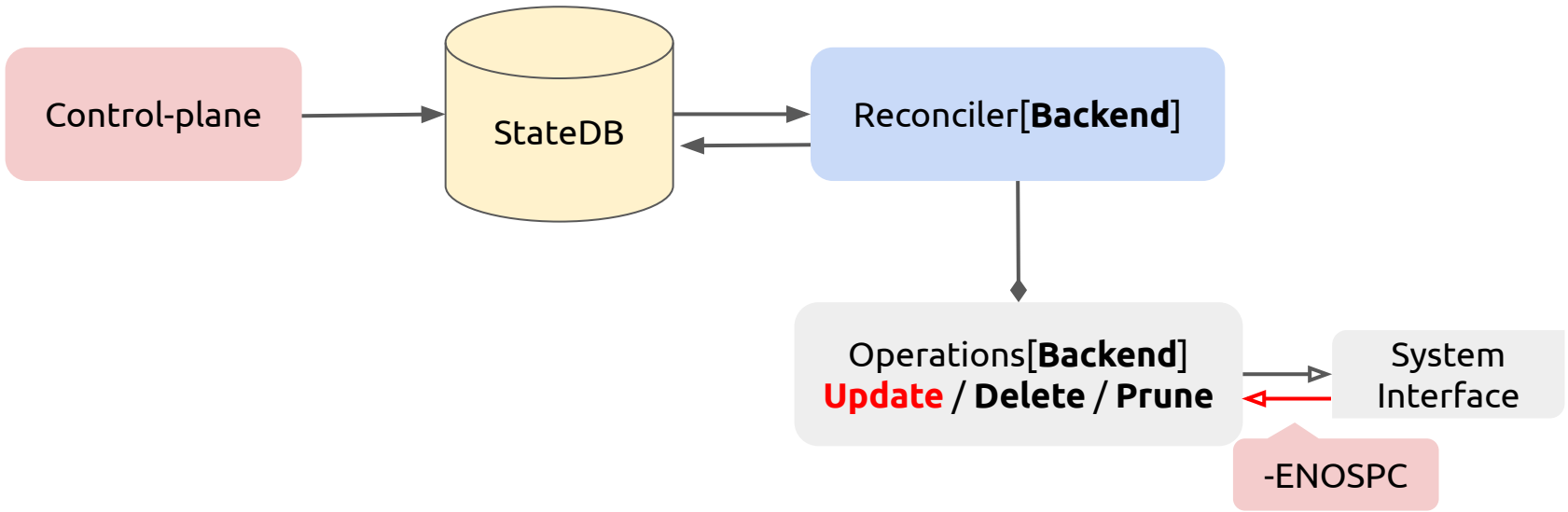
StateDB Reconciler



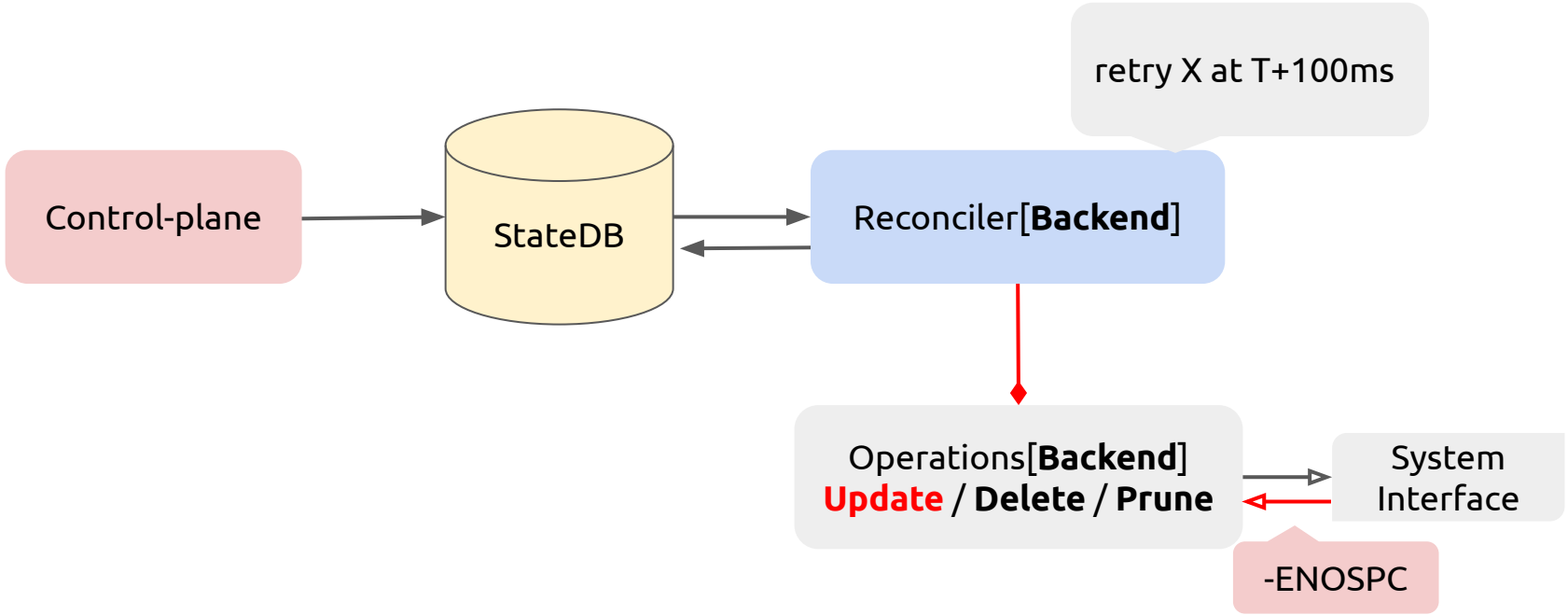
StateDB Reconciler



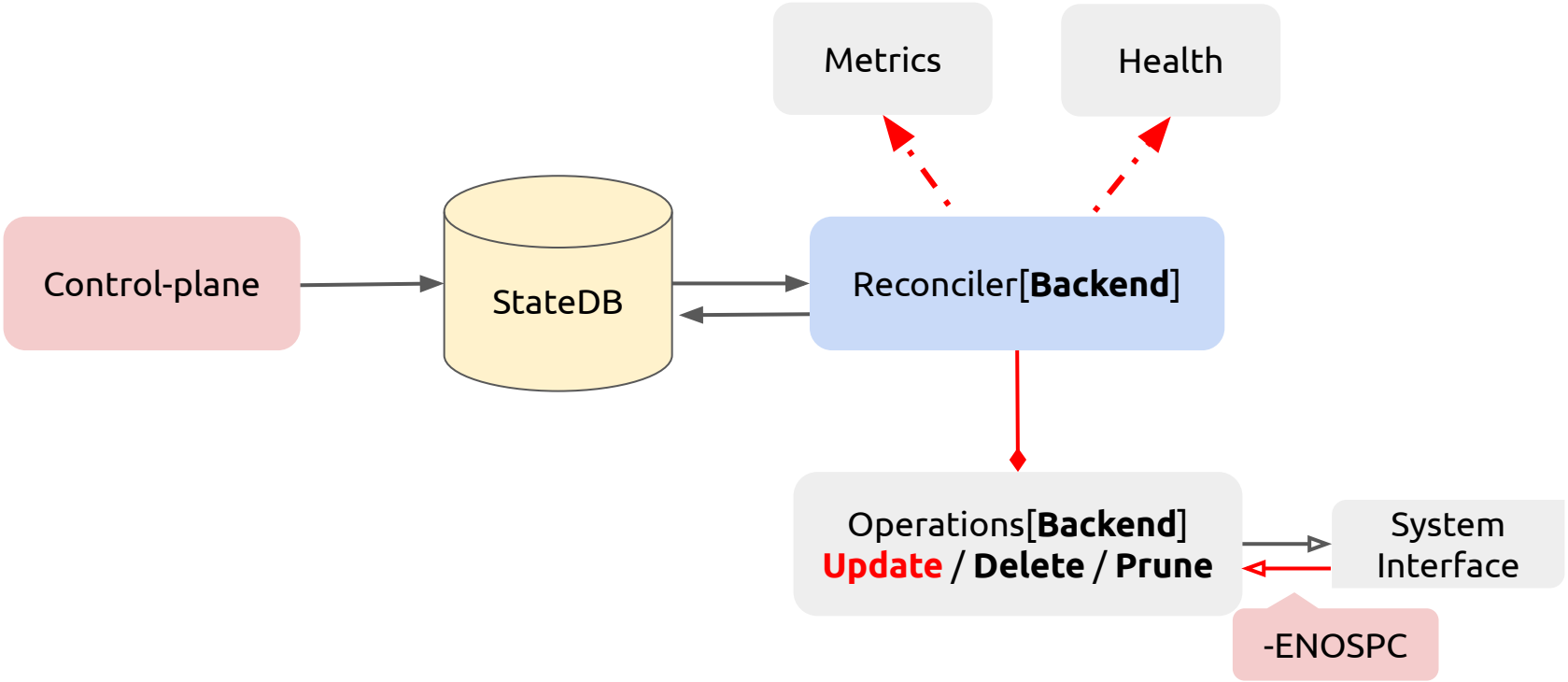
StateDB Reconciler



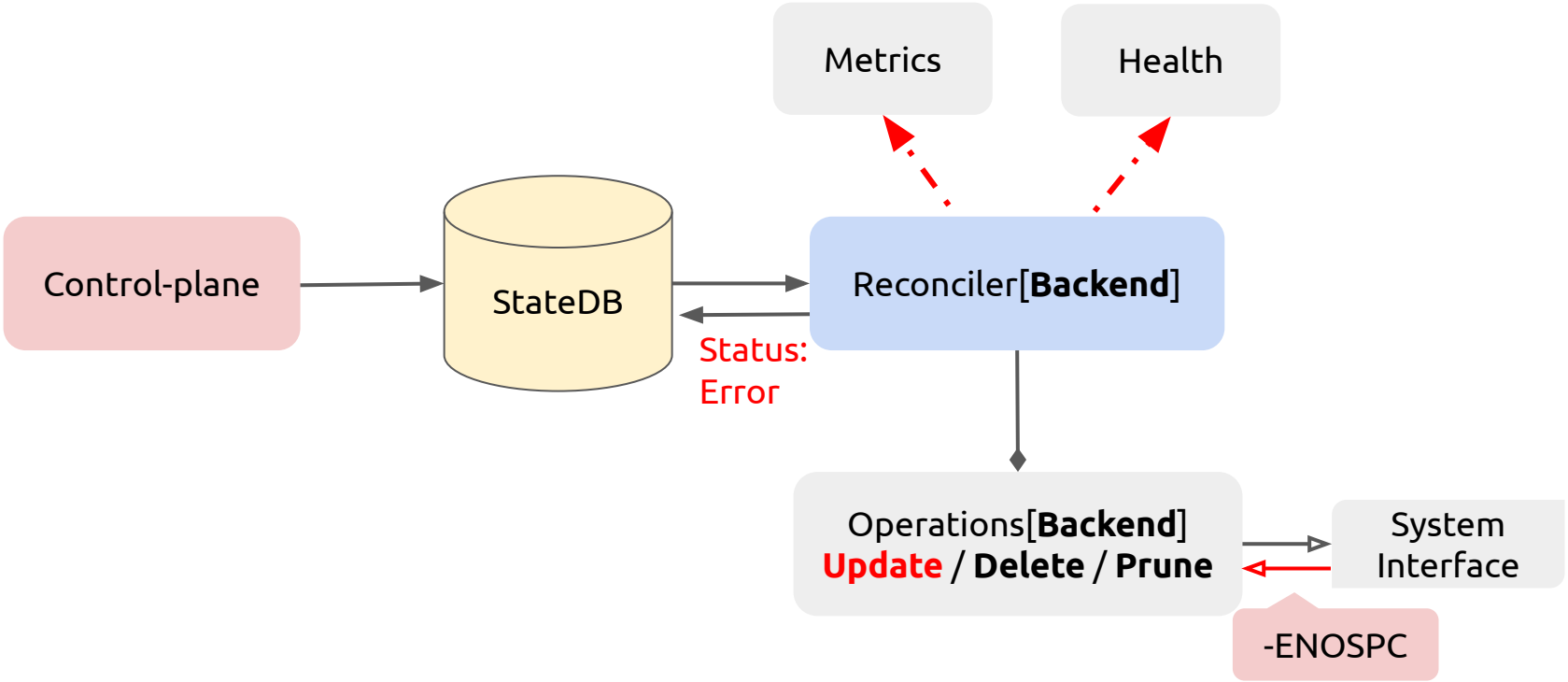
StateDB Reconciler



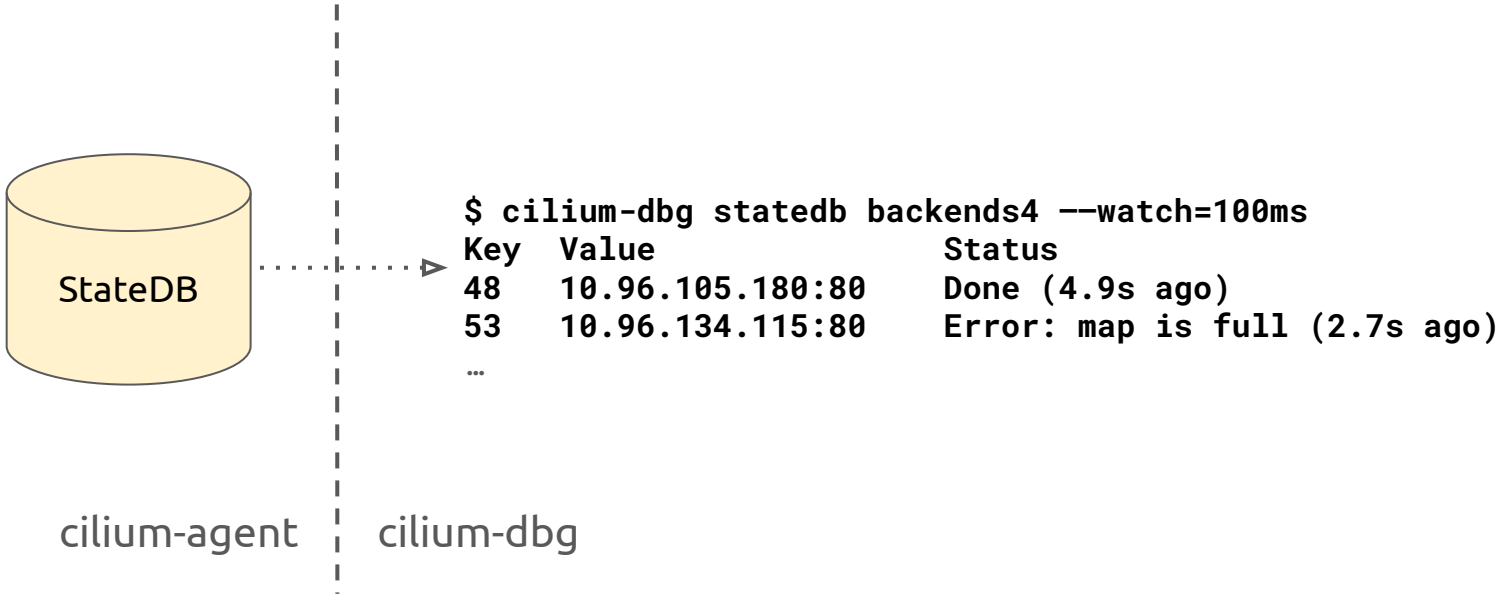
StateDB Reconciler



StateDB Reconciler



StateDB Reconciler



Summary

- Infrastructure is ready (StateDB, Hive and Agent Health, Reconciler)
- In Cilium v1.16 we’ll see first use-cases, for example runtime device reconfiguration.
- New features highly encouraged to leverage the new tooling for improved resiliency and observability

Thanks! Questions?

github.com/cilium/statedb (try out **reconciler/example**)