cloudstate

Towards Stateful Serverless

Jonas Bonér

@jboner

“We predict that Serverless Computing will grow to dominate the future of Cloud Computing.”

- BERKELEY CS DEPARTMENT

Cloud computing simplified: a Berkeley view on serverless computing

FaaS

FaaS = Function-as-a-Service

FaaS

IS VISIONARY

FaaS = Function-as-a-Service

FaaS

IS VISIONARY PAVED THE WAY

FaaS = Function-as-a-Service

FaaS

IS VISIONARY
PAVED THE WAY
JUST THE FIRST STEP

FaaS = Function-as-a-Service

SERVERLESS ≠ FAAS

GOOD USE-CASES FOR FAAS?

GOOD USE-CASES FOR FAAS?

Use-cases where throughput is key rather than low latency and requests can be completed in a short time window

GOOD USE-CASES FOR FAAS?

Use-cases where throughput is key rather than low latency and requests can be completed in a short time window

1. Embarrassingly parallel processing tasks—invoked on demand & intermittently, examples include: image processing, object recognition, log analysis
2. Low traffic applications—enterprise IT services, and spiky workloads
3. Stateless web applications—serving static content form S3 (or similar)
4. Orchestration functions—integration/coordination of calls to third-party services
5. Composing chains of functions—stateless workflow management, connected via data dependencies
6. Job scheduling—CRON jobs, triggers, etc.

FAAS: HARD TO BUILD GENERAL-PURPOSE APPLICATIONS

FAAS: HARD TO BUILD
GENERAL-PURPOSE APPLICATIONS

1. Functions are stateless, ephemeral, short-lived:
   expensive to lose computational context & rehydrate
2. Durable state is always “somewhere else”
3. No co-location of state and processing
4. No direct addressability—all communication over external storage
5. Limited options for managing & coordinating distributed state
6. Limited options for modelling data consistency guarantees

STATE

We Need Serverless Support For...

We Need Serverless Support For...

We Need Serverless Support For...

- Managing in-memory durable session state across individual requests

  E.g. User Sessions, Shopping Carts, Caching

We Need Serverless Support For...

- Managing in-memory durable session state across individual requests

  E.g. User Sessions, Shopping Carts, Caching
- Low-latency serving of dynamic in-memory models

  E.g. Serving of Machine Learning Models

We Need Serverless Support For...

- Managing in-memory durable session state across individual requests

  E.g. User Sessions, Shopping Carts, Caching
- Low-latency serving of dynamic in-memory models

  E.g. Serving of Machine Learning Models
- Real-time stream processing

  E.g. Recommendation, Anomaly Detection, Prediction Serving

We Need Serverless Support For...

- Managing in-memory durable session state across individual requests

  E.g. User Sessions, Shopping Carts, Caching
- Low-latency serving of dynamic in-memory models

  E.g. Serving of Machine Learning Models
- Real-time stream processing

  E.g. Recommendation, Anomaly Detection, Prediction Serving
- Distributed resilient transactional workflows

  E.g. Saga Pattern, Workflow Orchestration, Rollback/Compensating Actions

We Need Serverless Support For...

- **Managing in-memory durable session state across individual requests**

  E.g. User Sessions, Shopping Carts, Caching
- **Low-latency serving of dynamic in-memory models**

  E.g. Serving of Machine Learning Models
- **Real-time stream processing**

  E.g. Recommendation, Anomaly Detection, Prediction Serving
- **Distributed resilient transactional workflows**

  E.g. Saga Pattern, Workflow Orchestration, Rollback/Compensating Actions
- **Shared collaborative workspaces**

  E.g. Collaborative Document Editing, Blackboards, Chat Rooms

We Need Serverless Support For...

- Managing in-memory durable session state across individual requests

  E.g. User Sessions, Shopping Carts, Caching
- Low-latency serving of dynamic in-memory models

  E.g. Serving of Machine Learning Models
- Real-time stream processing

  E.g. Recommendation, Anomaly Detection, Prediction Serving
- Distributed resilient transactional workflows

  E.g. Saga Pattern, Workflow Orchestration, Rollback/Compensating Actions
- Shared collaborative workspaces

  E.g. Collaborative Document Editing, Blackboards, Chat Rooms
- Leader election, counting, voting

  . . . and other distributed systems patterns/protocols for coordination

Technical Requirements

Technical Requirements

1. Stateful long-lived addressable virtual components

   Actors

Technical Requirements

1. Stateful long-lived addressable virtual components

   Actors

2. Options for distributed coordination and communication patterns

   Pub-Sub, Point-To-Point, Broadcast—CRDTs, Sagas, etc.

Technical Requirements

1. Stateful long-lived addressable virtual components

   Actors
2. Options for distributed coordination and communication patterns

   Pub-Sub, Point-To-Point, Broadcast—CRDTs, Sagas, etc.
3. Options for managing distributed state reliably at scale

   Ranging from strong to eventual consistency (durable/ephemeral)

Technical Requirements

1. Stateful long-lived addressable virtual components

   Actors

2. Options for distributed coordination and communication patterns

   Pub-Sub, Point-To-Point, Broadcast—CRDTs, Sagas, etc.

3. Options for managing distributed state reliably at scale

   Ranging from strong to eventual consistency (durable/ephemeral)

4. Intelligent adaptive placement of stateful functions

   Physical co-location of state and processing, sharding, and sticky routing

Technical Requirements

1. **Stateful long-lived addressable virtual components**

   Actors

2. **Options for distributed coordination and communication patterns**

   Pub-Sub, Point-To-Point, Broadcast—CRDTs, Sagas, etc.

3. **Options for managing distributed state reliably at scale**

   Ranging from strong to eventual consistency (durable/ephemeral)

4. **Intelligent adaptive placement of stateful functions**

   Physical co-location of state and processing, sharding, and sticky routing

5. **Predictable performance, latency, and throughput**

   In startup time, communication/coordination, and storage of data

Technical Requirements

1. Stateful long-lived addressable virtual components

   Actors
2. Options for distributed coordination and communication patterns

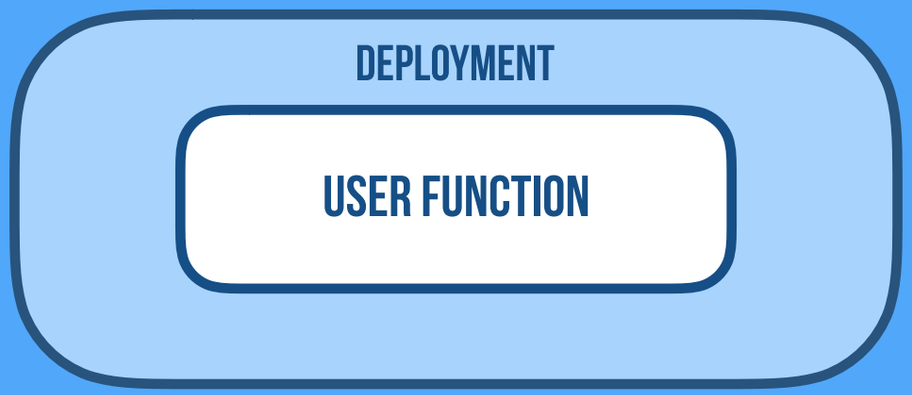
   Pub-Sub, Point-To-Point, Broadcast—CRDTs, Sagas, etc.
3. Options for managing distributed state reliably at scale

   Ranging from strong to eventual consistency (durable/ephemeral)
4. Intelligent adaptive placement of stateful functions

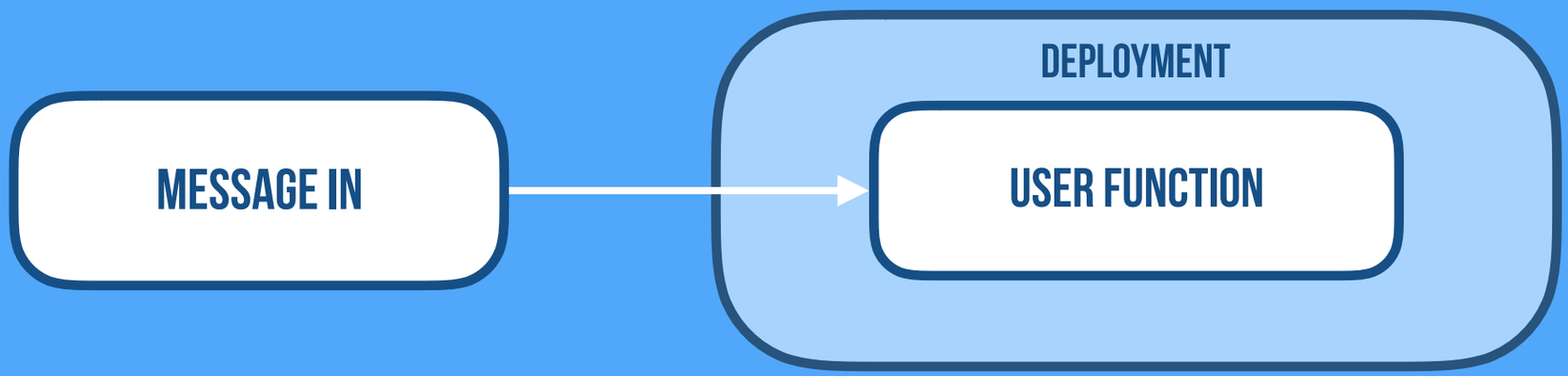
   Physical co-location of state and processing, sharding, and sticky routing
5. Predictable performance, latency, and throughput

   In startup time, communication/coordination, and storage of data
6. Ways of managing end-to-end guarantees and correctness

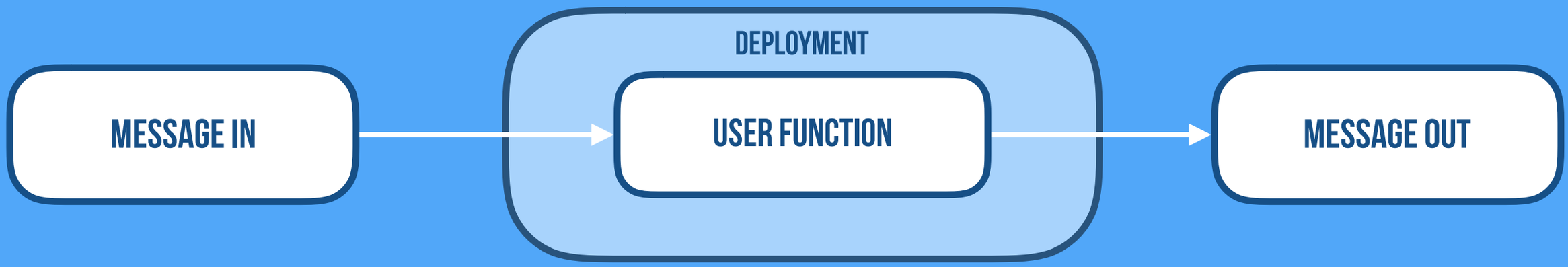
FaaS Is Great At Abstracting Over Communication



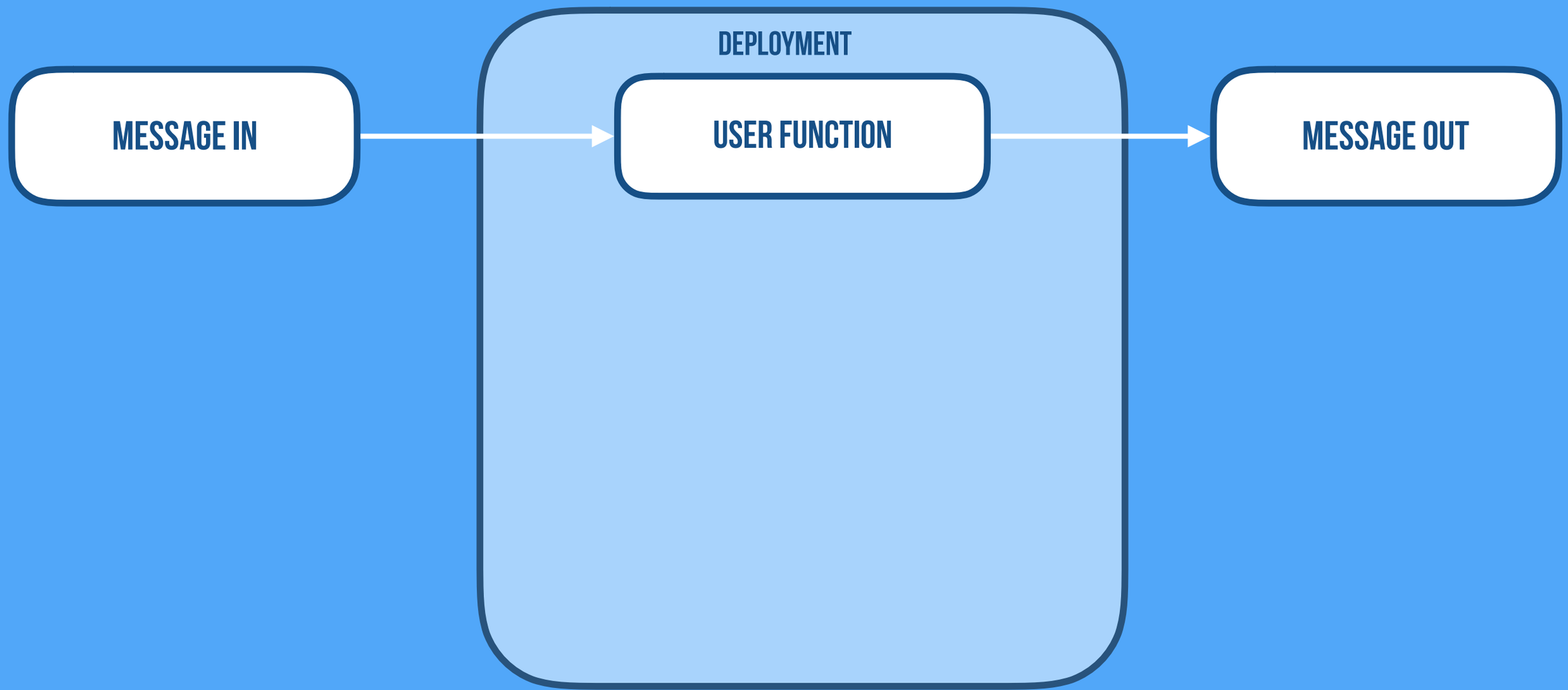
FaaS Is Great At Abstracting Over Communication



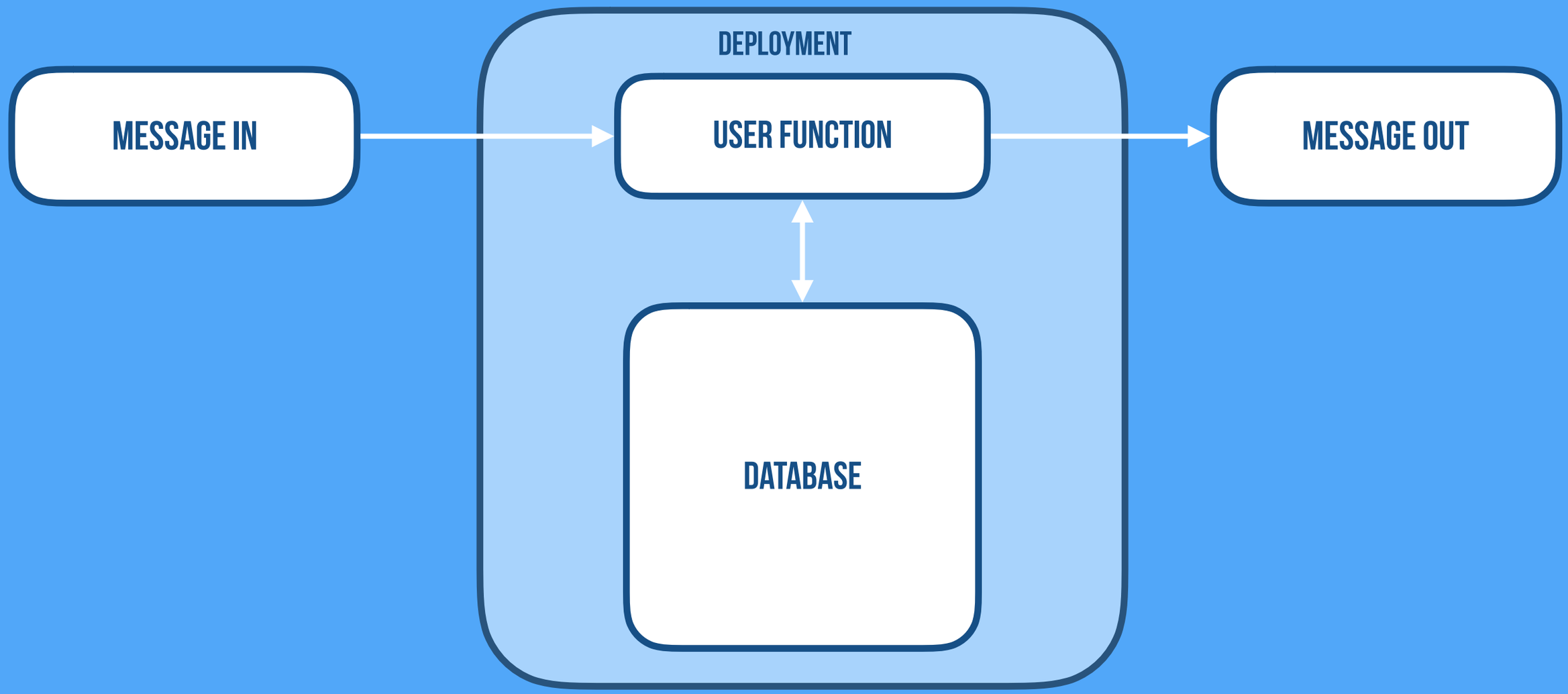
FaaS Is Great At Abstracting Over Communication



FaaS With CRUD

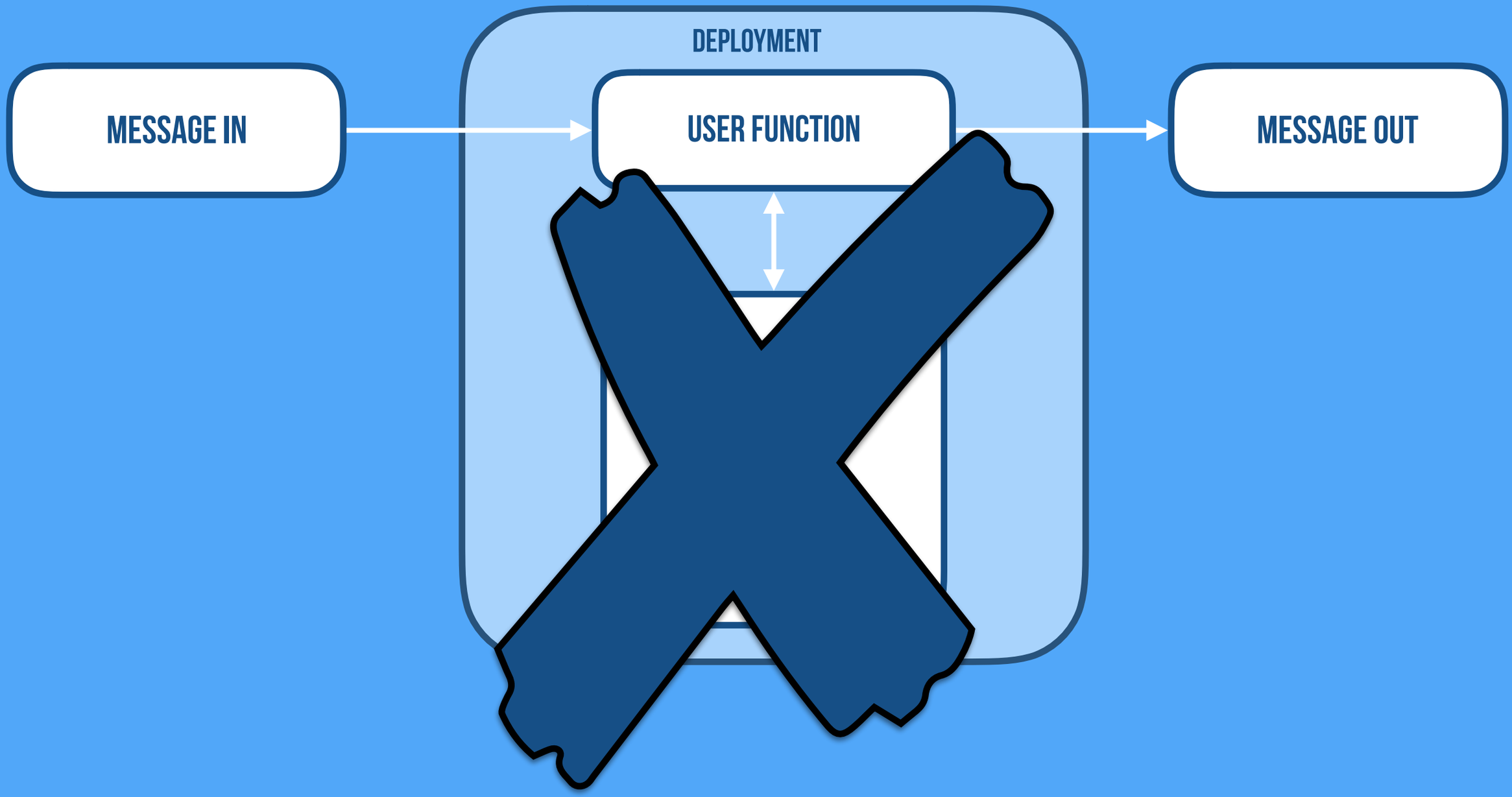


FaaS With CRUD



Not Serverless

Leaky Abstraction



The Problem

The Problem

THE FUNCTION IS A

BLACK BOX

The Problem

The Problem

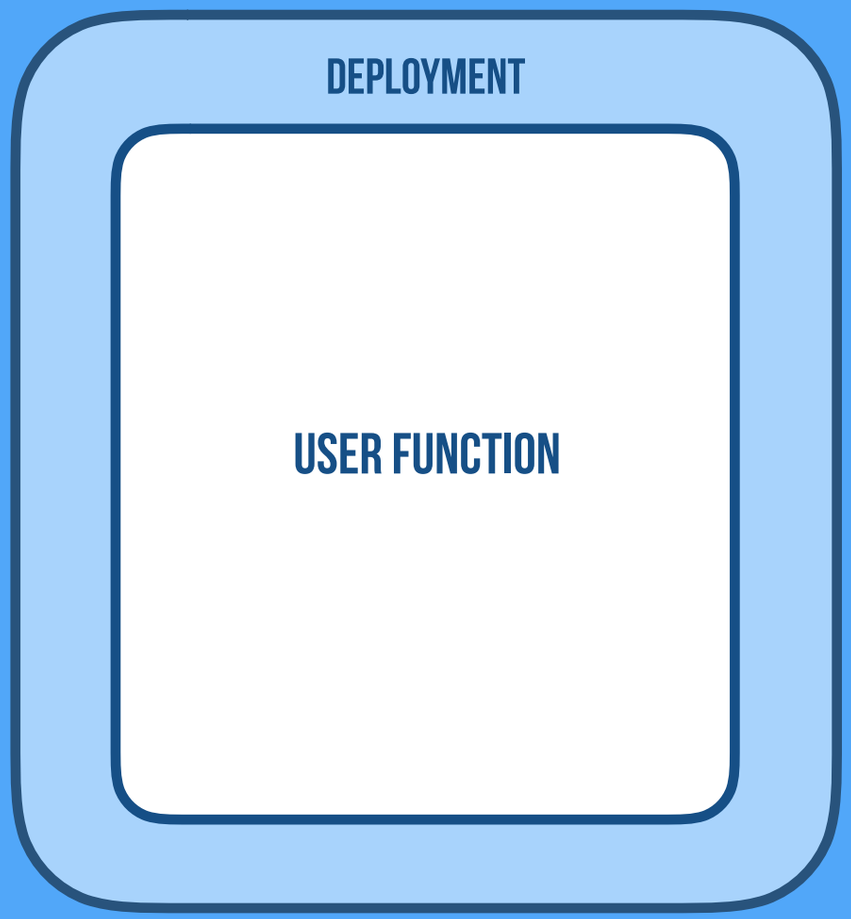
UNCONSTRAINED DATABASE ACCESS MAKES IT HARD TO AUTOMATE OPERATIONS

“Freedom is not so much the absence of restrictions as finding the right ones, the liberating restrictions.”

- TIMOTHY KELLER

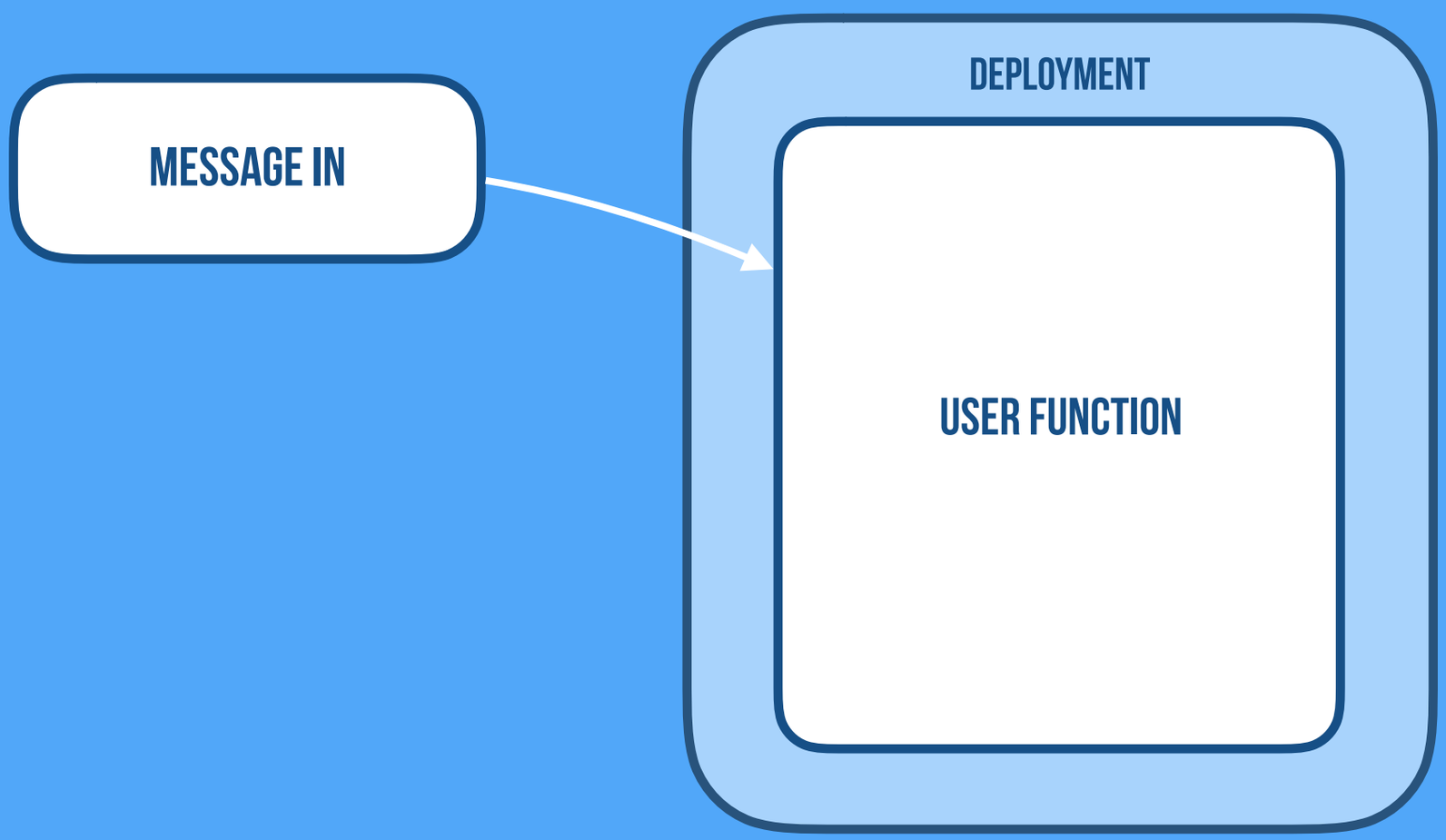
FaaS

Abstracting Over Communication



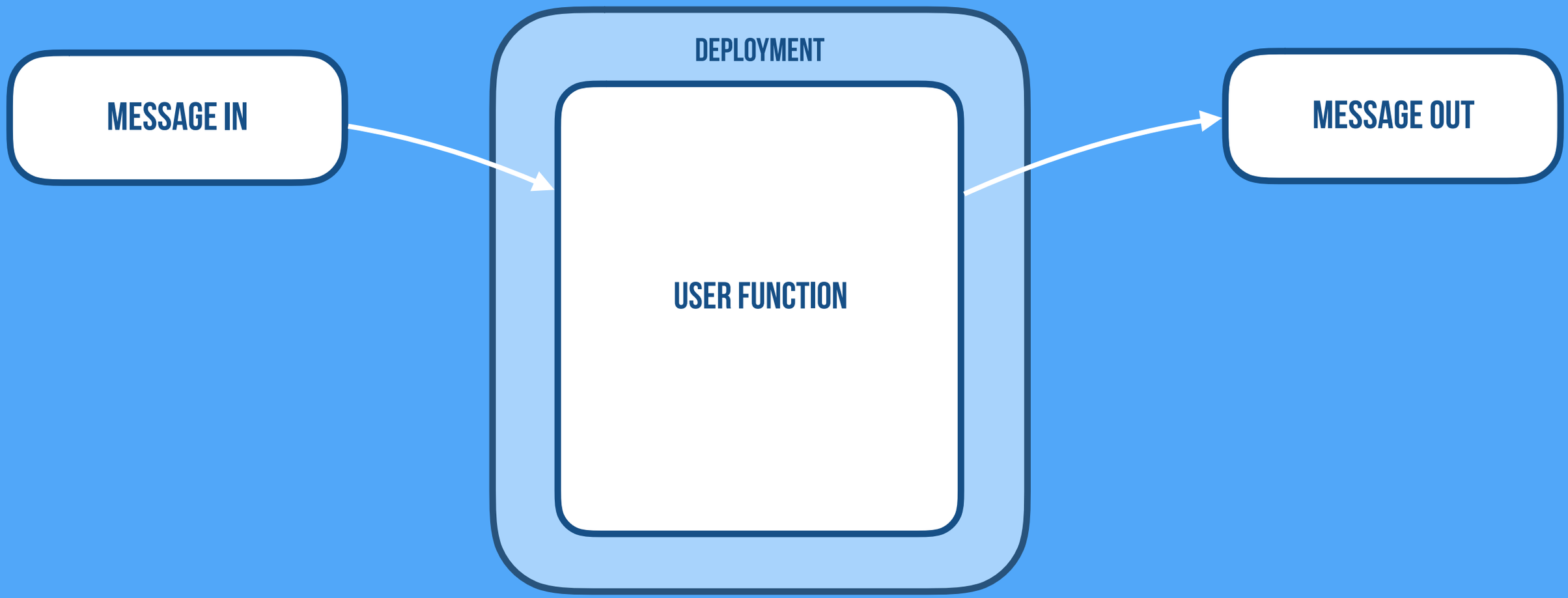
FaaS

Abstracting Over Communication

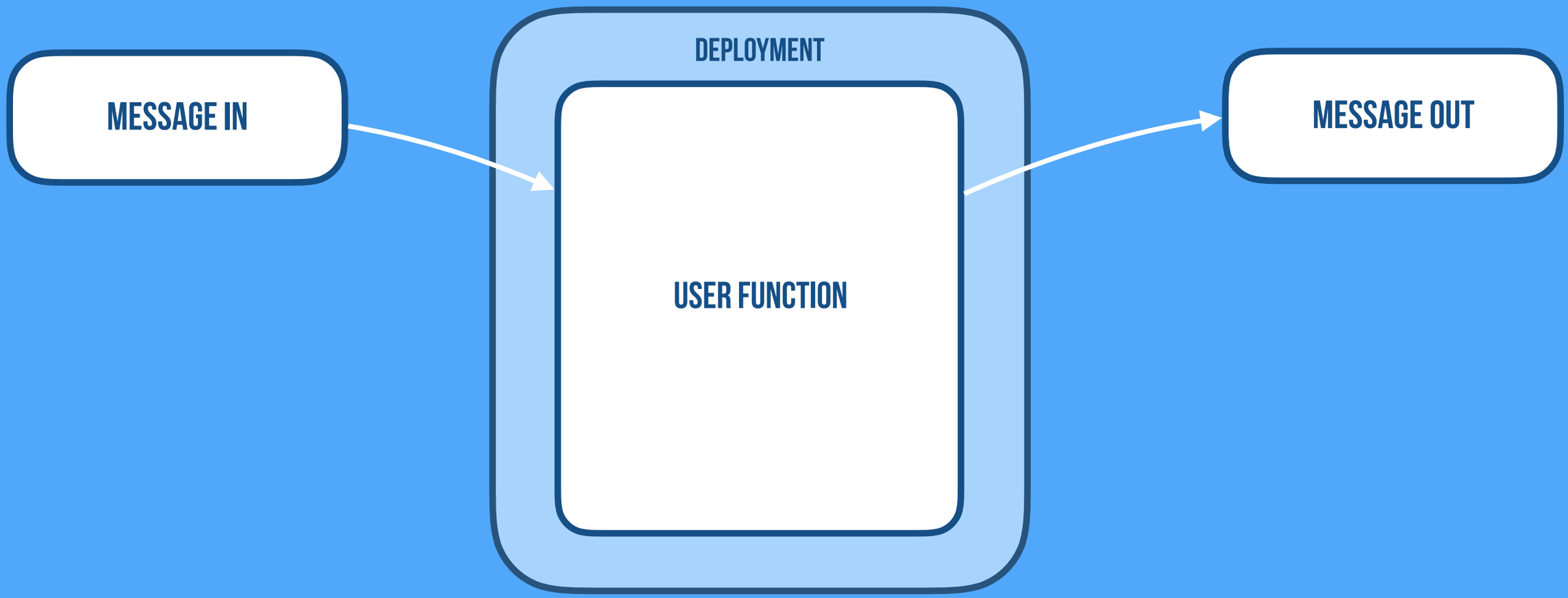


FaaS

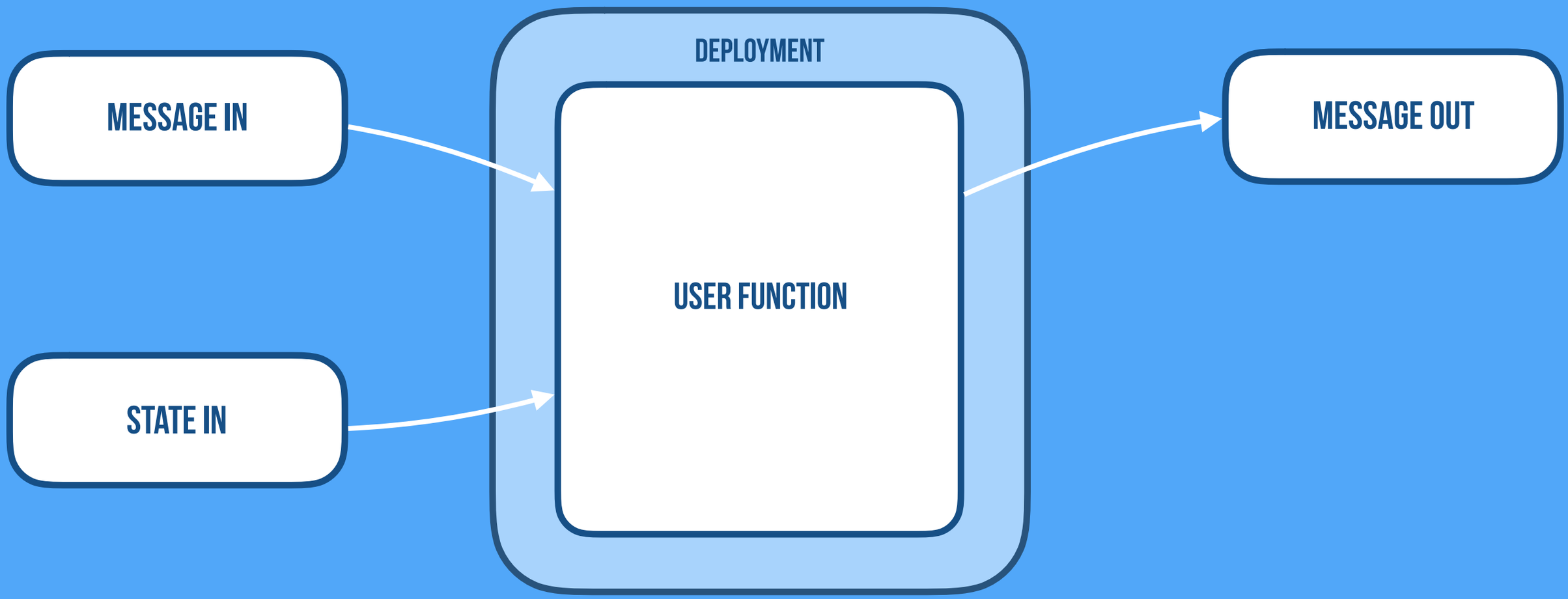
Abstracting Over Communication



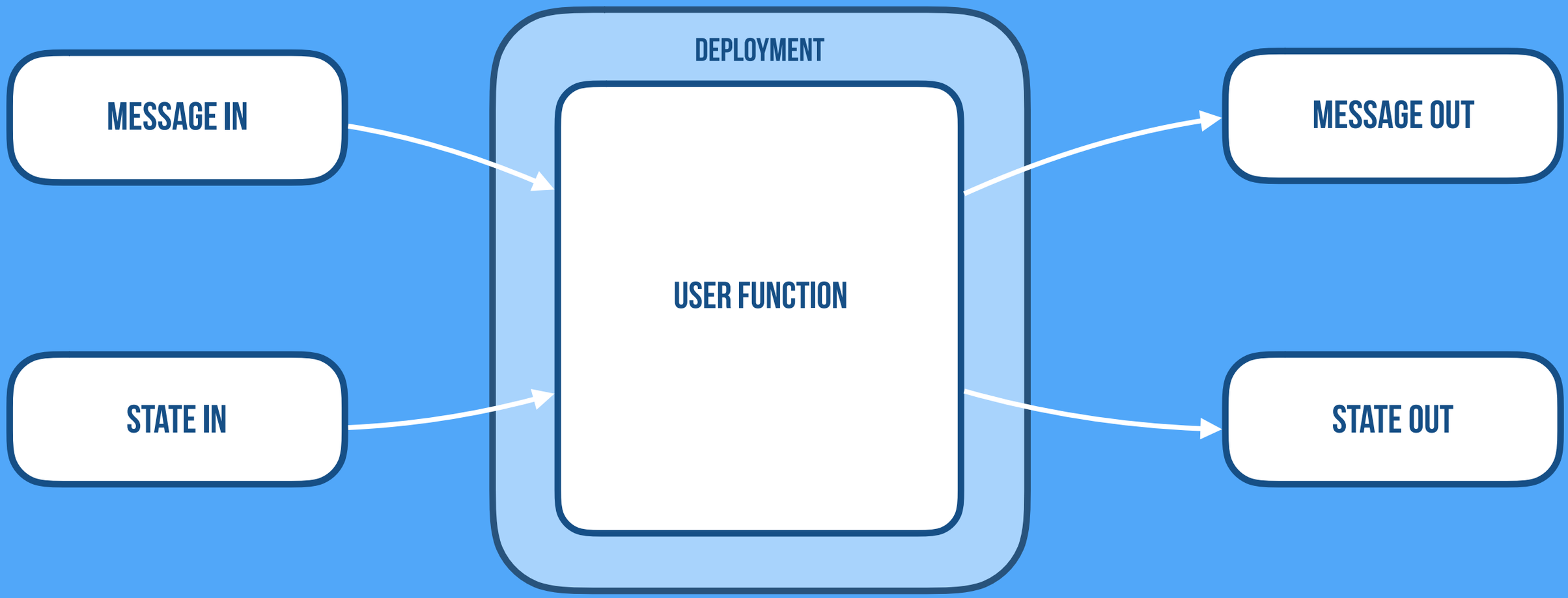
Stateful Serverless
Abstracting Over State



Stateful Serverless
Abstracting Over State



Stateful Serverless
Abstracting Over State



Enter

cloudstate

WHAT IS CLOUDSTATE?

https://cloudstate.io

WHAT IS CLOUDSTATE?

https://cloudstate.io

Overview:

1. Open Source (Apache 2.0) project

WHAT IS CLOUDSTATE?

https://cloudstate.io

Overview:

1. Open Source (Apache 2.0) project
2. Makes Stateful Serverless applications easy

WHAT IS CLOUDSTATE?

https://cloudstate.io

Overview:

1. Open Source (Apache 2.0) project
2. Makes Stateful Serverless applications easy
3. Reference implementation for a standard (protocol and spec)

WHAT IS CLOUDSTATE?

https://cloudstate.io

Overview:

1. Open Source (Apache 2.0) project
2. Makes Stateful Serverless applications easy
3. Reference implementation for a standard (protocol and spec)
4. Let's you focus on business logic, data model, and workflow

WHAT IS CLOUDSTATE?

https://cloudstate.io

WHAT IS CLOUDSTATE?

https://cloudstate.io

Don't worry about:

1. Managing: Complexities of Distributed and Concurrent systems

WHAT IS CLOUDSTATE?

https://cloudstate.io

Don't worry about:

1. Managing: Complexities of Distributed and Concurrent systems
2. Managing: Distributed State—Consistency, Replication, Persistence

WHAT IS CLOUDSTATE?

https://cloudstate.io

Don't worry about:

1. Managing: Complexities of Distributed and Concurrent systems
2. Managing: Distributed State—Consistency, Replication, Persistence
3. Managing: Databases, Service Meshes, and other infrastructure

WHAT IS CLOUDSTATE?

https://cloudstate.io

Don't worry about:

1. Managing: Complexities of Distributed and Concurrent systems
2. Managing: Distributed State—Consistency, Replication, Persistence
3. Managing: Databases, Service Meshes, and other infrastructure
4. Managing: Message Routing, Scalability, Fail-over & Recovery

WHAT IS CLOUDSTATE?

https://cloudstate.io

Don't worry about:

1. Managing: Complexities of Distributed and Concurrent systems
2. Managing: Distributed State—Consistency, Replication, Persistence
3. Managing: Databases, Service Meshes, and other infrastructure
4. Managing: Message Routing, Scalability, Fail-over & Recovery
5. Running & Operating your application

WHAT IS CLOUDSTATE?

https://cloudstate.io

WHAT IS CLOUDSTATE?

https://cloudstate.io

Technical Highlights:

1. Polyglot: Client libs in JavaScript, Java, Go—with upcoming support for Python, .NET, Rust, Swift, Scala

WHAT IS CLOUDSTATE?

https://cloudstate.io

Technical Highlights:

1. Polyglot: Client libs in JavaScript, Java, Go—with upcoming support for Python, .NET, Rust, Swift, Scala
2. PolyState: Powerful state models—Event Sourcing, CRDTs, Key Value

WHAT IS CLOUDSTATE?

https://cloudstate.io

Technical Highlights:

1. Polyglot: Client libs in JavaScript, Java, Go—with upcoming support for Python, .NET, Rust, Swift, Scala
2. PolyState: Powerful state models—Event Sourcing, CRDTs, Key Value
3. PolyDB: Supporting SQL, NoSQL, NewSQL and in-memory replication

WHAT IS CLOUDSTATE?

https://cloudstate.io

Technical Highlights:

1. Polyglot: Client libs in JavaScript, Java, Go—with upcoming support for Python, .NET, Rust, Swift, Scala
2. PolyState: Powerful state models—Event Sourcing, CRDTs, Key Value
3. PolyDB: Supporting SQL, NoSQL, NewSQL and in-memory replication
4. Leveraging Akka, gRPC, Knative, GraalVM, running on Kubernetes

CLOUDSTATE ARCHITECTURE

CLOUDSTATE ARCHITECTURE

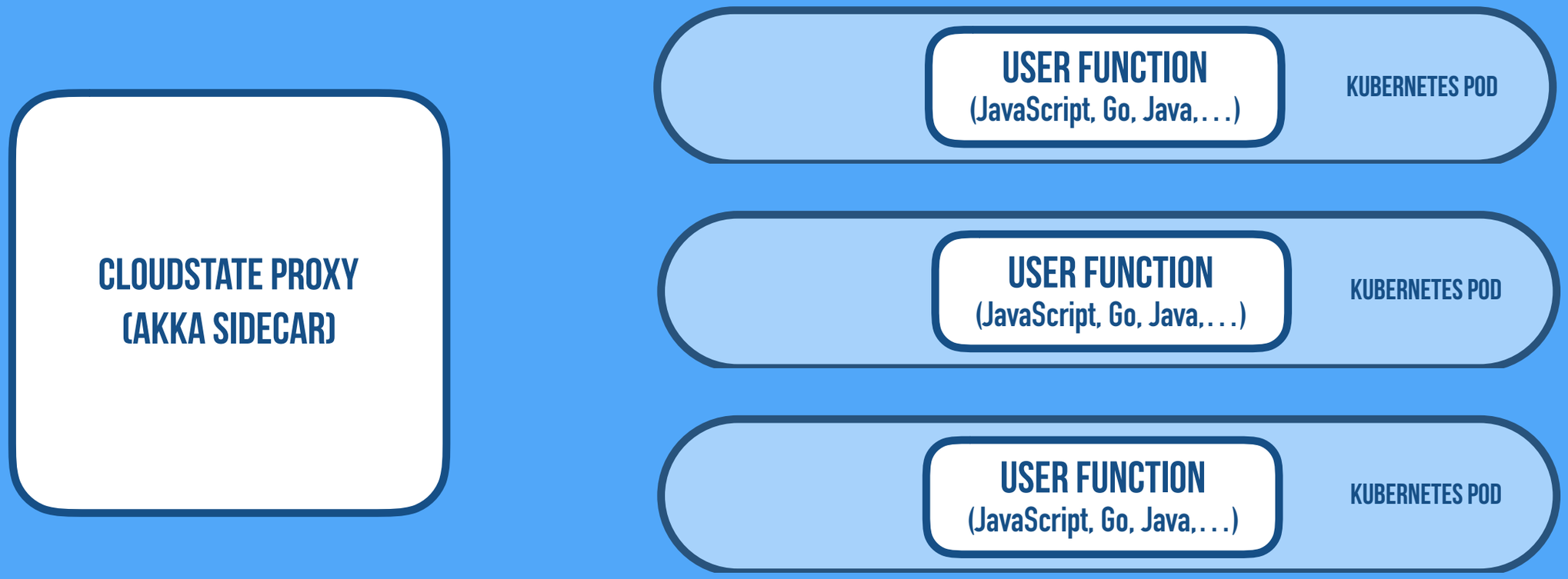
KUBERNETES POD

KUBERNETES POD

KUBERNETES POD

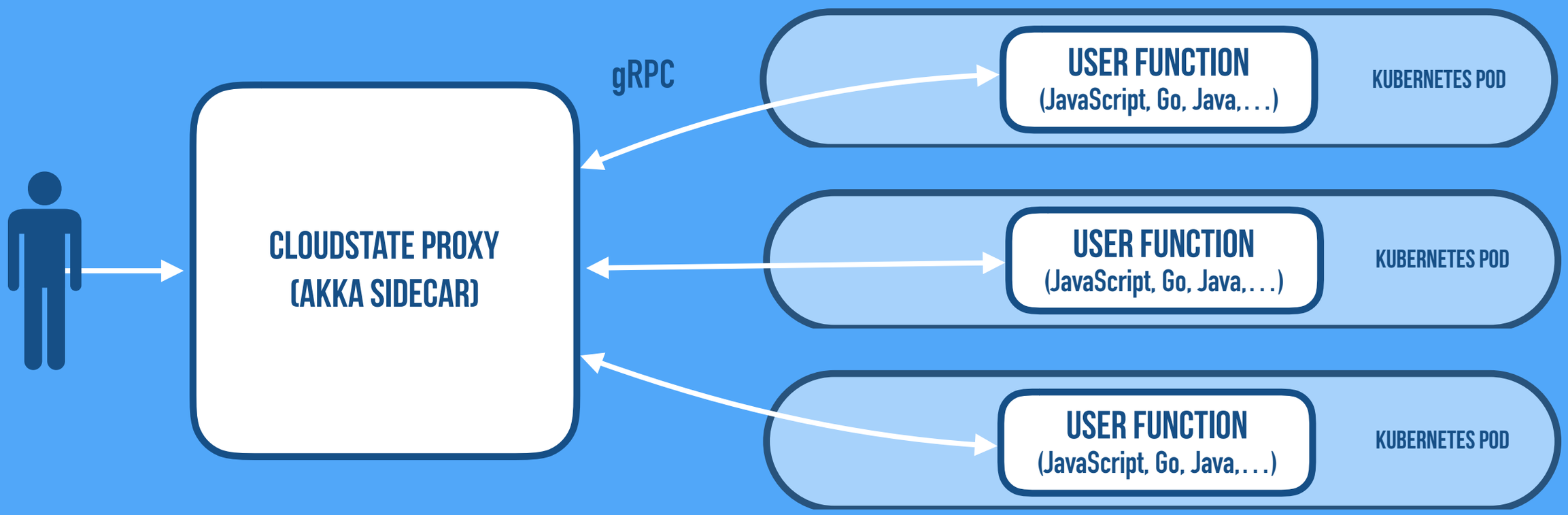
CLOUDSTATE ARCHITECTURE

CLOUDSTATE ARCHITECTURE

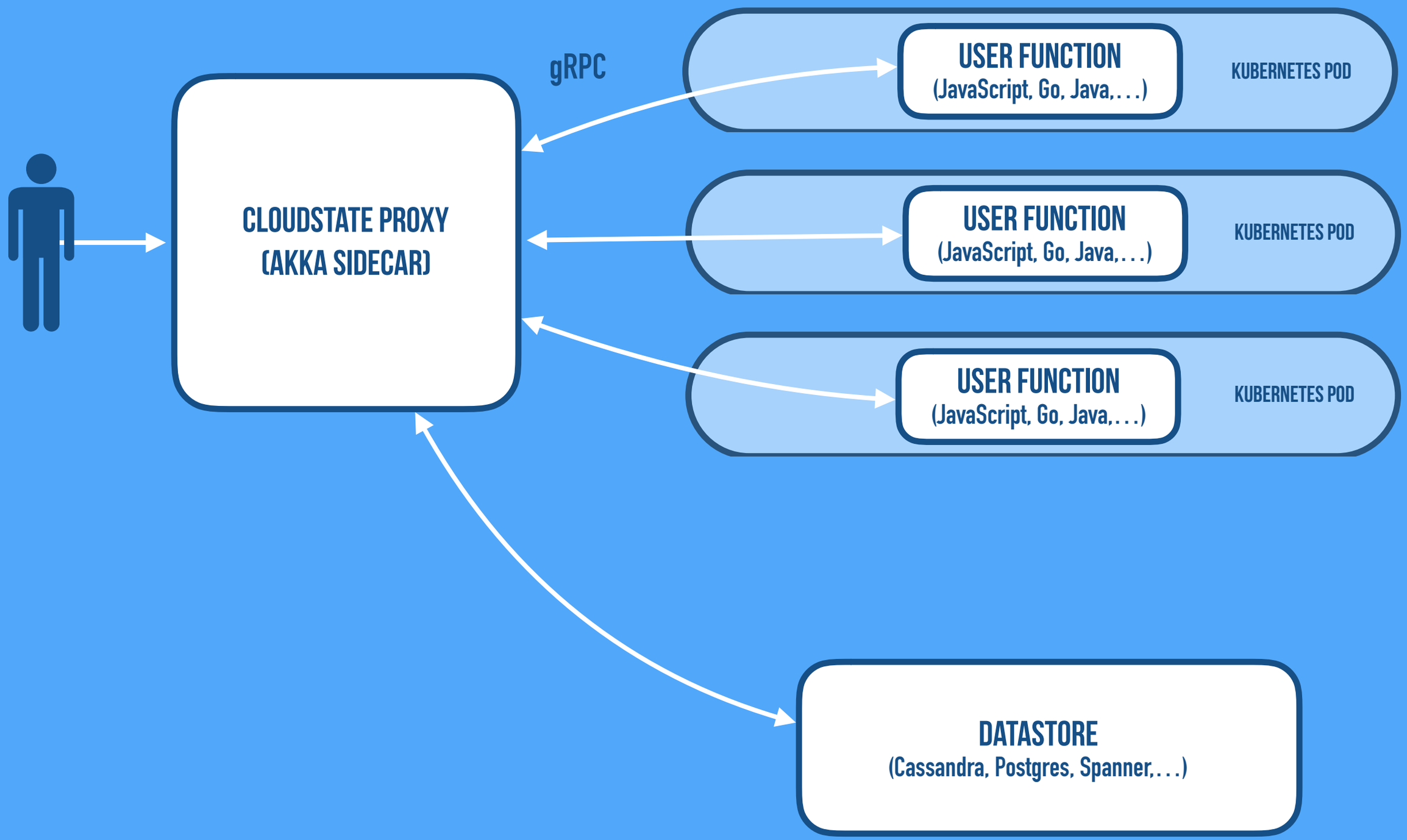


CLOUDSTATE ARCHITECTURE

CLOUDSTATE ARCHITECTURE



CLOUDSTATE ARCHITECTURE




USER FUNCTION
(JavaScript, Go, Java, . . .)
KUBERNETES POD


USER FUNCTION
(JavaScript, Go, Java, . . .)
KUBERNETES POD


USER FUNCTION
(JavaScript, Go, Java, . . .)
KUBERNETES POD

cloudstate


AKKA SIDECAR
USER FUNCTION
(JavaScript, Go, Java, . . .)
KUBERNETES POD


AKKA SIDECAR
USER FUNCTION
(JavaScript, Go, Java, . . .)
KUBERNETES POD


AKKA SIDECAR
USER FUNCTION
(JavaScript, Go, Java, . . .)
KUBERNETES POD

cloudstate

AKKA CLUSTER

AKKA SIDECAR

USER FUNCTION
(JavaScript, Go, Java, . . .)

KUBERNETES POD

AKKA SIDECAR

USER FUNCTION
(JavaScript, Go, Java, . . .)

KUBERNETES POD

AKKA SIDECAR

USER FUNCTION
(JavaScript, Go, Java, . . .)

KUBERNETES POD

cloudstate

AKKA CLUSTER

HTTP

AKKA SIDECAR

USER FUNCTION
(JavaScript, Go, Java, . . .)

KUBERNETES POD

AKKA SIDECAR

USER FUNCTION
(JavaScript, Go, Java, . . .)

KUBERNETES POD

AKKA SIDECAR

USER FUNCTION
(JavaScript, Go, Java, . . .)

KUBERNETES POD

cloudstate

AKKA CLUSTER

HTTP

AKKA SIDECAR

USER FUNCTION
(JavaScript, Go, Java, . . .)

KUBERNETES POD

AKKA SIDECAR

USER FUNCTION
(JavaScript, Go, Java, . . .)

KUBERNETES POD

AKKA SIDECAR

USER FUNCTION
(JavaScript, Go, Java, . . .)

KUBERNETES POD

cloudstate

AKKA CLUSTER

HTTP

AKKA SIDECAR

gRPC

USER FUNCTION
(JavaScript, Go, Java, . . .)

KUBERNETES POD

AKKA SIDECAR

USER FUNCTION
(JavaScript, Go, Java, . . .)

KUBERNETES POD

AKKA SIDECAR

USER FUNCTION
(JavaScript, Go, Java, . . .)

KUBERNETES POD

cloudstate

AKKA CLUSTER

HTTP

AKKA SIDECAR

gRPC

USER FUNCTION
(JavaScript, Go, Java, . . .)

KUBERNETES POD

Gossip, State replication, Routing

AKKA SIDECAR

USER FUNCTION
(JavaScript, Go, Java, . . .)

KUBERNETES POD

Gossip, State replication, Routing

AKKA SIDECAR

USER FUNCTION
(JavaScript, Go, Java, . . .)

KUBERNETES POD

cloudstate

AKKA CLUSTER

HTTP

AKKA SIDECAR

gRPC

USER FUNCTION
(JavaScript, Go, Java, . . .)

KUBERNETES POD

Gossip, State replication, Routing

AKKA SIDECAR

USER FUNCTION
(JavaScript, Go, Java, . . .)

KUBERNETES POD

Gossip, State replication, Routing

AKKA SIDECAR

USER FUNCTION
(JavaScript, Go, Java, . . .)

KUBERNETES POD

DATASTORE
(Cassandra, Postgres, Spanner, . . .)

cloudstate

AKKA CLUSTER

HTTP

gRPC

AKKA SIDECAR

gRPC

USER FUNCTION
(JavaScript, Go, Java, . . .)

KUBERNETES POD

Gossip, State replication, Routing

gRPC

AKKA SIDECAR

gRPC

USER FUNCTION
(JavaScript, Go, Java, . . .)

KUBERNETES POD

Gossip, State replication, Routing

AKKA SIDECAR

USER FUNCTION
(JavaScript, Go, Java, . . .)

KUBERNETES POD

DATASTORE
(Cassandra, Postgres, Spanner, . . .)

cloudstate

AKKA CLUSTER

HTTP

gRPC

Events

AKKA SIDECAR

gRPC

USER FUNCTION
(JavaScript, Go, Java, . . .)

KUBERNETES POD

Gossip, State replication, Routing

AKKA SIDECAR

gRPC

USER FUNCTION
(JavaScript, Go, Java, . . .)

KUBERNETES POD

Gossip, State replication, Routing

AKKA SIDECAR

gRPC

USER FUNCTION
(JavaScript, Go, Java, . . .)

KUBERNETES POD

DATASTORE
(Cassandra, Postgres, Spanner, . . .)

CLOUDSTATE HELPS YOU WITH

(when being a managed service)

CLOUDSTATE HELPS YOU WITH

(when being a managed service)

- Pay-as-you-go:
  - On-demand Instance Creation, Passivation, and Failover
  - Autoscaling—up and down

CLOUDSTATE HELPS YOU WITH

(when being a managed service)

- Pay-as-you-go:
  - On-demand Instance Creation, Passivation, and Failover
  - Autoscaling—up and down
- ZeroOps:
  - Automation of Message Routing and Delivery
  - Automation of State Management
    - Service of Record—In-Memory Cluster Sharding, Co-location of Data & Processing
    - Coordination State—Replication, Consistency
  - Automation of Deployment, Provisioning, Upgrades

AKKA CLUSTER STATE MANAGEMENT

AKKA CLUSTER STATE MANAGEMENT

AKKA SIDECAR

AKKA SIDECAR

AKKA SIDECAR

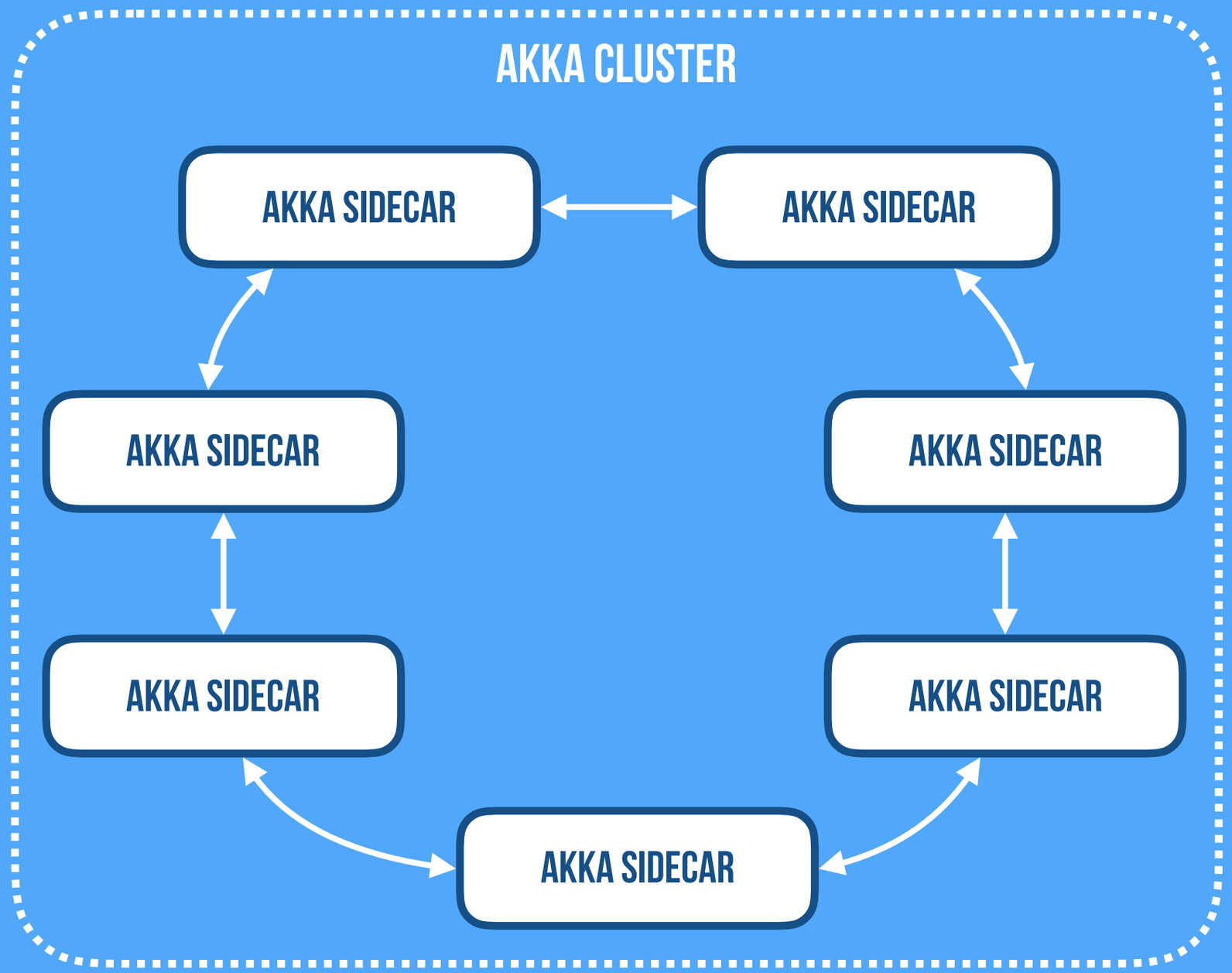
AKKA SIDECAR

AKKA SIDECAR

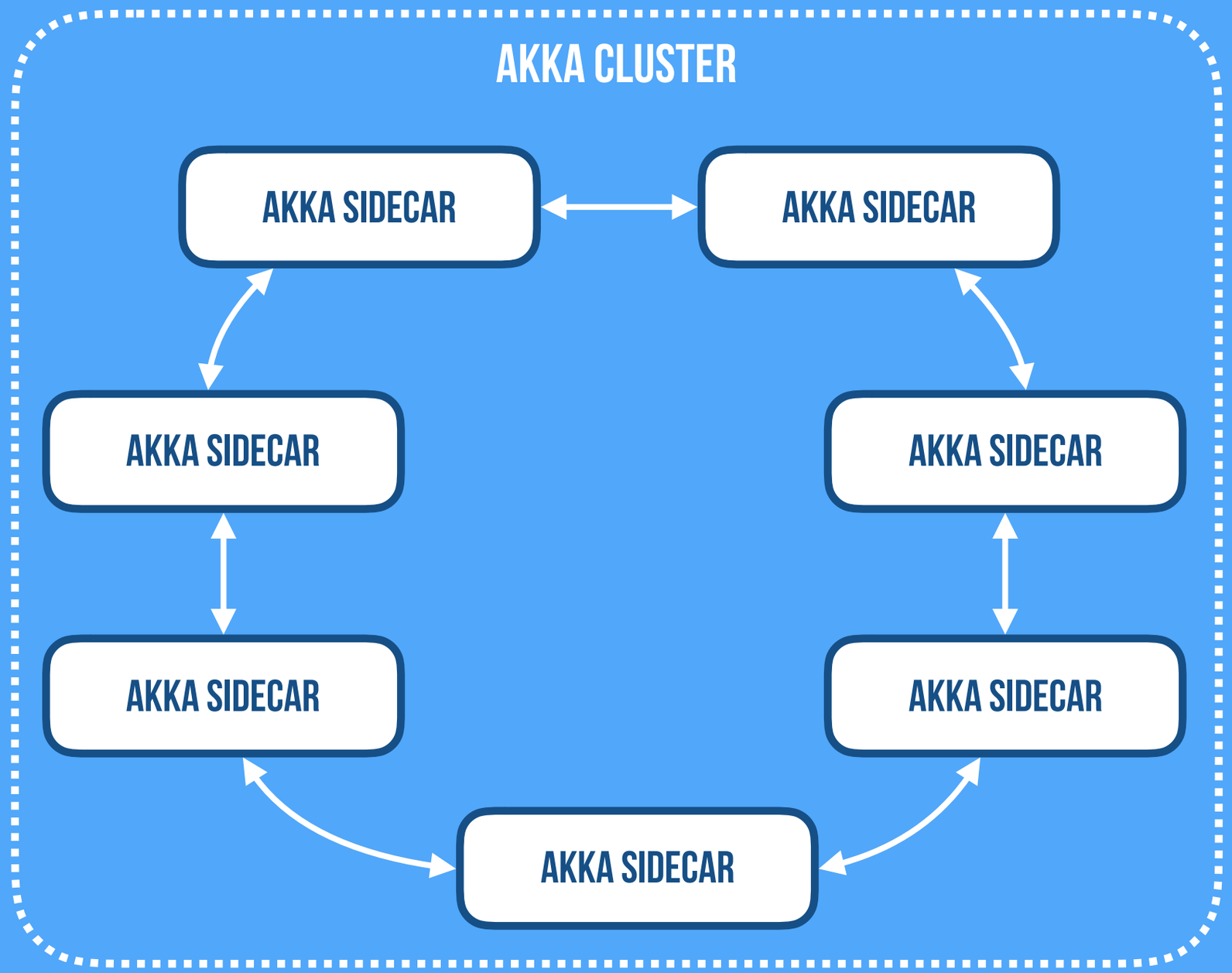
AKKA SIDECAR

AKKA SIDECAR

AKKA CLUSTER STATE MANAGEMENT



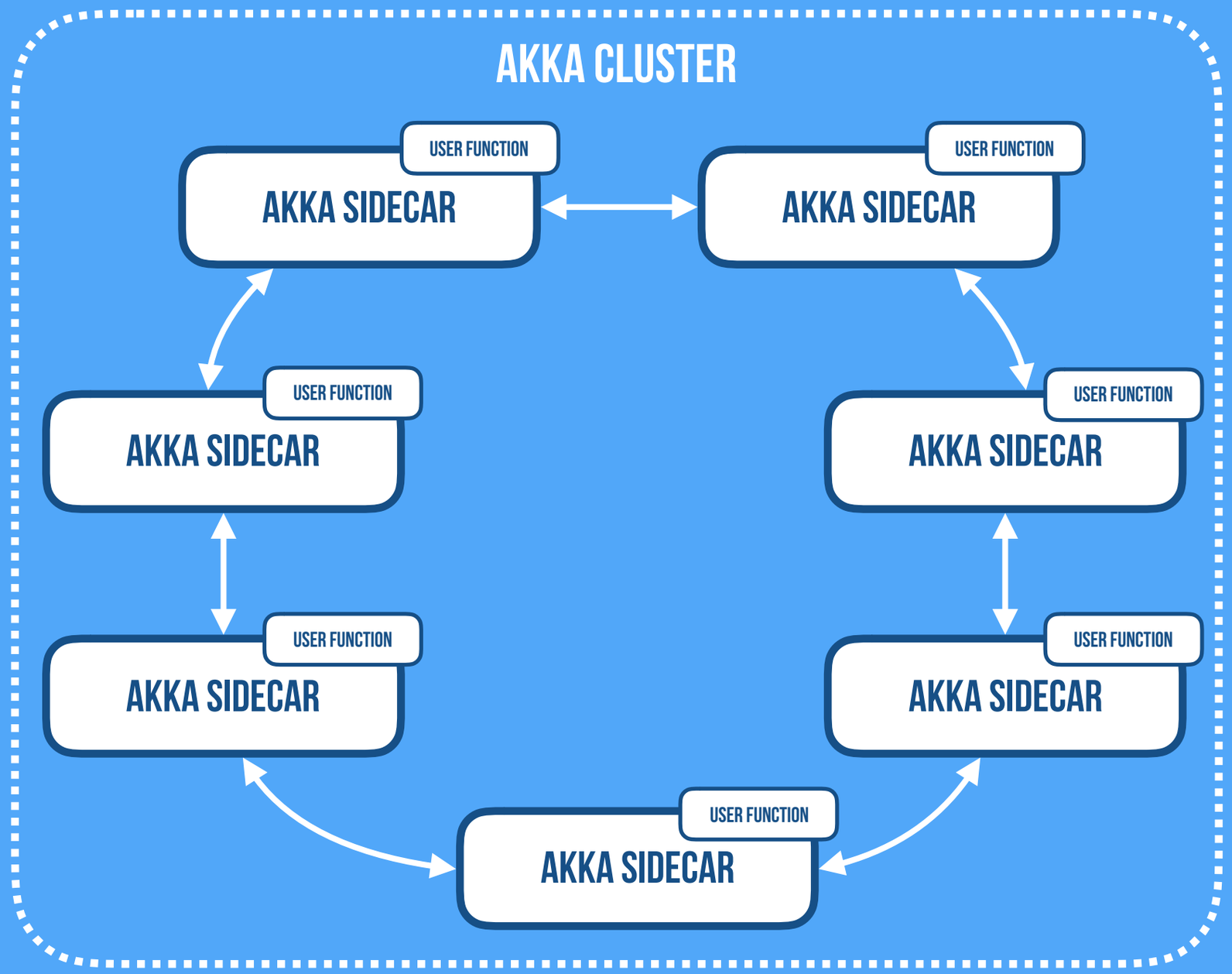
AKKA CLUSTER STATE MANAGEMENT



- Actor-based Distributed Runtime
- Decentralized Masterless P2P
- Epidemic Gossiping, Self-healing

https://akka.io

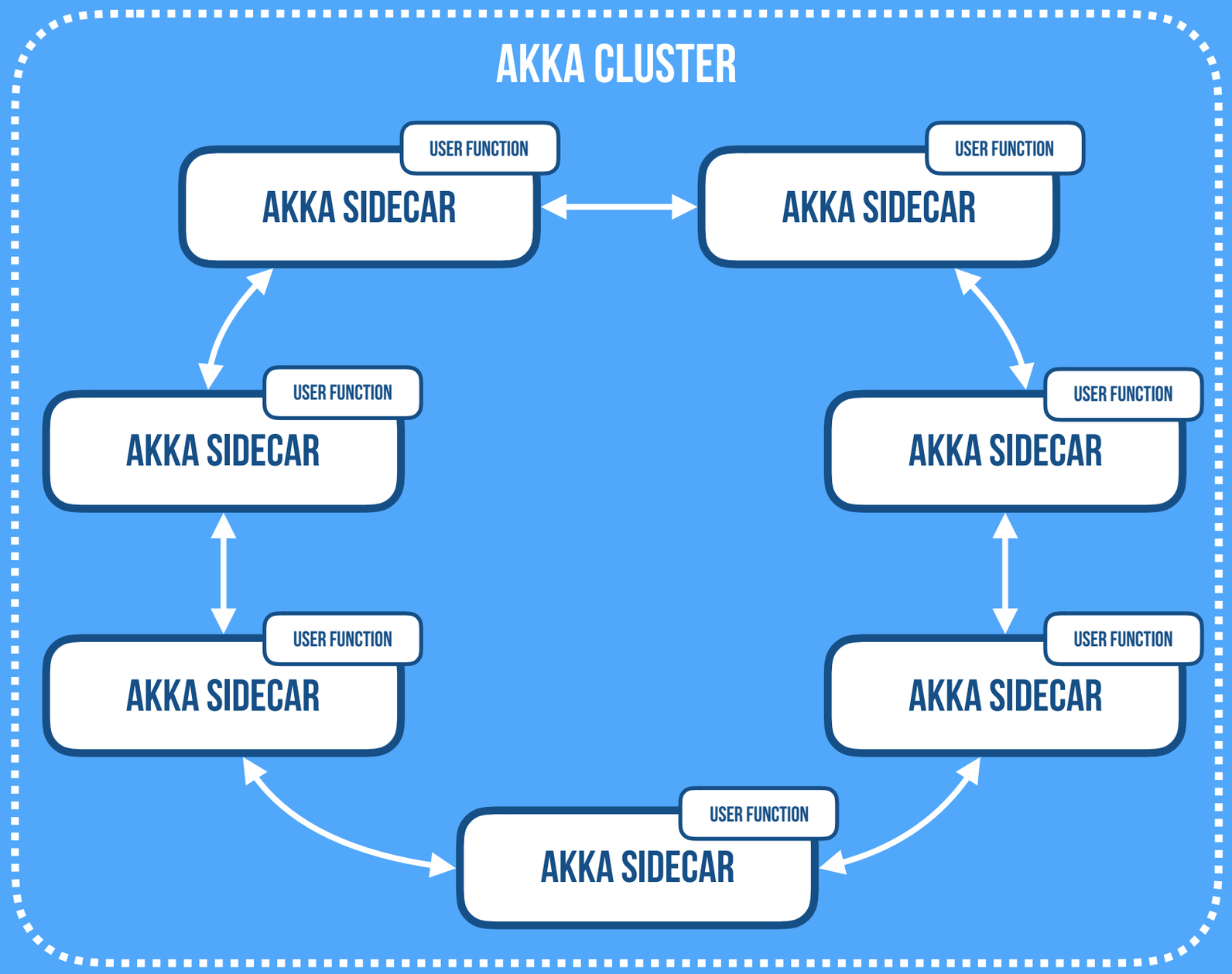
AKKA CLUSTER STATE MANAGEMENT



- Actor-based Distributed Runtime
- Decentralized Masterless P2P
- Epidemic Gossiping, Self-healing

https://akka.io

AKKA CLUSTER STATE MANAGEMENT

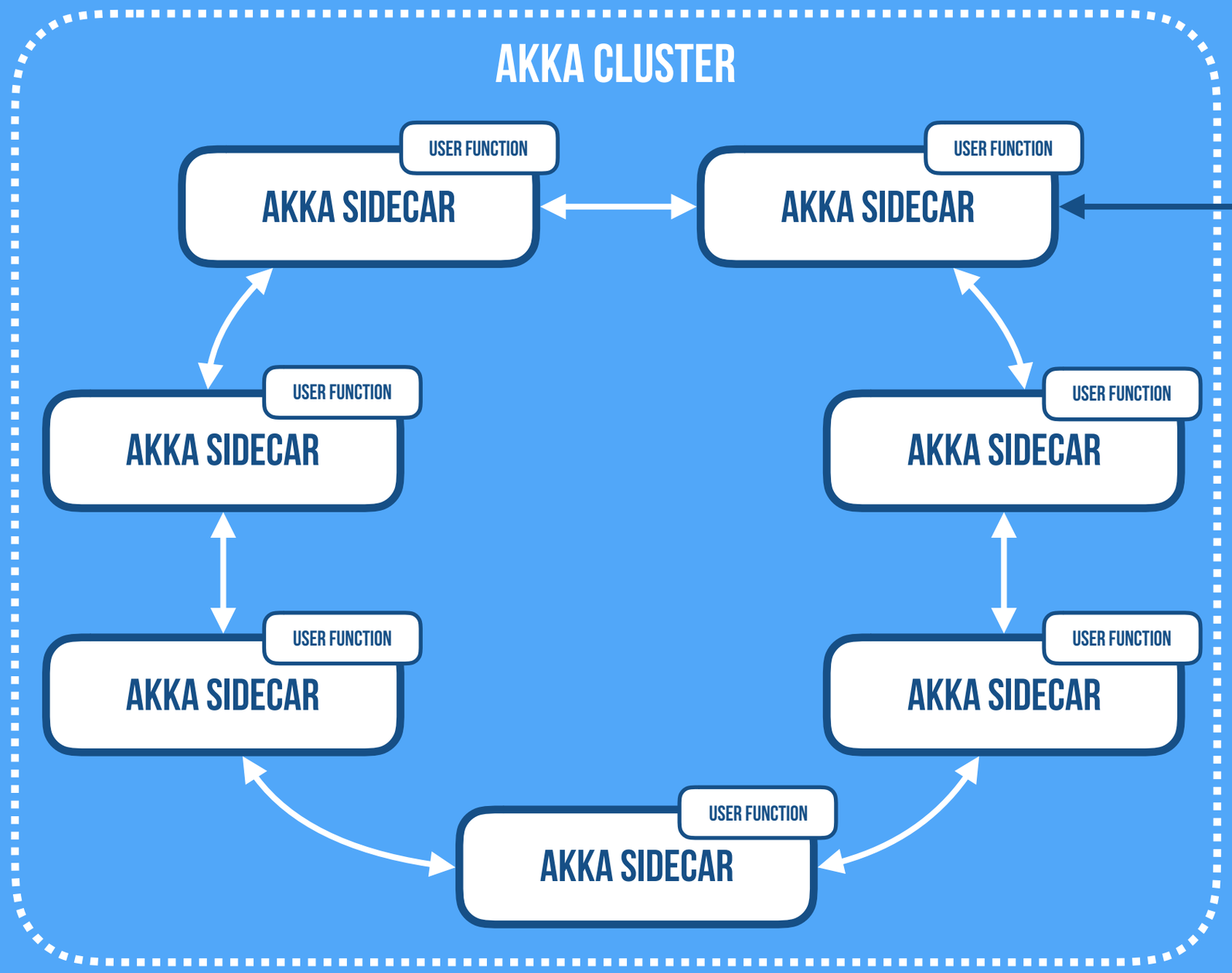


- Actor-based Distributed Runtime
- Decentralized Masterless P2P
- Epidemic Gossiping, Self-healing

- State Sharding & Routing on Entity Key

https://akka.io

AKKA CLUSTER STATE MANAGEMENT

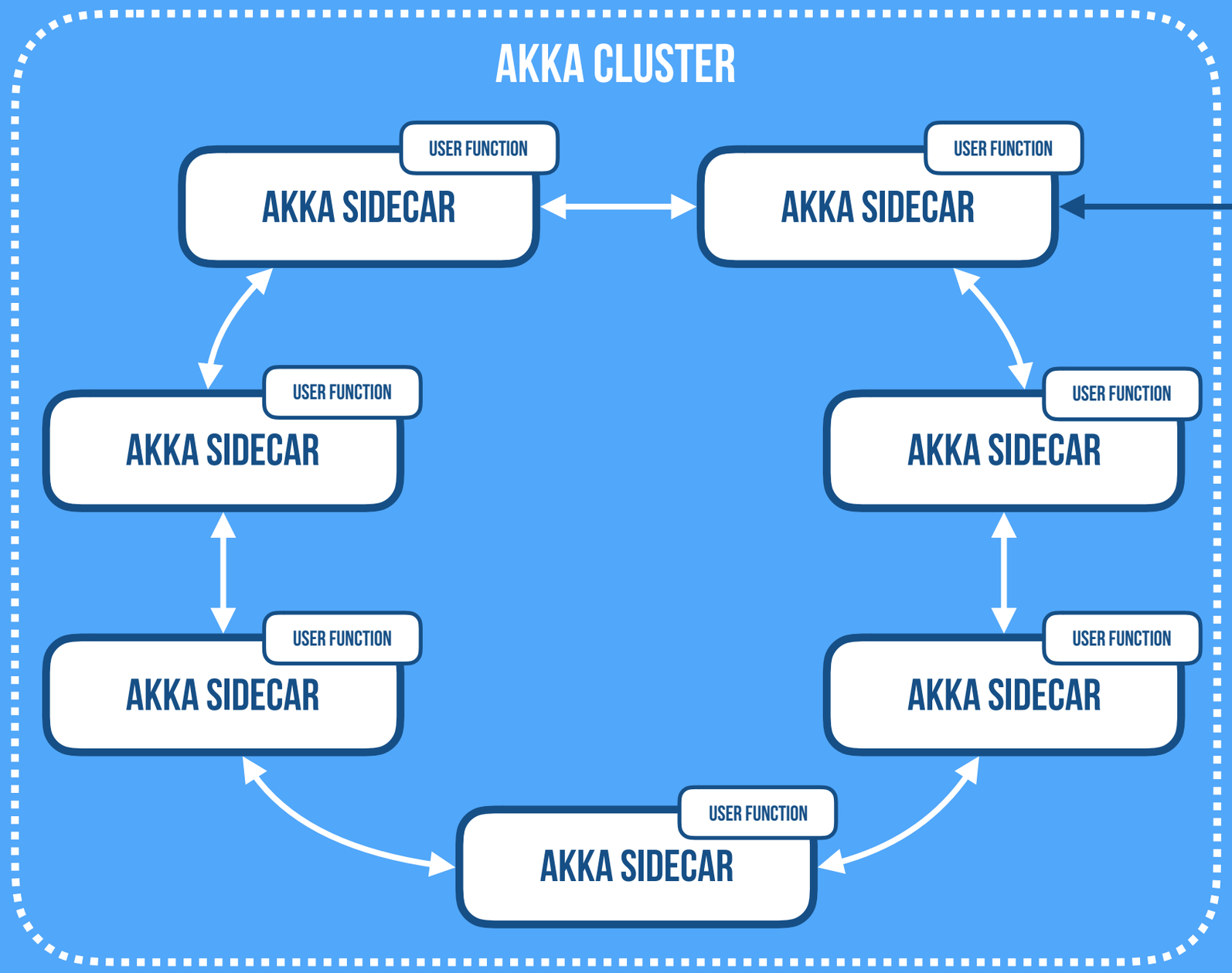


- Actor-based Distributed Runtime
- Decentralized Masterless P2P
- Epidemic Gossiping, Self-healing

- State Sharding & Routing on Entity Key

https://akka.io

AKKA CLUSTER STATE MANAGEMENT

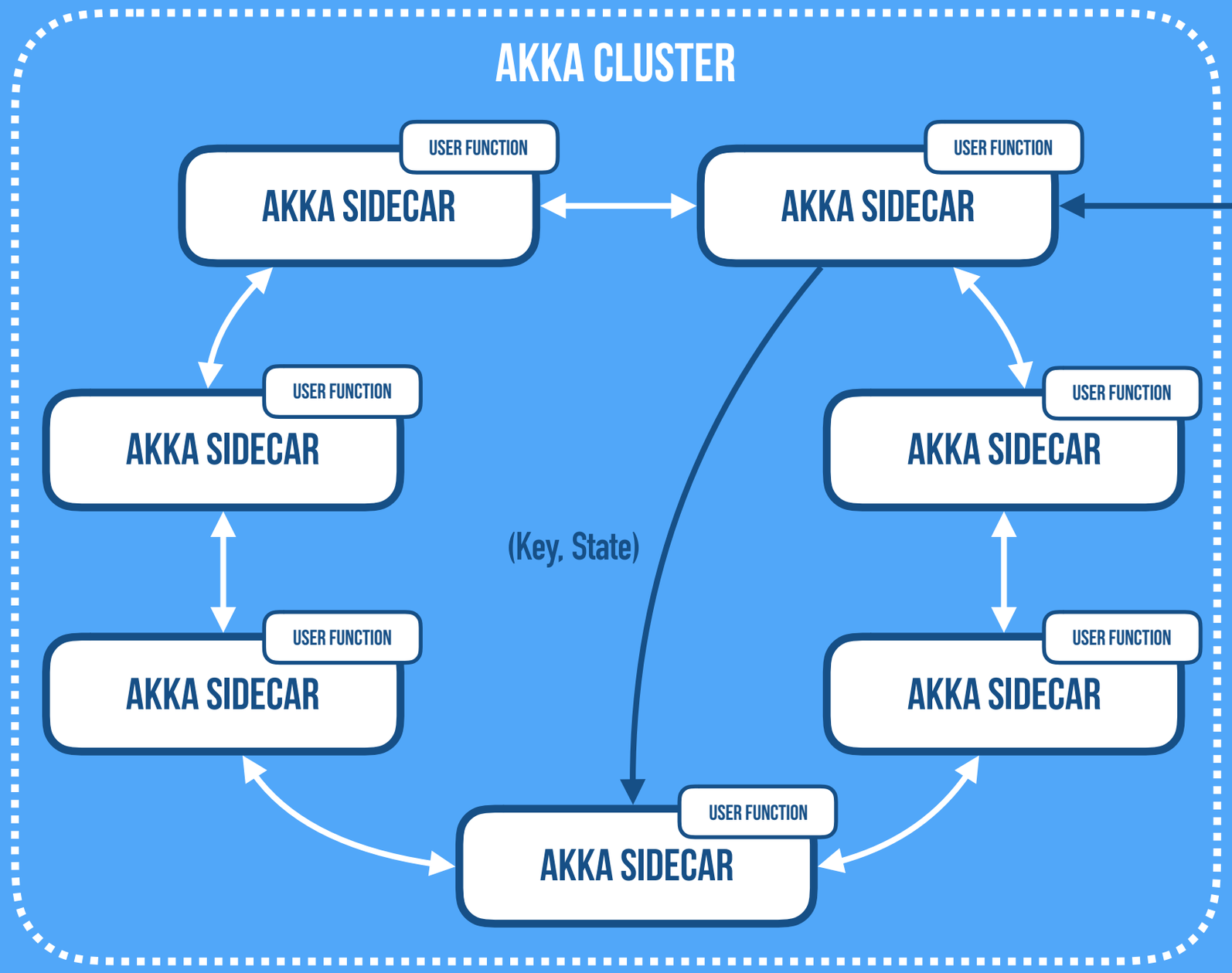


- Actor-based Distributed Runtime
- Decentralized Masterless P2P
- Epidemic Gossiping, Self-healing

- State Sharding & Routing on Entity Key
- Forwarding of Requests (if needed)

https://akka.io

AKKA CLUSTER STATE MANAGEMENT

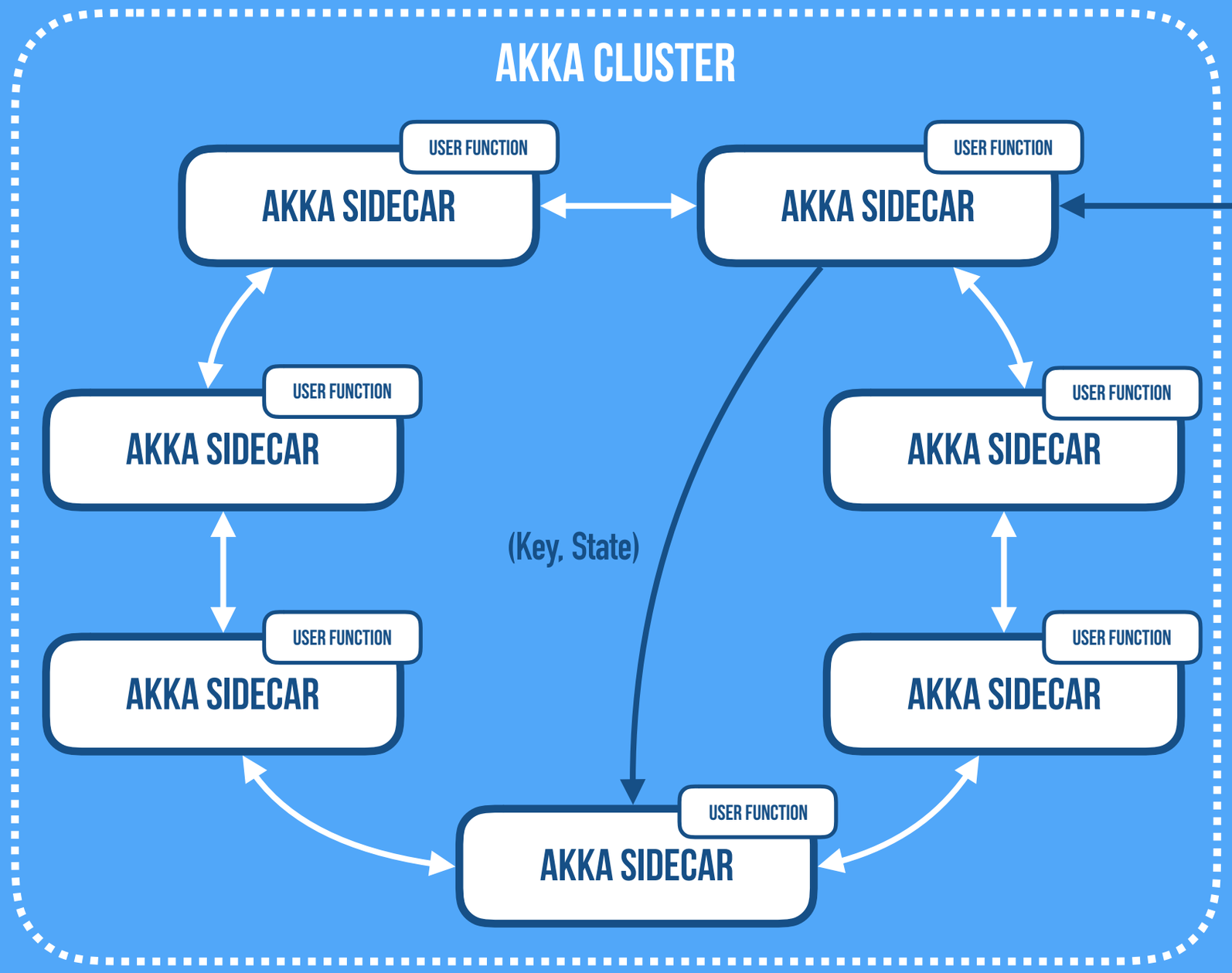


- Actor-based Distributed Runtime
- Decentralized Masterless P2P
- Epidemic Gossiping, Self-healing

- State Sharding & Routing on Entity Key
- Forwarding of Requests (if needed)

https://akka.io

AKKA CLUSTER STATE MANAGEMENT

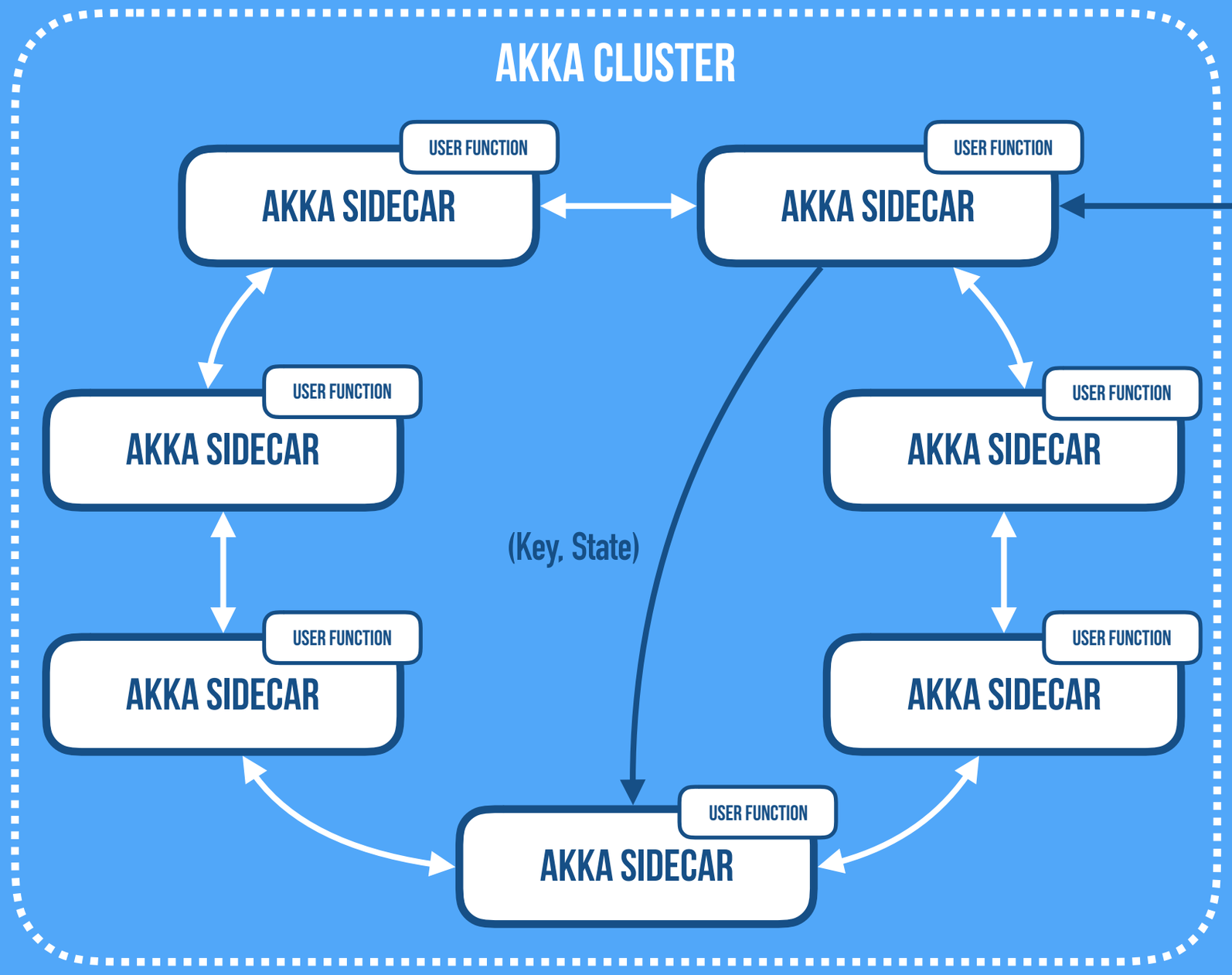


- Actor-based Distributed Runtime
- Decentralized Masterless P2P
- Epidemic Gossiping, Self-healing

- State Sharding & Routing on Entity Key
- Forwarding of Requests (if needed)
- Co-Location of State & Processing

https://akka.io

AKKA CLUSTER STATE MANAGEMENT

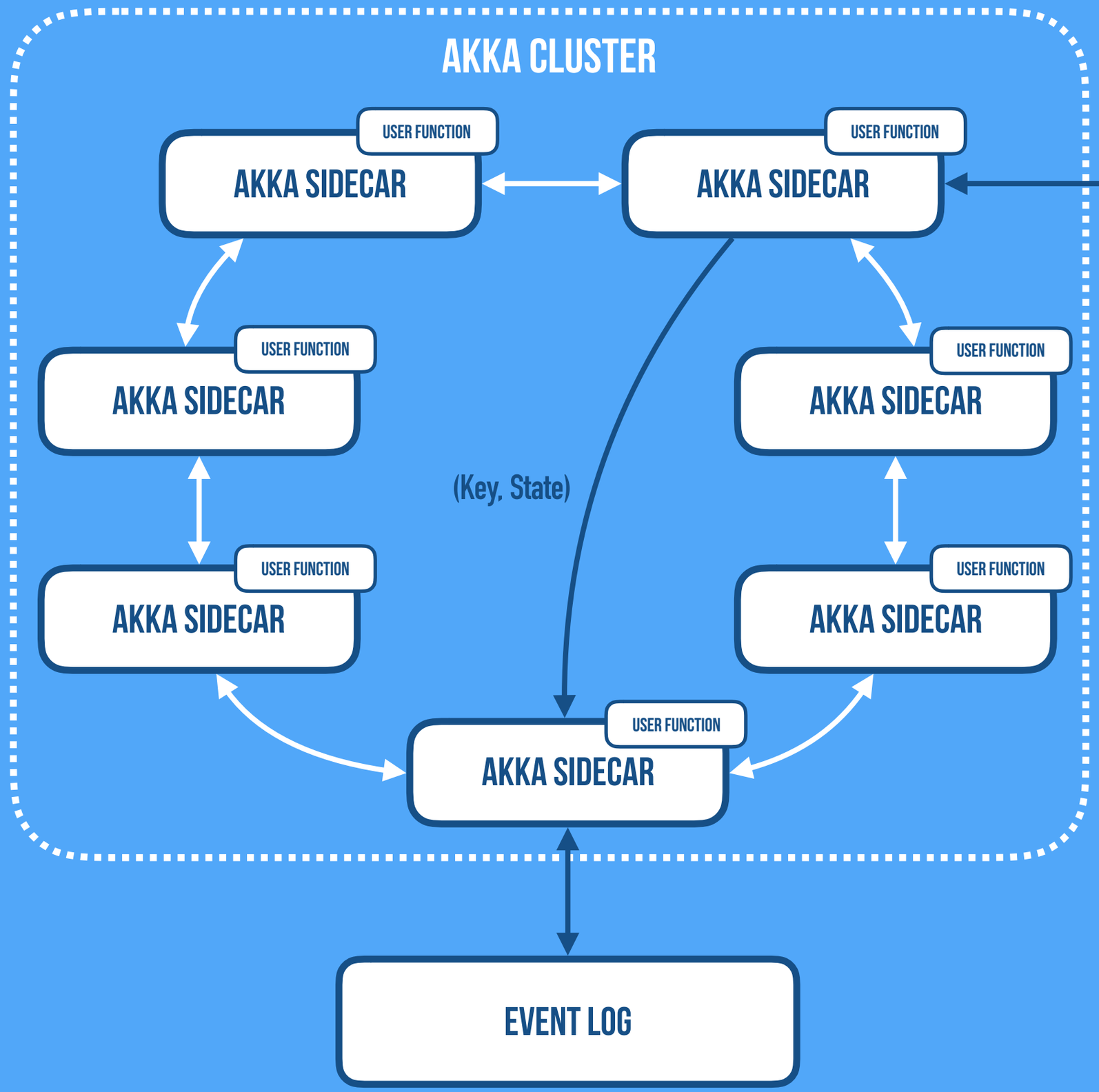


- Actor-based Distributed Runtime
- Decentralized Masterless P2P
- Epidemic Gossiping, Self-healing

- State Sharding & Routing on Entity Key
- Forwarding of Requests (if needed)
- Co-Location of State & Processing
- Backed by Event Log

https://akka.io

AKKA CLUSTER STATE MANAGEMENT

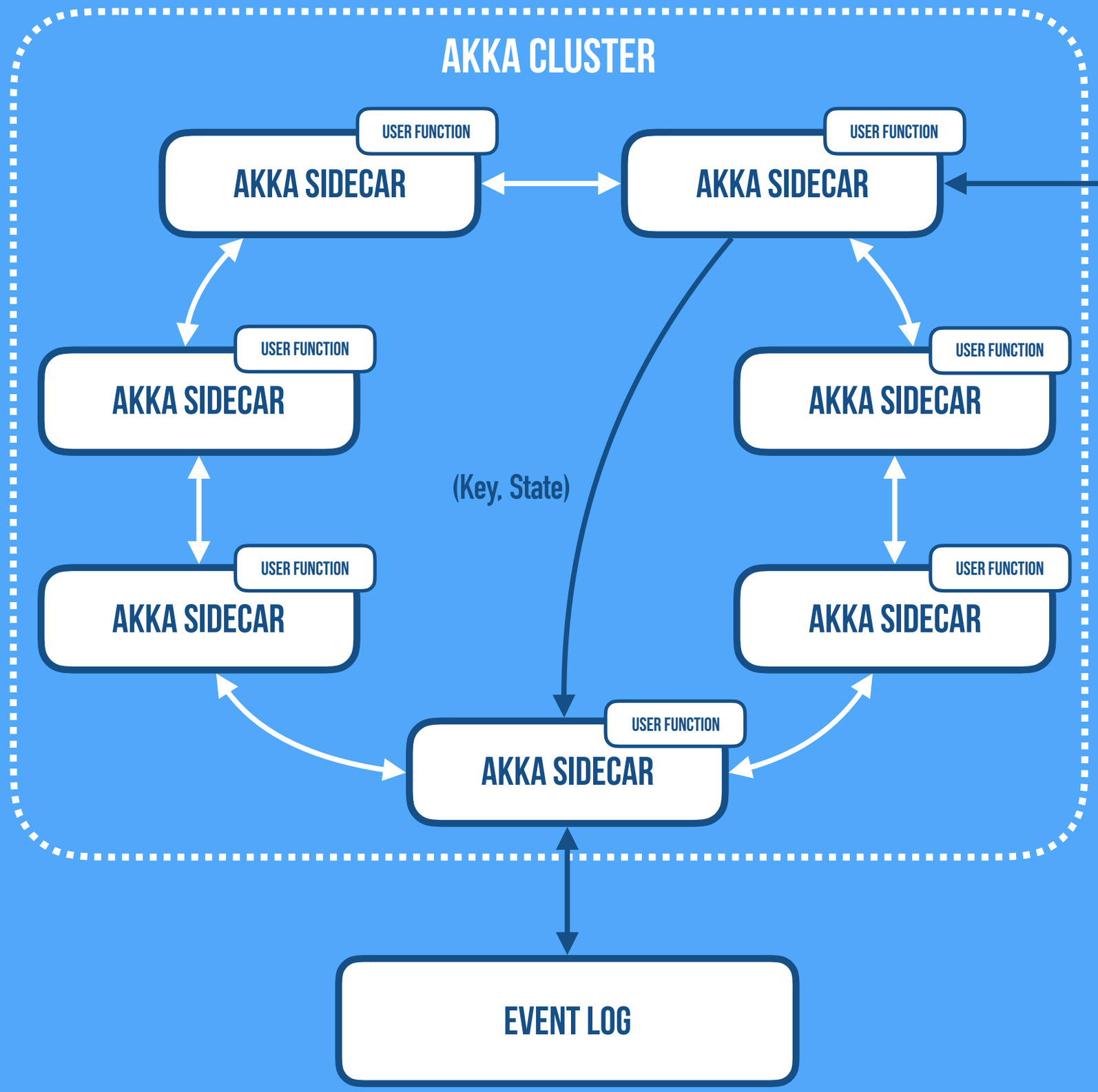


- Actor-based Distributed Runtime
- Decentralized Masterless P2P
- Epidemic Gossiping, Self-healing

- State Sharding & Routing on Entity Key
- Forwarding of Requests (if needed)
- Co-Location of State & Processing
- Backed by Event Log

https://akka.io

AKKA CLUSTER STATE MANAGEMENT

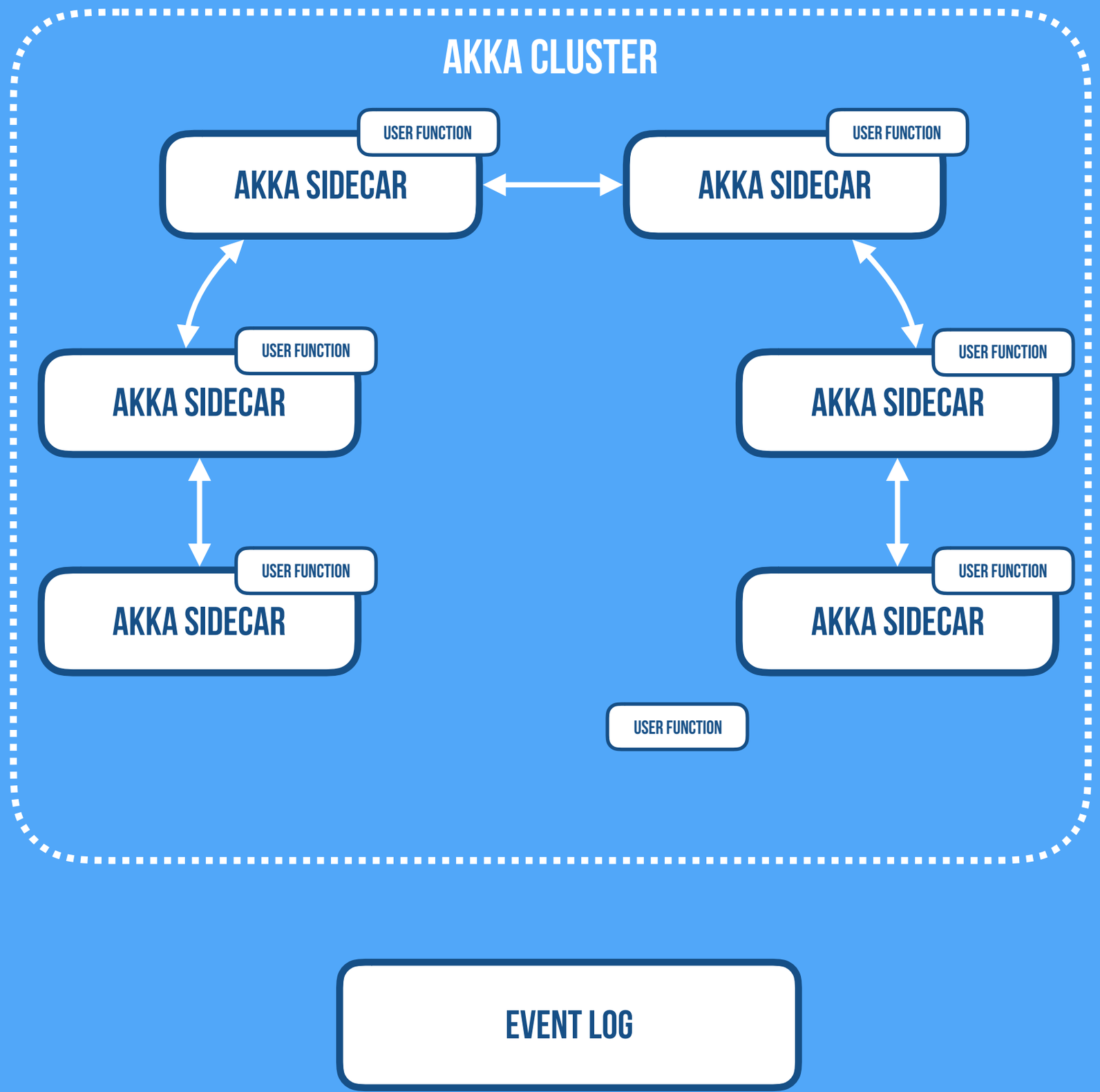


- Actor-based Distributed Runtime
- Decentralized Masterless P2P
- Epidemic Gossiping, Self-healing

- State Sharding & Routing on Entity Key
- Forwarding of Requests (if needed)
- Co-Location of State & Processing
- Backed by Event Log
- Automatic Failover, Rehydration, and Rebalancing

https://akka.io

AKKA CLUSTER STATE MANAGEMENT

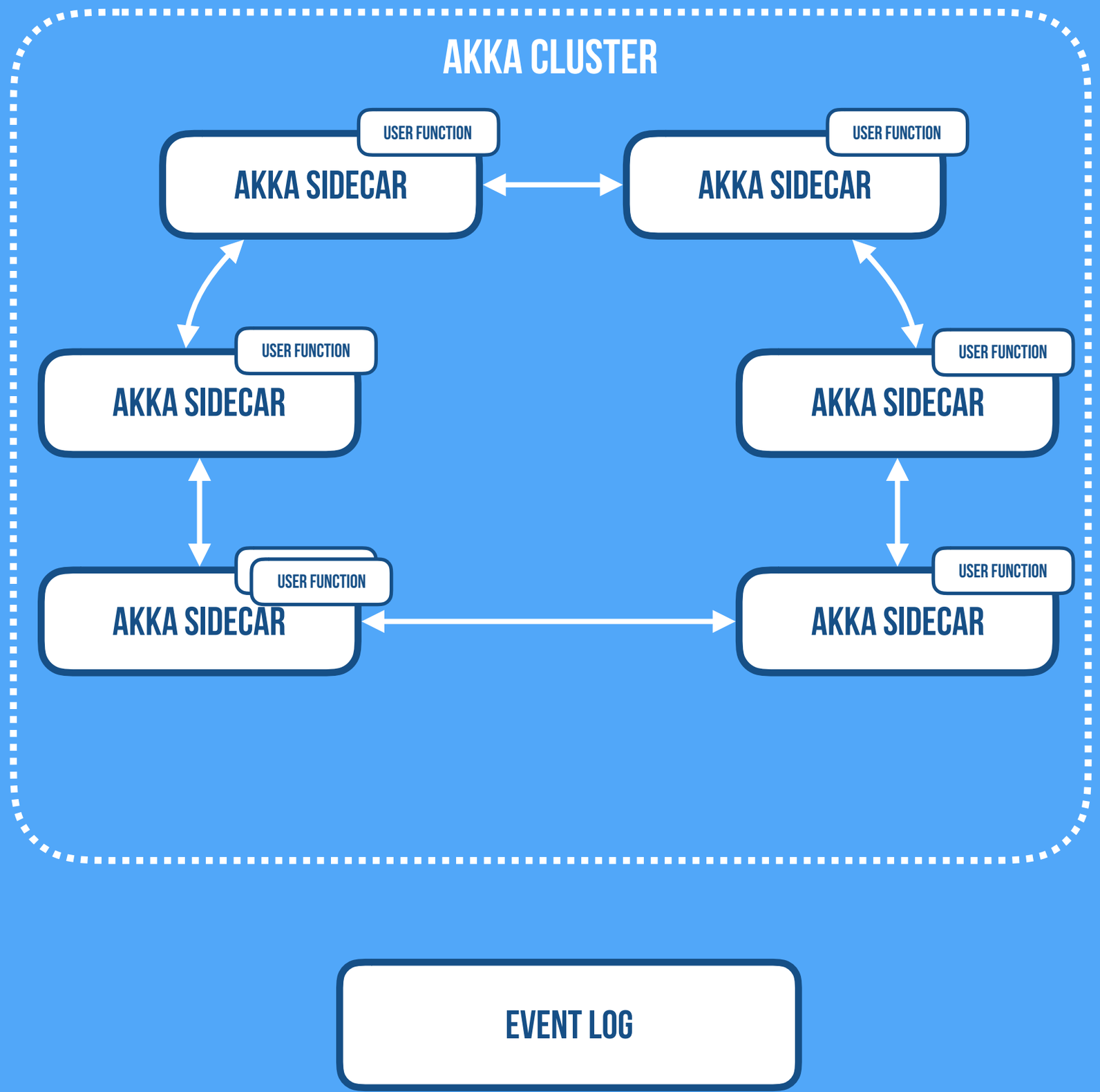


- Actor-based Distributed Runtime
- Decentralized Masterless P2P
- Epidemic Gossiping, Self-healing

- State Sharding & Routing on Entity Key
- Forwarding of Requests (if needed)
- Co-Location of State & Processing
- Backed by Event Log
- Automatic Failover, Rehydration, and Rebalancing

https://akka.io

AKKA CLUSTER STATE MANAGEMENT

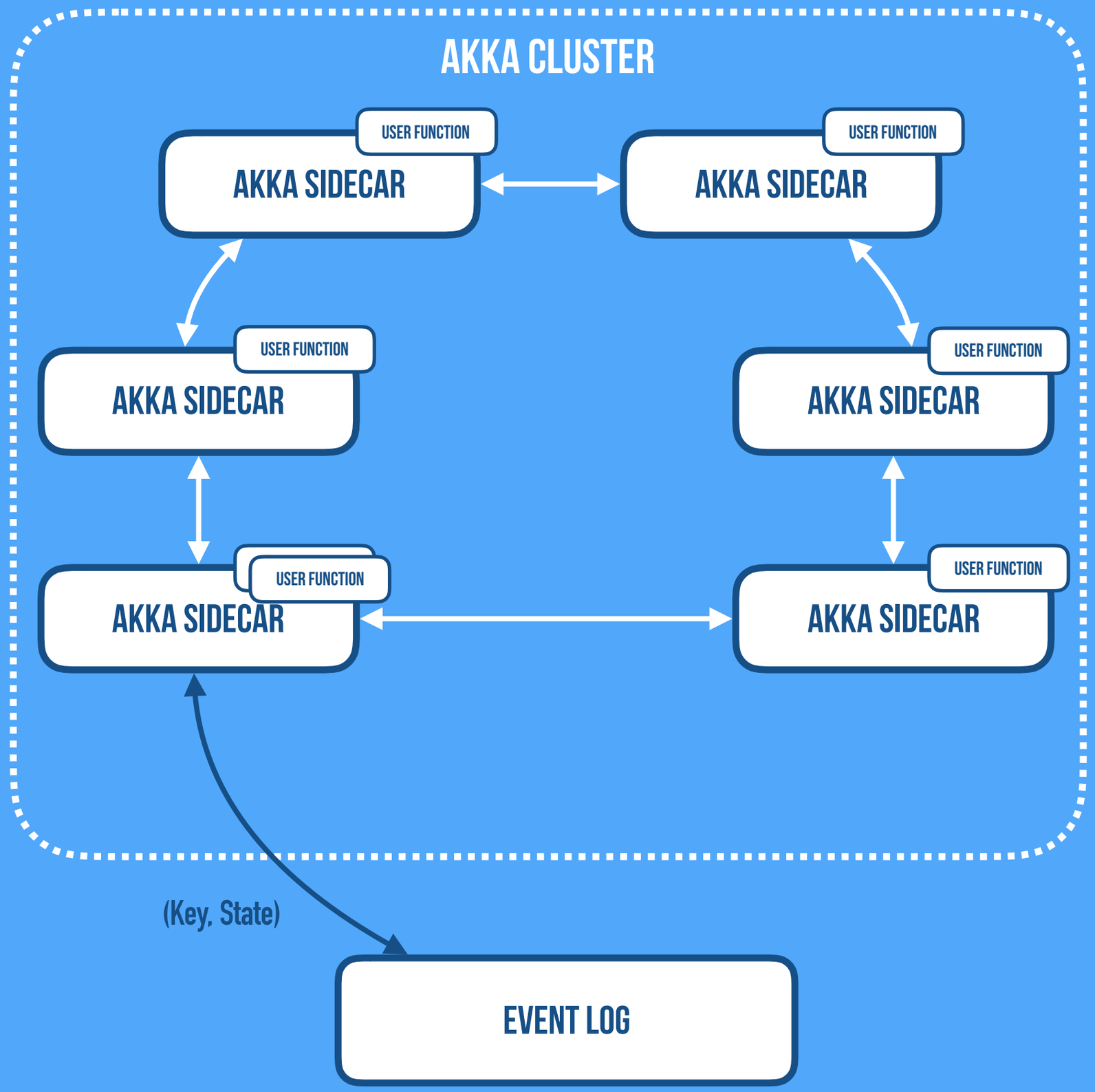


- Actor-based Distributed Runtime
- Decentralized Masterless P2P
- Epidemic Gossiping, Self-healing

- State Sharding & Routing on Entity Key
- Forwarding of Requests (if needed)
- Co-Location of State & Processing
- Backed by Event Log
- Automatic Failover, Rehydration, and Rebalancing

https://akka.io

AKKA CLUSTER STATE MANAGEMENT

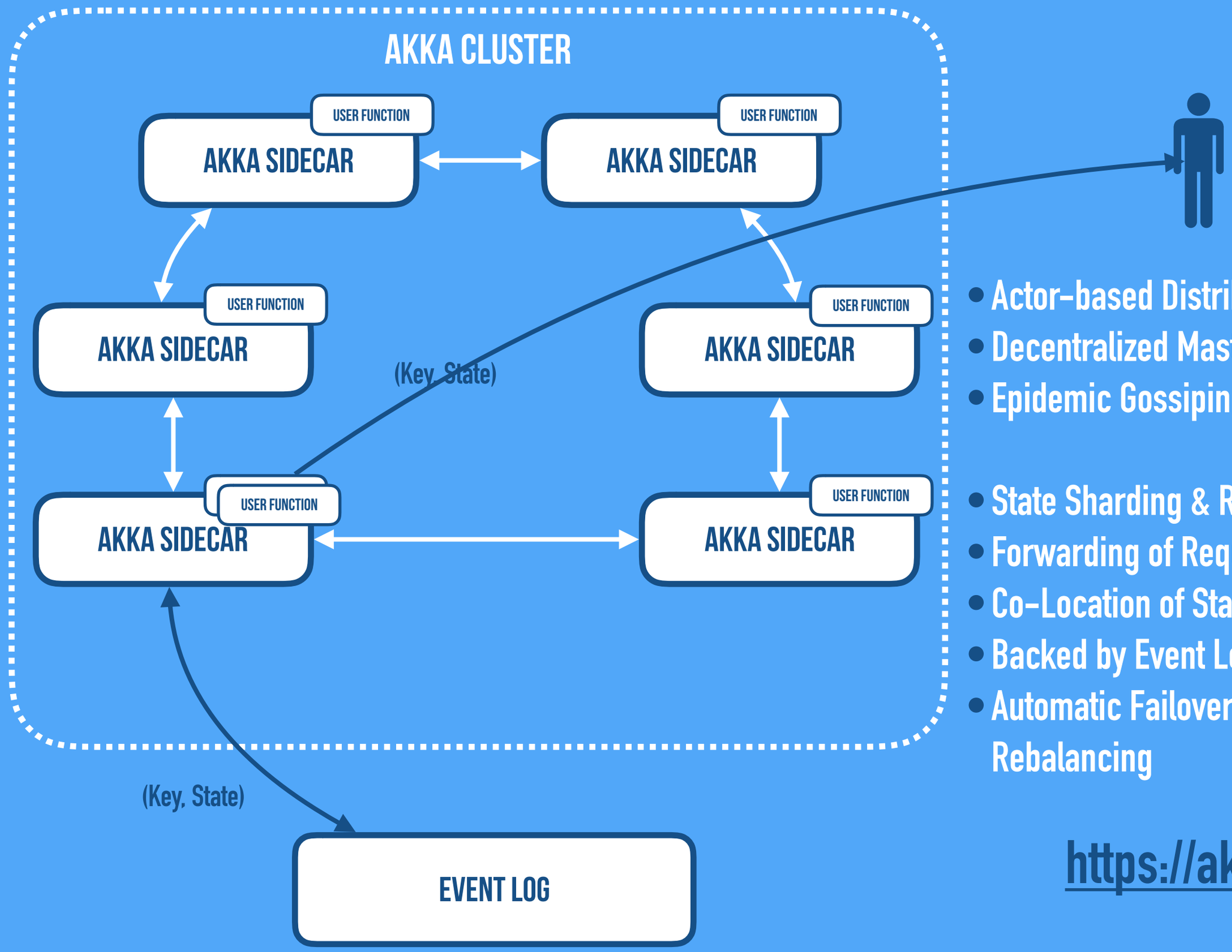


- Actor-based Distributed Runtime
- Decentralized Masterless P2P
- Epidemic Gossiping, Self-healing

- State Sharding & Routing on Entity Key
- Forwarding of Requests (if needed)
- Co-Location of State & Processing
- Backed by Event Log
- Automatic Failover, Rehydration, and Rebalancing

https://akka.io

AKKA CLUSTER STATE MANAGEMENT

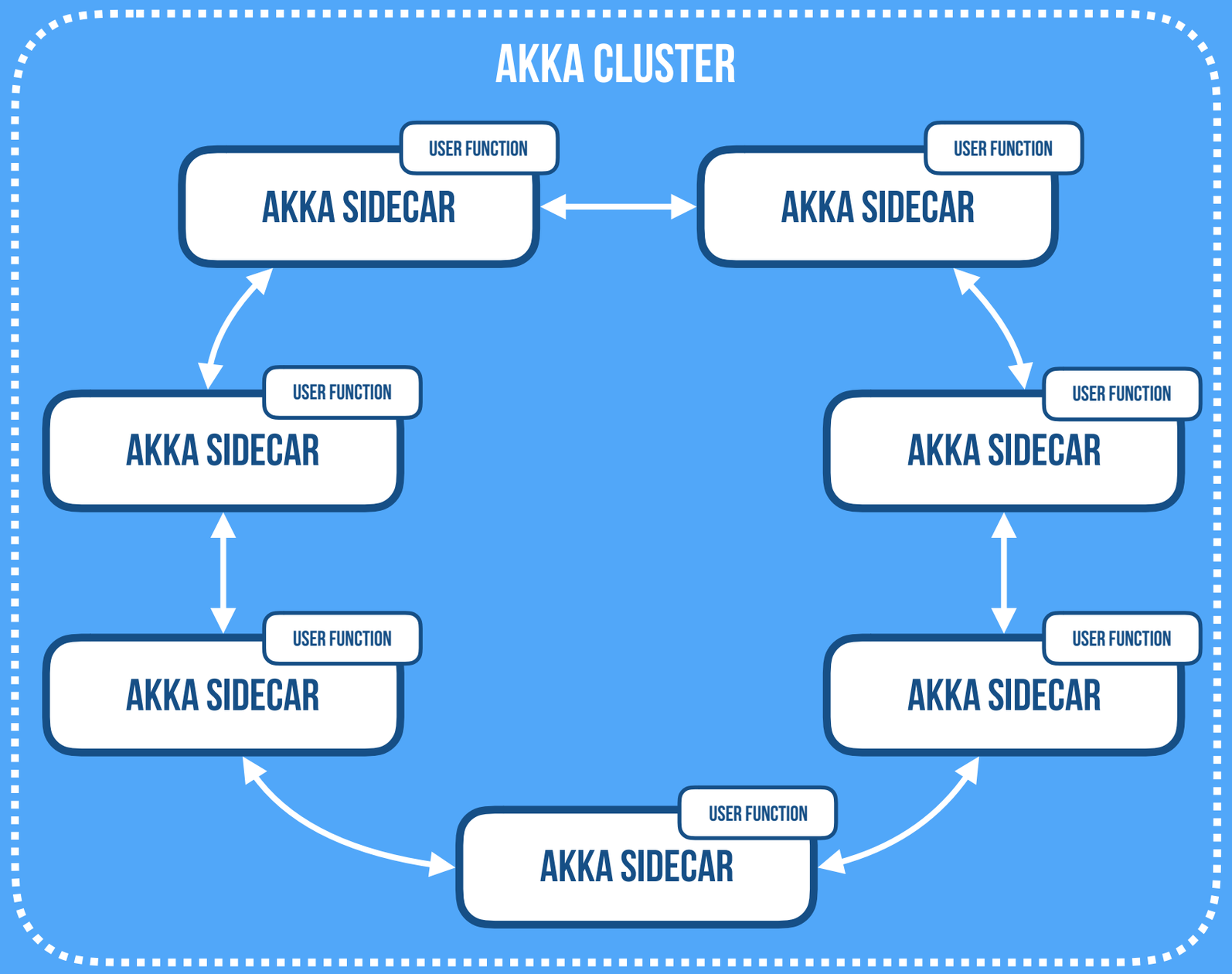


- Actor-based Distributed Runtime
- Decentralized Masterless P2P
- Epidemic Gossiping, Self-healing

- State Sharding & Routing on Entity Key
- Forwarding of Requests (if needed)
- Co-Location of State & Processing
- Backed by Event Log
- Automatic Failover, Rehydration, and Rebalancing

https://akka.io

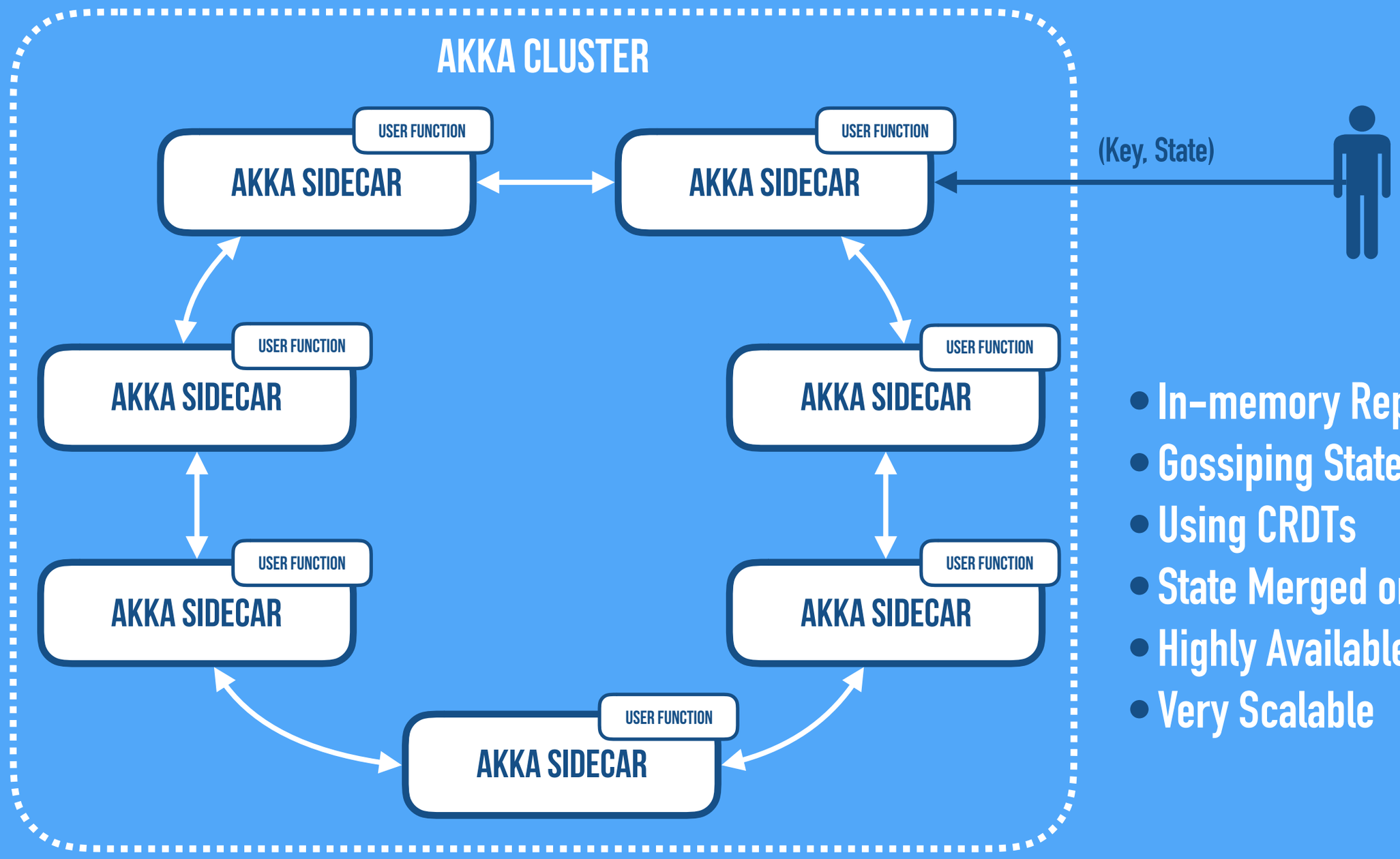
AKKA CLUSTER STATE MANAGEMENT



- In-memory Replication of State
- Gossiping State Changes
- Using CRDTs
- State Merged on Local Node
- Highly Available (N Replicas)
- Very Scalable

https://akka.io

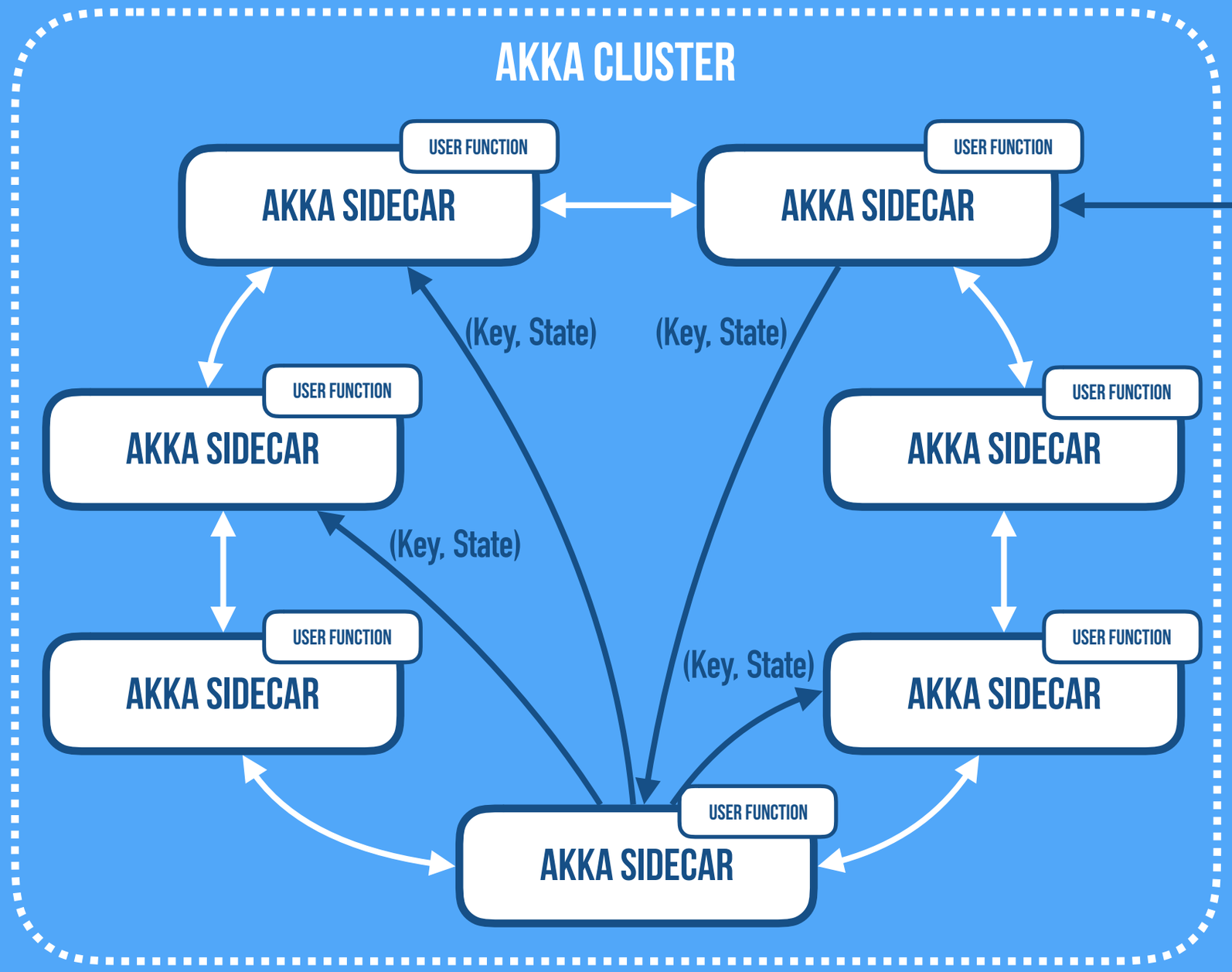
AKKA CLUSTER STATE MANAGEMENT



- In-memory Replication of State
- Gossiping State Changes
- Using CRDTs
- State Merged on Local Node
- Highly Available (N Replicas)
- Very Scalable

https://akka.io

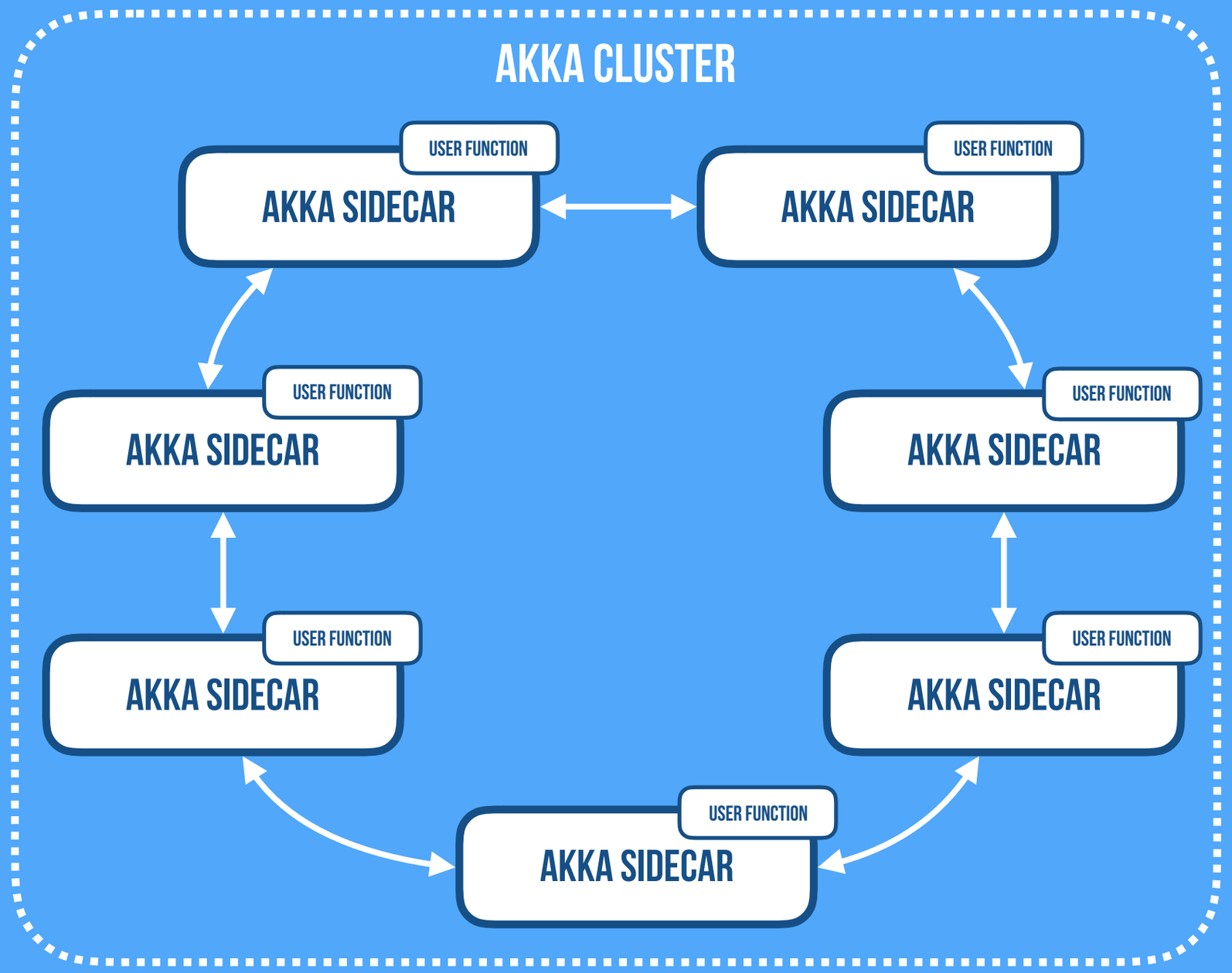
AKKA CLUSTER STATE MANAGEMENT



- In-memory Replication of State
- Gossiping State Changes
- Using CRDTs
- State Merged on Local Node
- Highly Available (N Replicas)
- Very Scalable

https://akka.io

AKKA CLUSTER STATE MANAGEMENT



- In-memory Replication of State
- Gossiping State Changes
- Using CRDTs
- State Merged on Local Node
- Highly Available (N Replicas)
- Very Scalable

https://akka.io

Cloudstate Uses Better Models
For Distributed State

Cloudstate Uses Better Models For Distributed State

BATTLE-TESTED, YET CONSTRAINED, MODELS LIKE:

Cloudstate Uses Better Models For Distributed State

BATTLE-TESTED, YET CONSTRAINED, MODELS LIKE:

- Event Sourcing

Cloudstate Uses Better Models For Distributed State

BATTLE-TESTED, YET CONSTRAINED, MODELS LIKE:

Event Sourcing

CRDTs

Cloudstate Uses Better Models For Distributed State

BATTLE-TESTED, YET CONSTRAINED, MODELS LIKE:

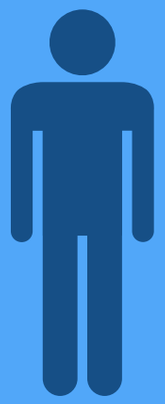
Event Sourcing

CRDTs

Key Value

Event Sourced Entities

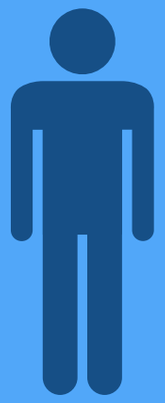
HAPPY PATH



Event Sourced Entities

HAPPY PATH

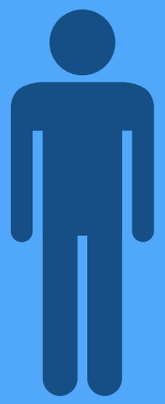
Event Sourced Entities



COMMAND

HAPPY PATH

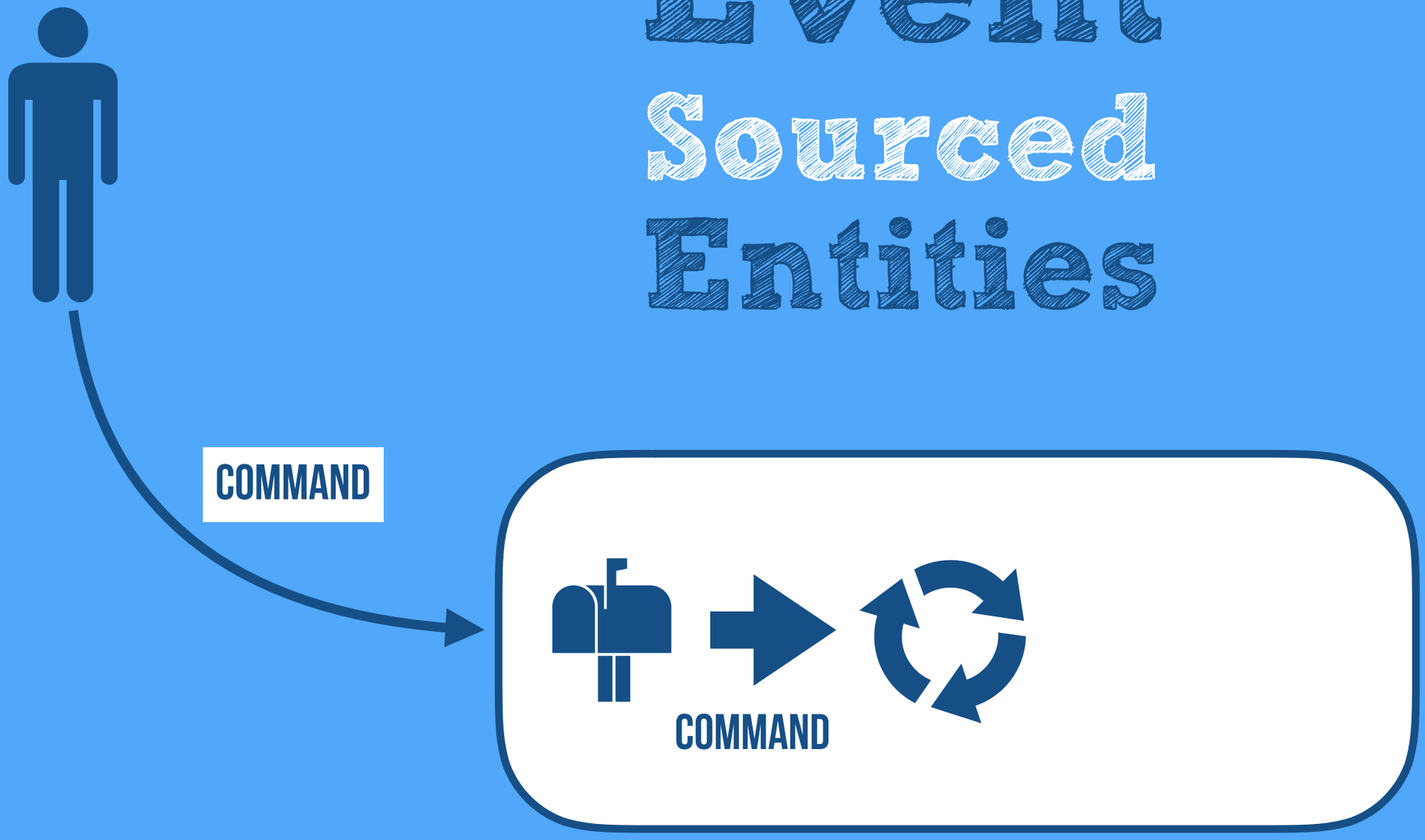
Event Sourced Entities



COMMAND

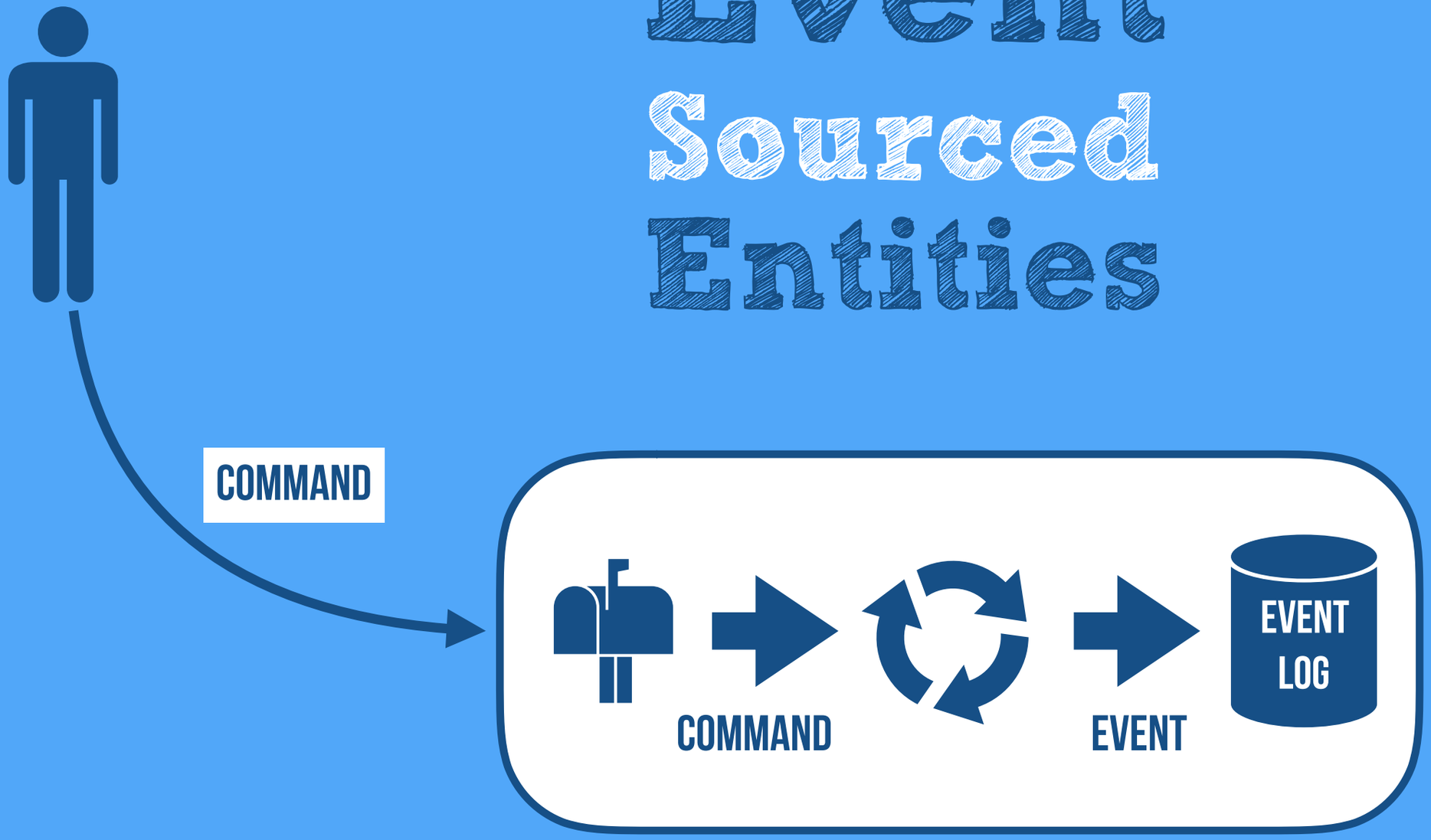
HAPPY PATH

Event Sourced Entities



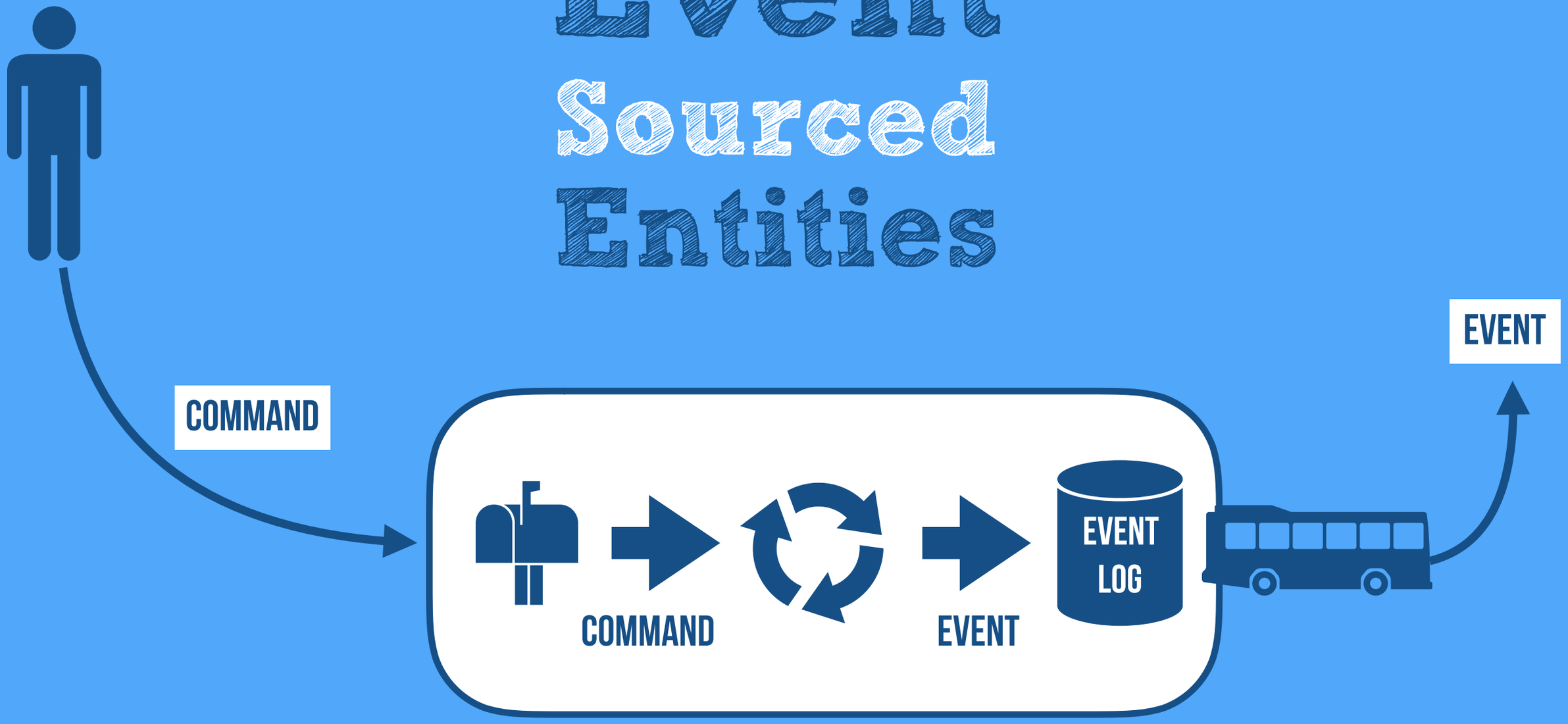
HAPPY PATH

Event Sourced Entities



HAPPY PATH

Event Sourced Entities



HAPPY PATH

Event Sourced Entities

HAPPY PATH

Event Sourced Entities

HAPPY PATH

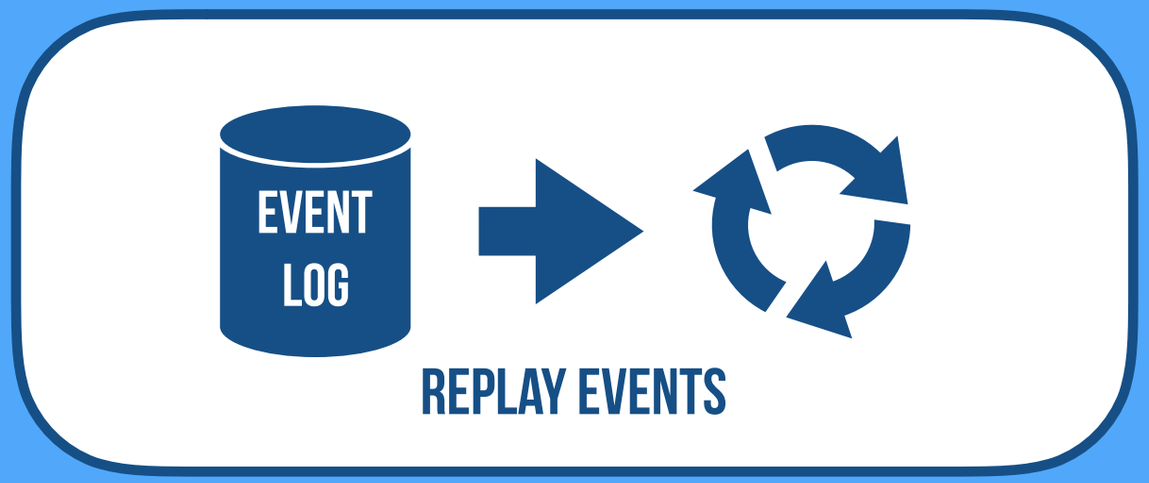
Event Sourced Entities

SAD PATH, RECOVER FROM FAILURE

Event Sourced Entities

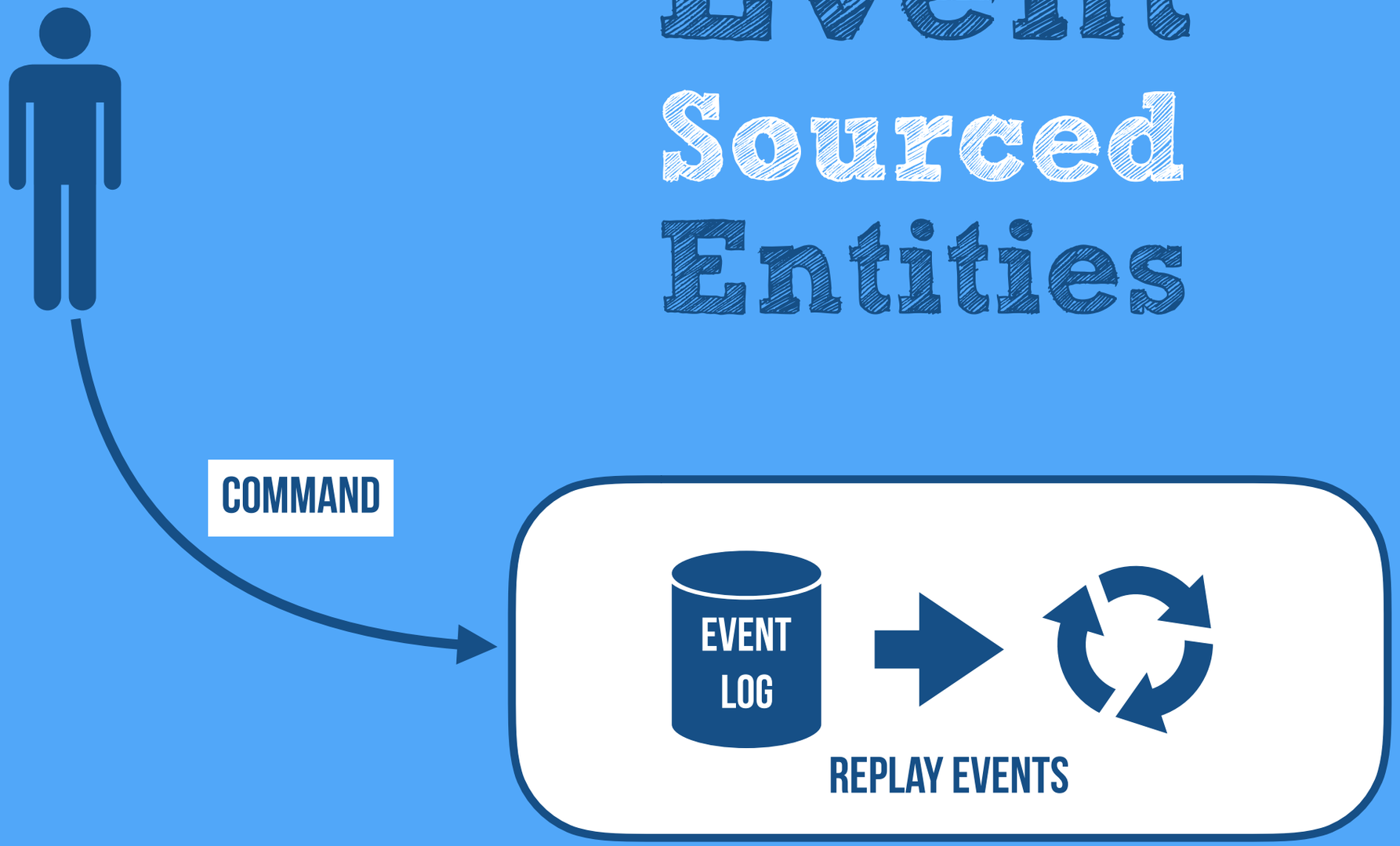
SAD PATH, RECOVER FROM FAILURE

Event Sourced Entities



SAD PATH, RECOVER FROM FAILURE

Event Sourced Entities



SAD PATH, RECOVER FROM FAILURE

Benefits of Event Sourcing

Benefits of Event Sourcing

- One single SOURCE OF TRUTH with ALL HISTORY

Benefits of Event Sourcing

- One single SOURCE OF TRUTH with ALL HISTORY
- Allows for MEMORY IMAGE (Durable In-Memory State)

Benefits of Event Sourcing

- ✳ One single SOURCE OF TRUTH with ALL HISTORY
- ✳ Allows for MEMORY IMAGE (Durable In-Memory State)
- ✳ Avoids the OBJECT-RELATIONAL MISMATCH

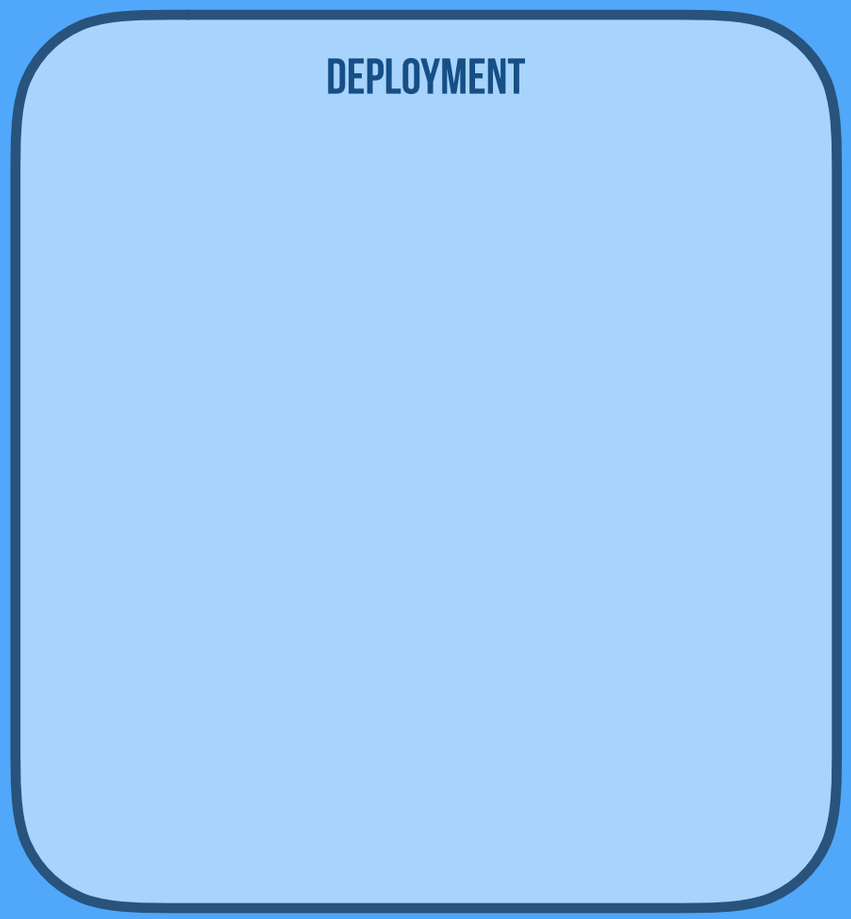
Benefits of Event Sourcing

- One single SOURCE OF TRUTH with ALL HISTORY
- Allows for MEMORY IMAGE (Durable In-Memory State)
- Avoids the OBJECT-RELATIONAL MISMATCH
- Allows others to SUBSCRIBE TO STATE CHANGES

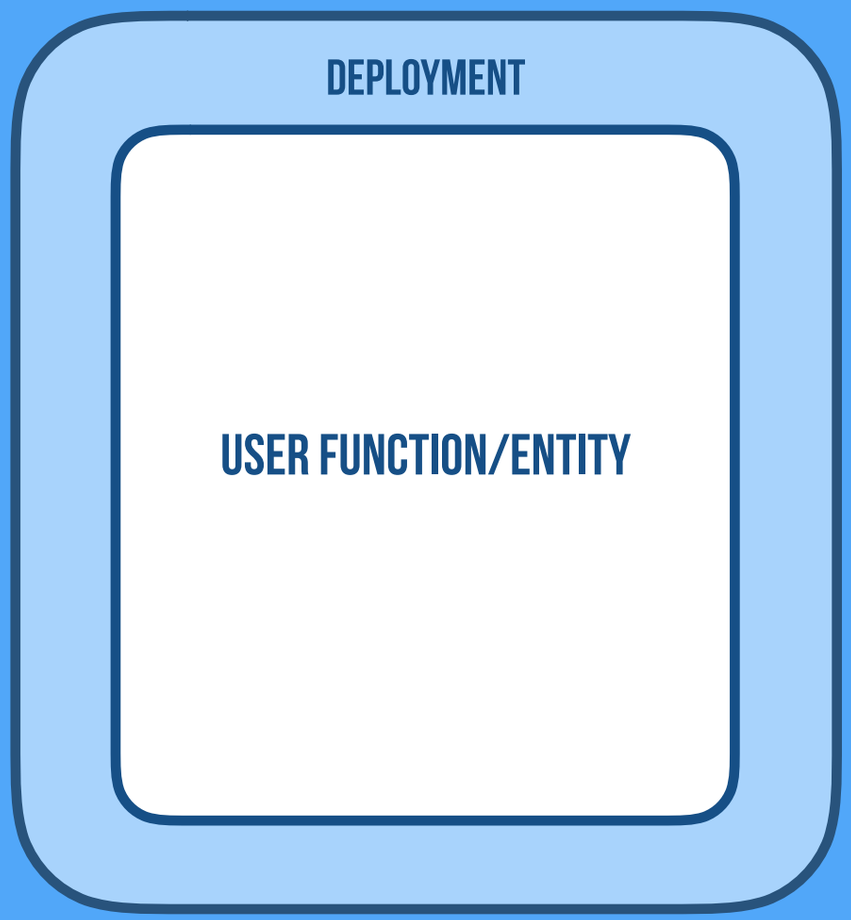
Benefits of Event Sourcing

- ✳ One single SOURCE OF TRUTH with ALL HISTORY
- ✳ Allows for MEMORY IMAGE (Durable In-Memory State)
- ✳ Avoids the OBJECT-RELATIONAL MISMATCH
- ✳ Allows others to SUBSCRIBE TO STATE CHANGES
- ✳ Has good MECHANICAL SYMPATHY (Single Writer Principle)

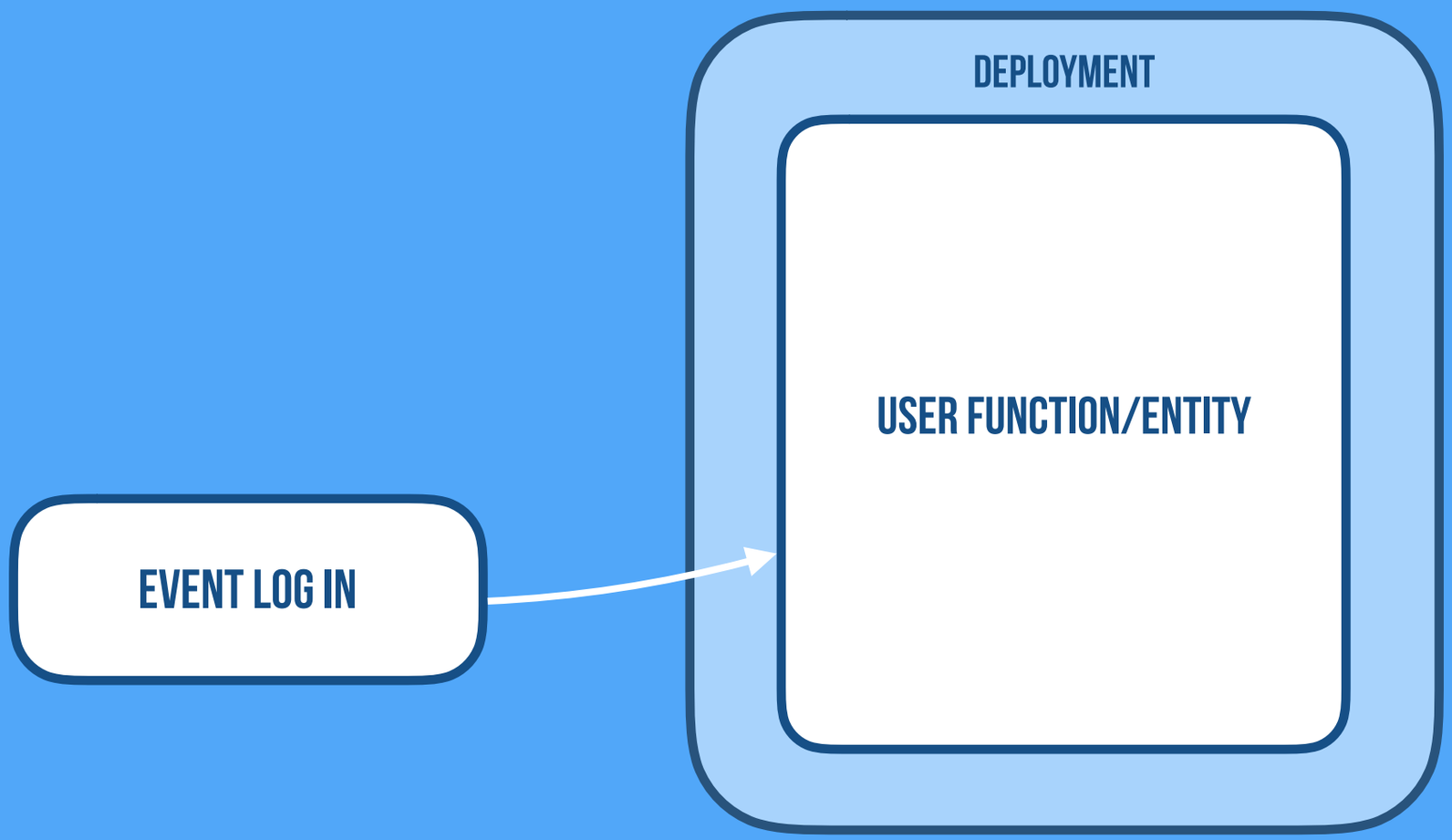
Serverless
Event Sourcing



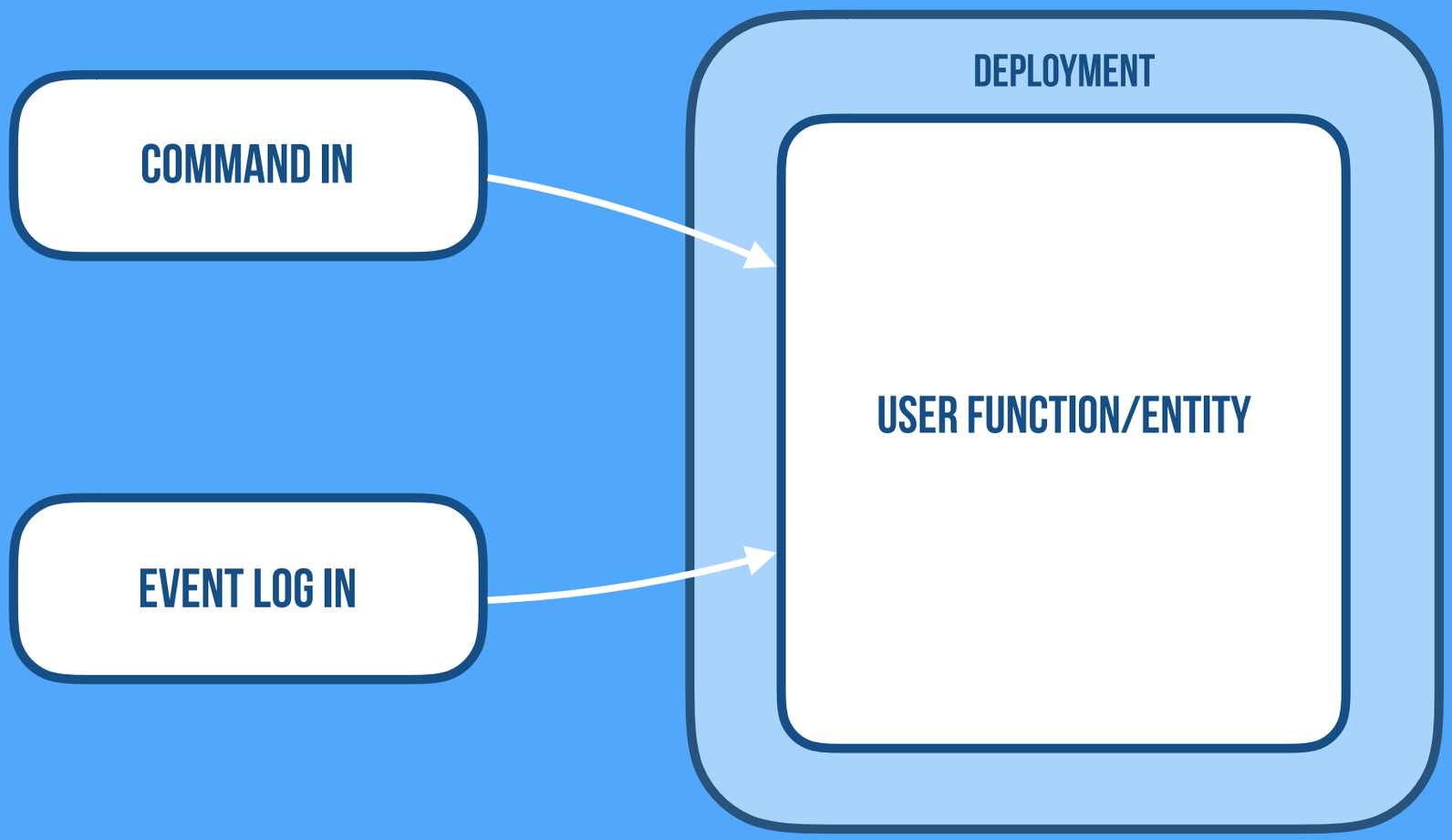
Serverless
Event Sourcing



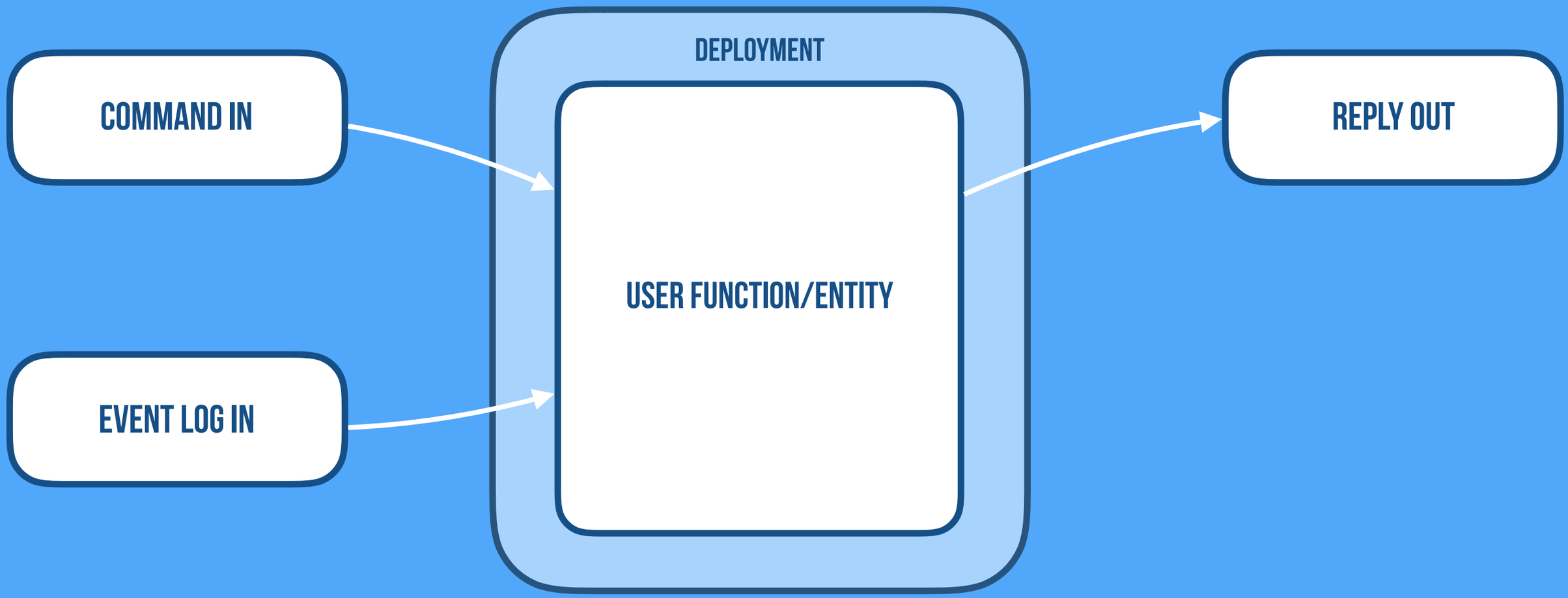
Serverless
Event Sourcing



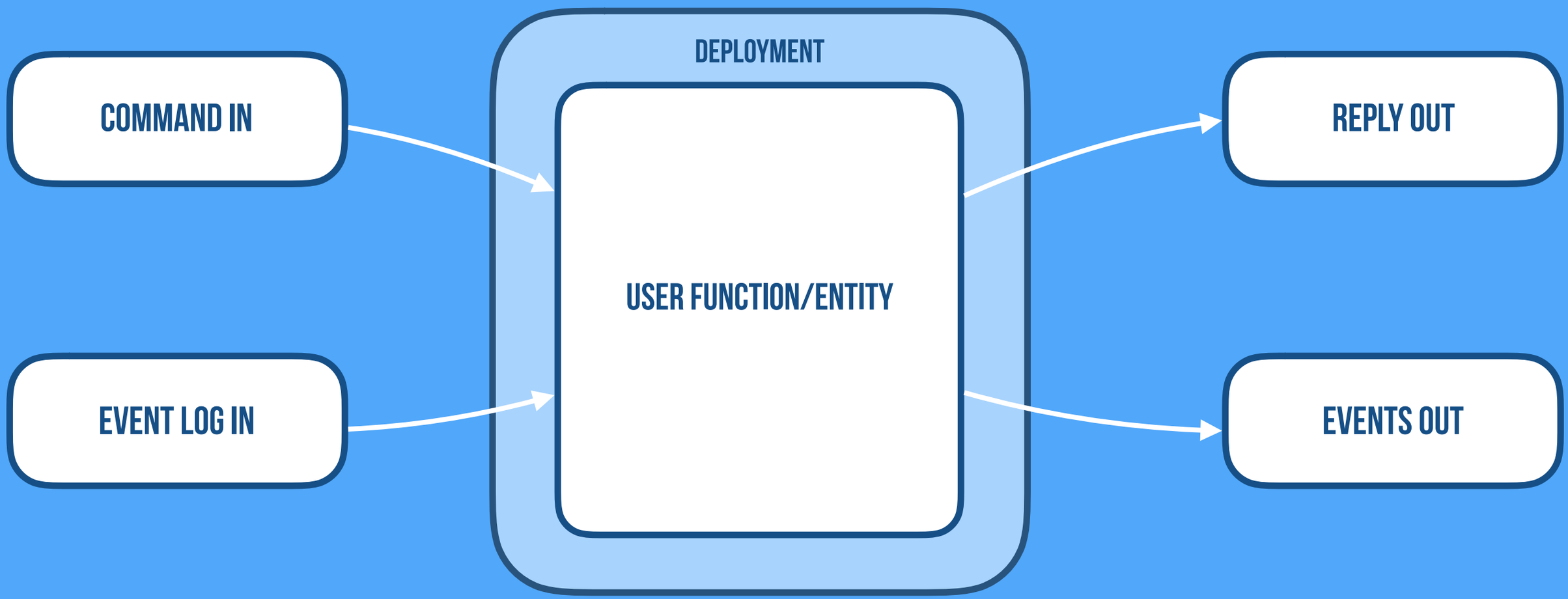
Serverless
Event Sourcing



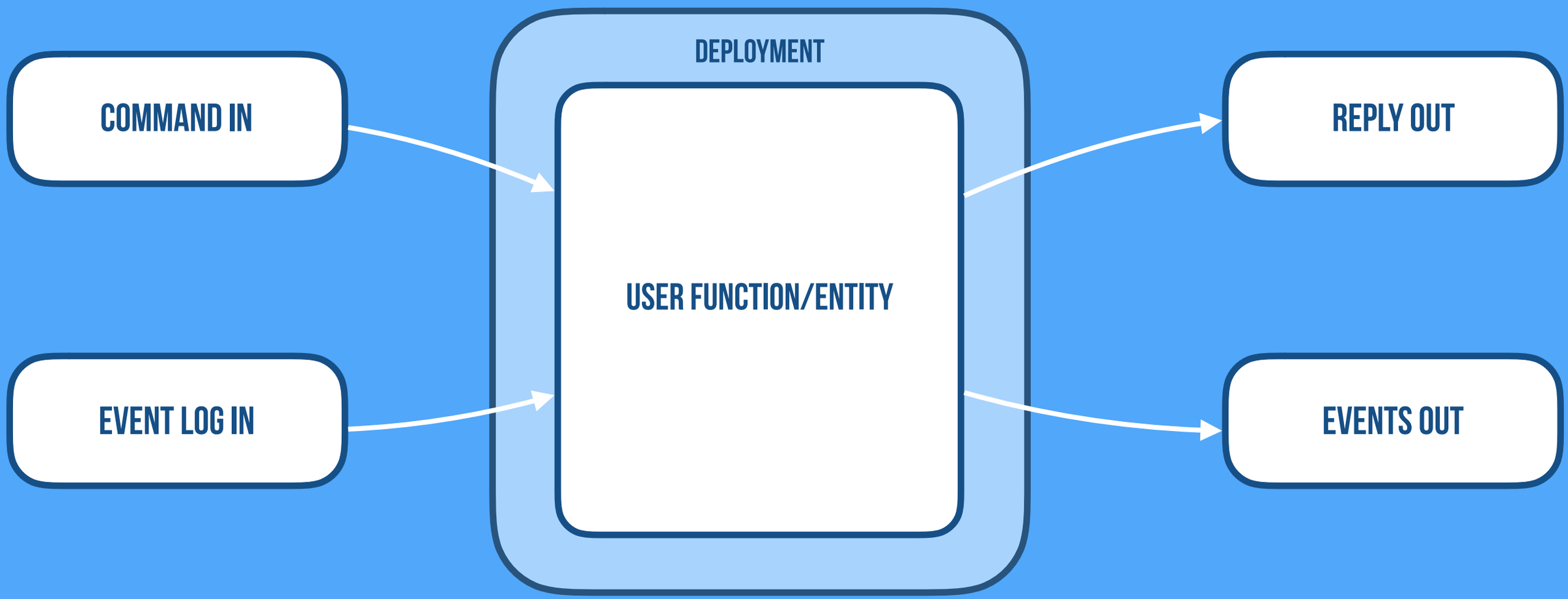
Serverless
Event Sourcing



Serverless
Event Sourcing



Serverless
Event Sourcing



CONFLICT-FREE REPLICATED DATA TYPES

Convergent & Commutative Replicated Data Types - Shapiro et. al. 2011

CONFLICT-FREE REPLICATED DATA TYPES

CRDT

Convergent & Commutative Replicated Data Types - Shapiro et. al. 2011

CONFLICT-FREE REPLICATED DATA TYPES

CRDT

Strong Eventual Consistency

Replicated & Decentralized

Highly Available & Very Scalable

Data Types Contain Resolution Logic

Always Converge Correctly

Convergent & Commutative Replicated Data Types - Shapiro et. al. 2011

CONFLICT-FREE REPLICATED DATA TYPES

CRDT

DATA TYPES

Counters

Registers

Sets

Maps

Graphs

(that all compose)

Strong Eventual Consistency

Replicated & Decentralized

Highly Available & Very Scalable

Data Types Contain Resolution Logic

Always Converge Correctly

Convergent & Commutative Replicated Data Types - Shapiro et. al. 2011

CRDTs are . . .

CRDTs are . . .

Associative

Batch-insensitive

(grouping doesn't matter)

$a+(b+c)=(a+b)+c$

CRDTs are . . .

Associative

Batch-insensitive

(grouping doesn't matter)

$a+(b+c)=(a+b)+c$

Commutative

Order-insensitive

(order doesn't matter)

$a+b=b+a$

CRDTs are . . .

Associative

Batch-insensitive

(grouping doesn't matter)

$a+(b+c)=(a+b)+c$

Commutative

Order-insensitive

(order doesn't matter)

$a+b=b+a$

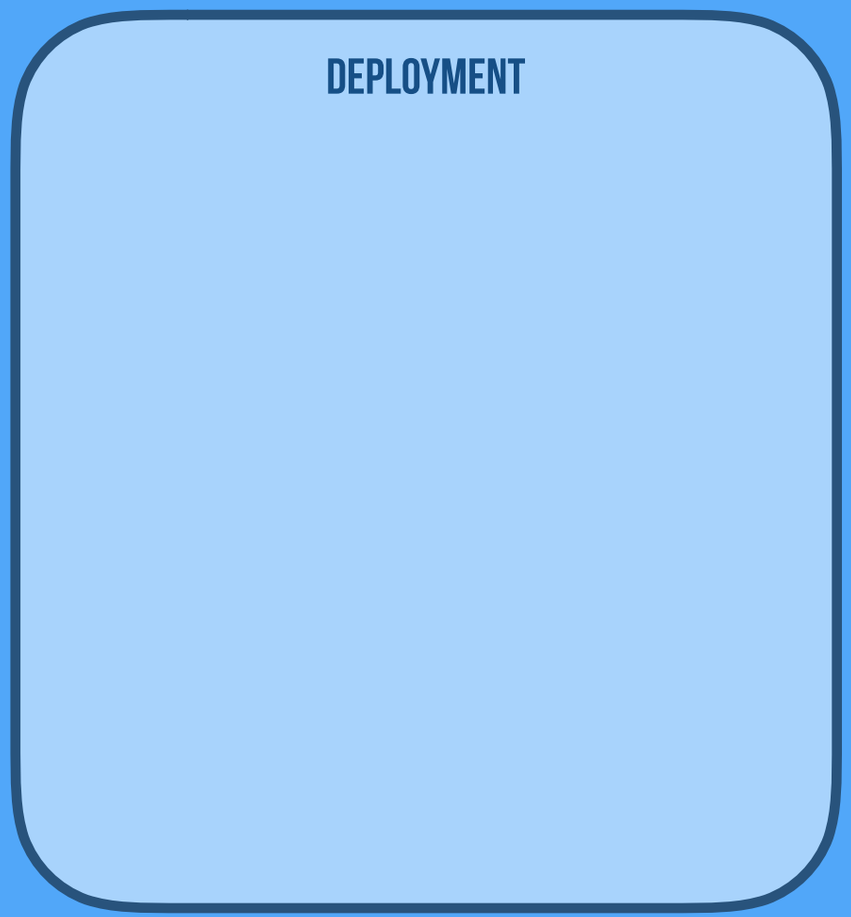
Idempotent

Retransmission-insensitive

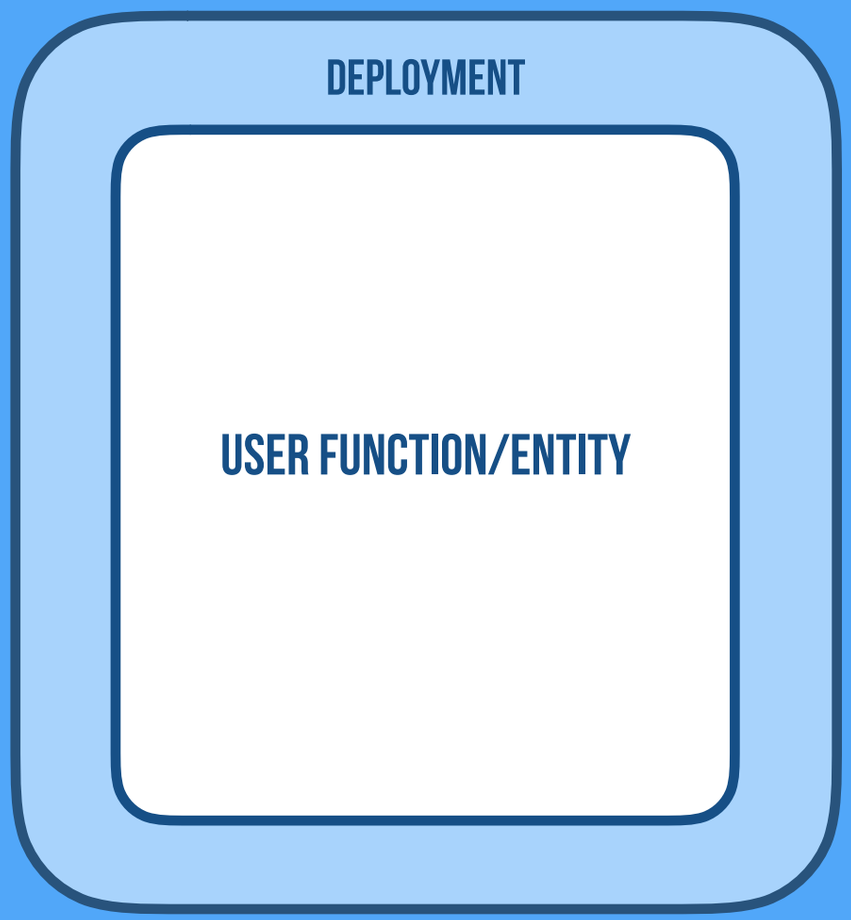
(duplication does not matter)

$a+a=a$

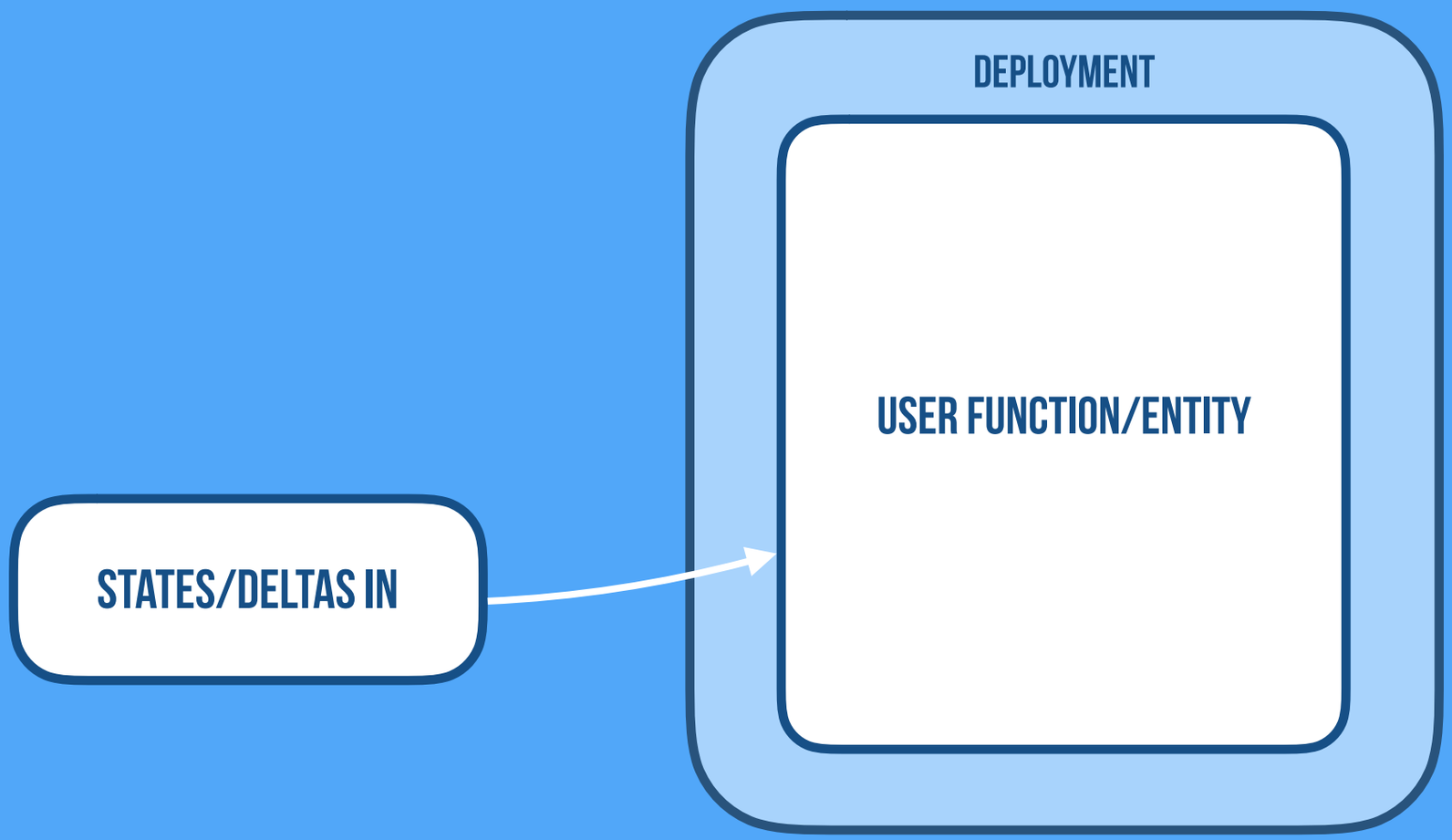
Serverless CRDTs



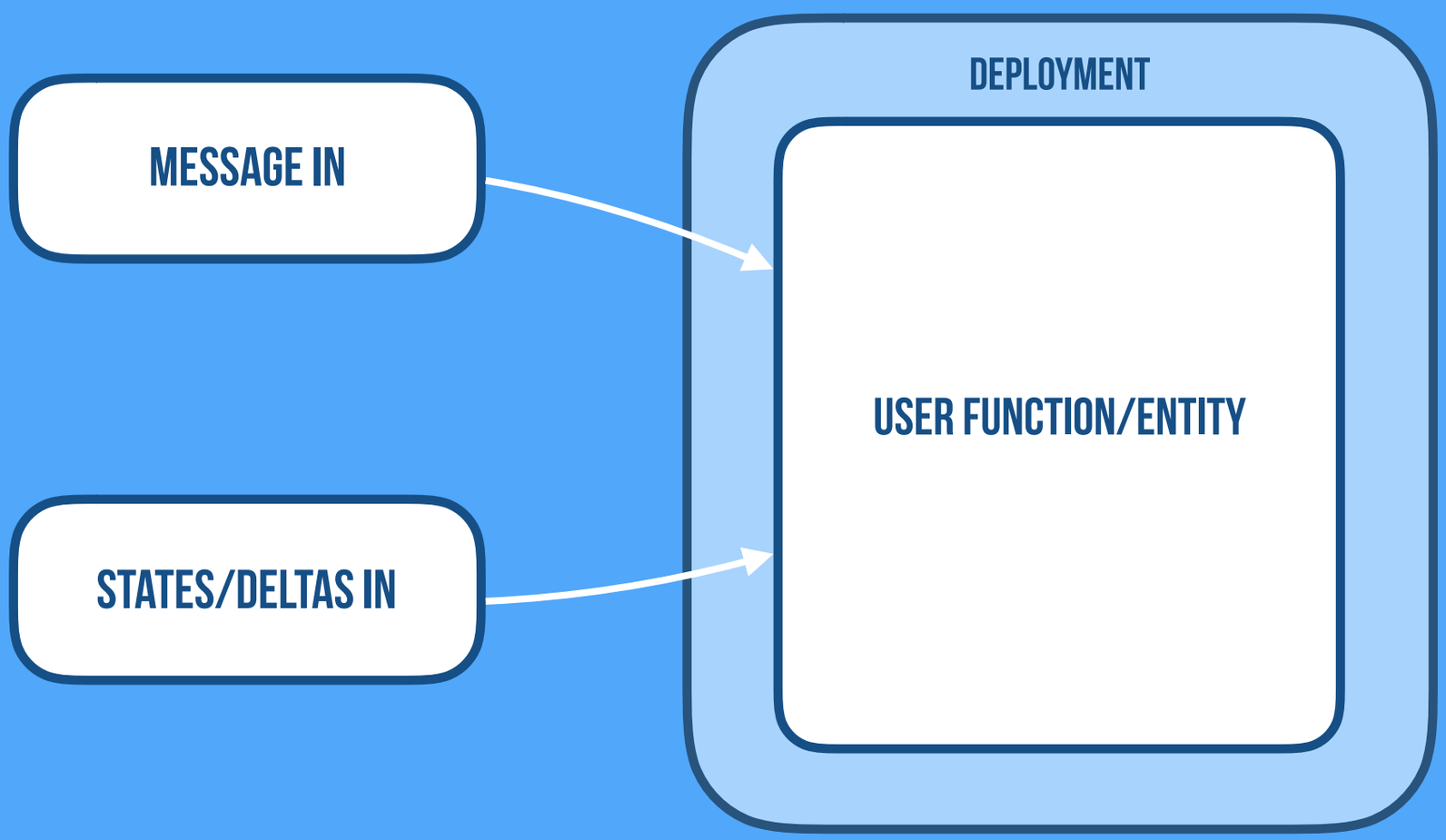
Serverless CRDTs



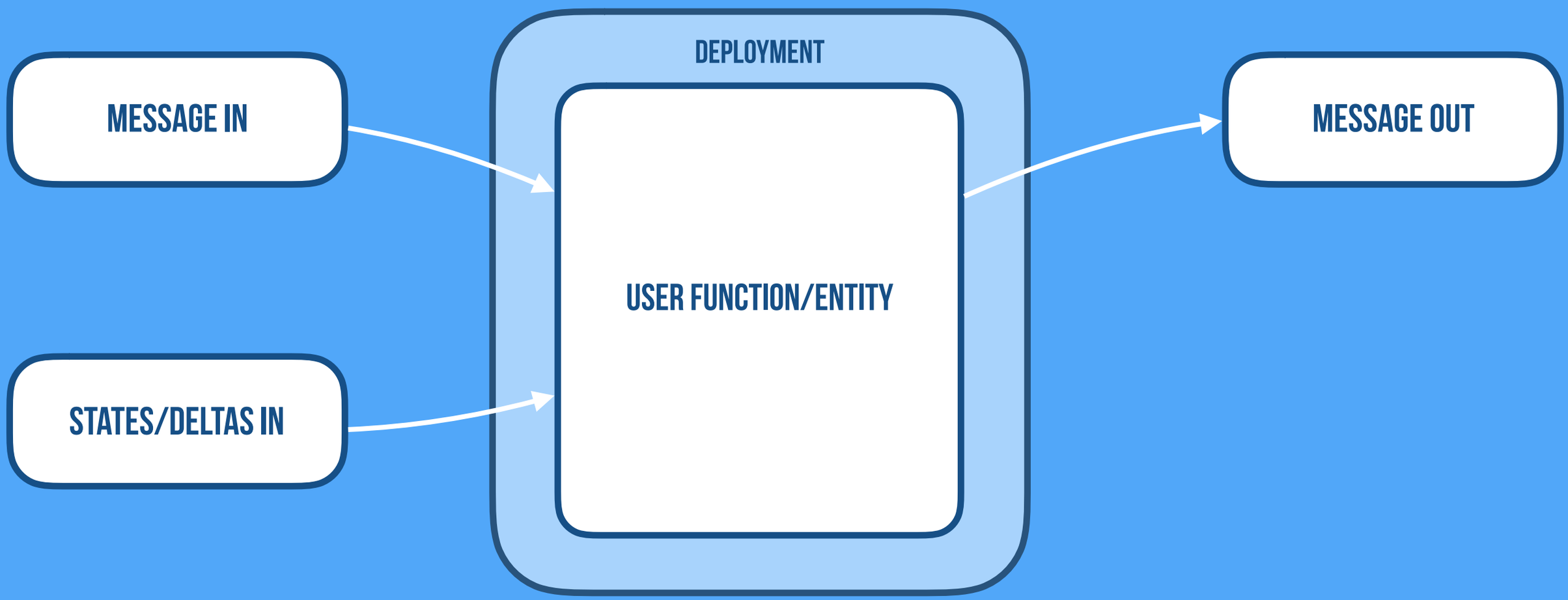
Serverless CRDTs



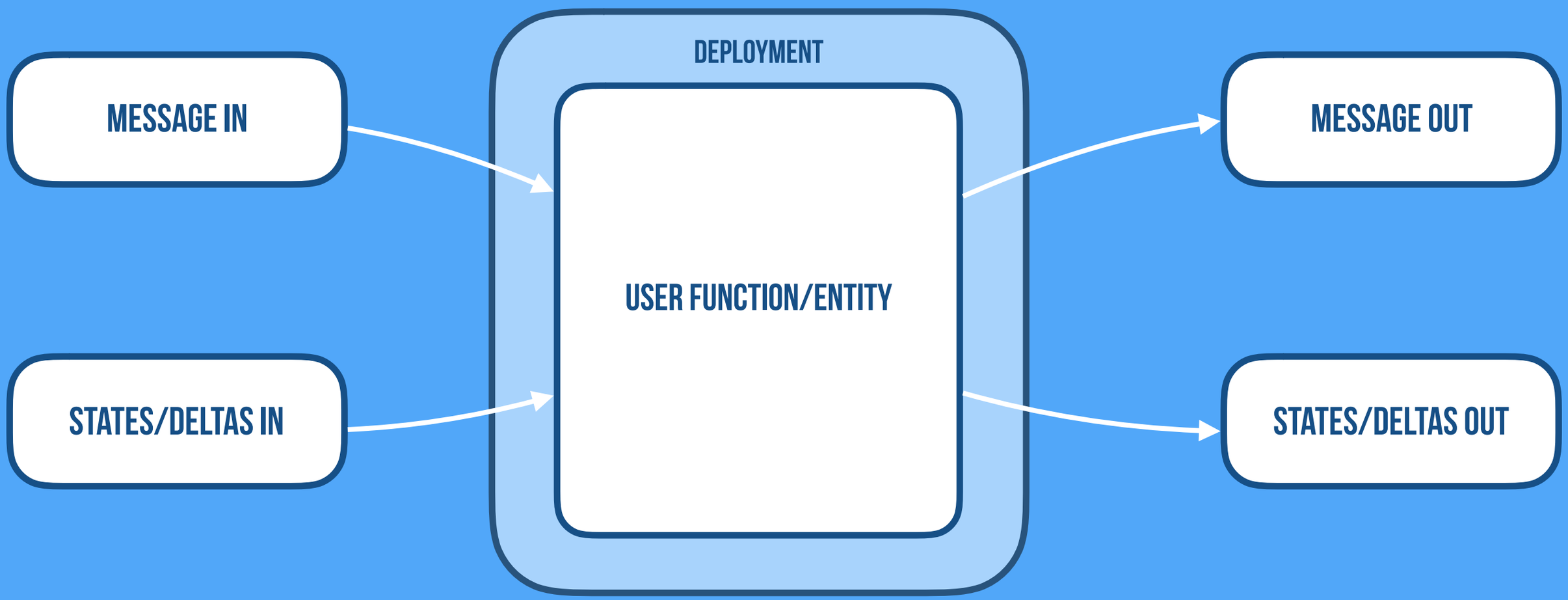
Serverless CRDTs



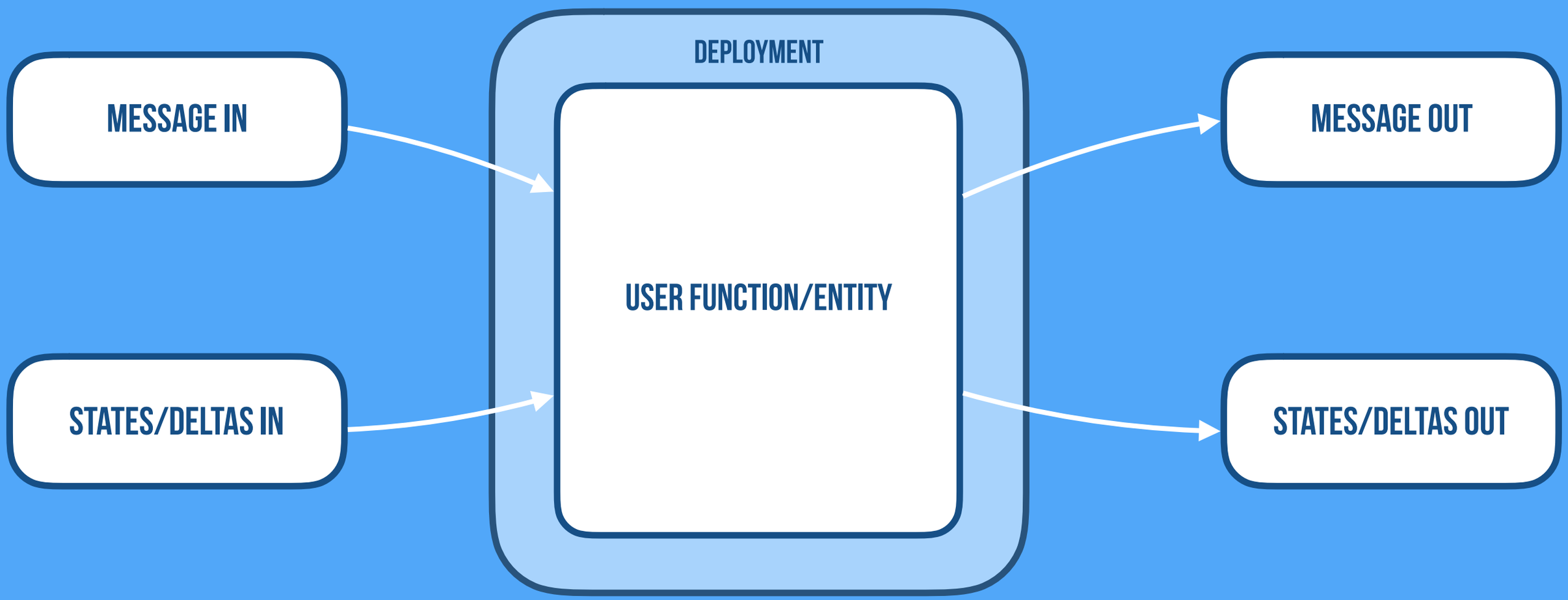
Serverless CRDTs



Serverless CRDTs

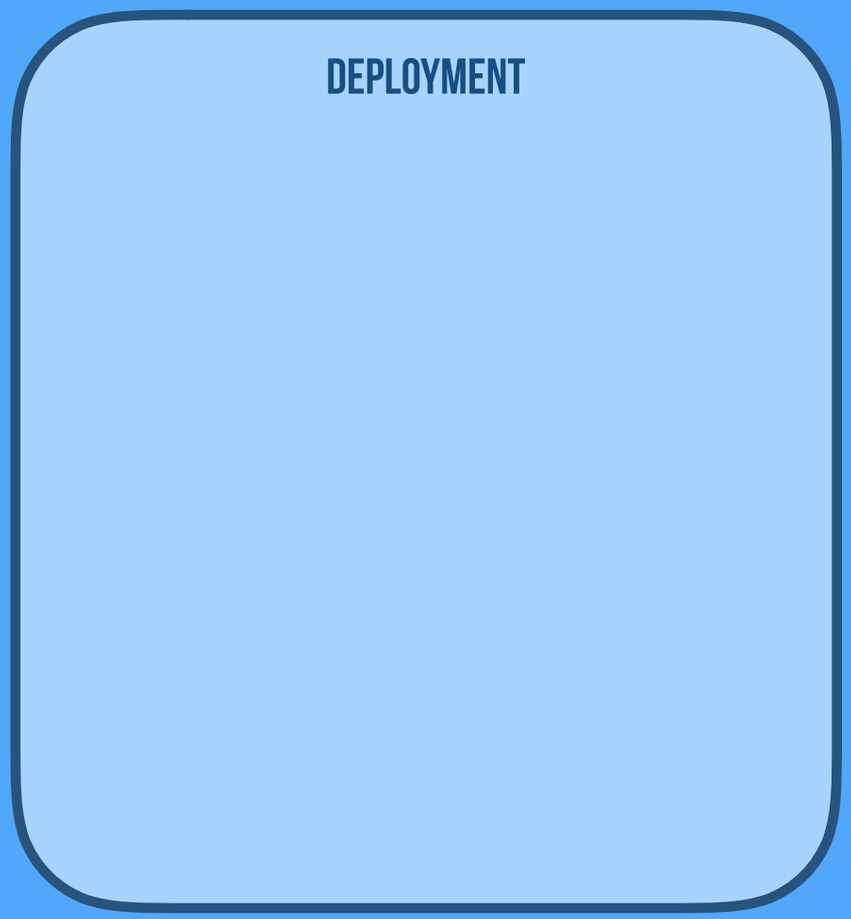


Serverless CRDTs



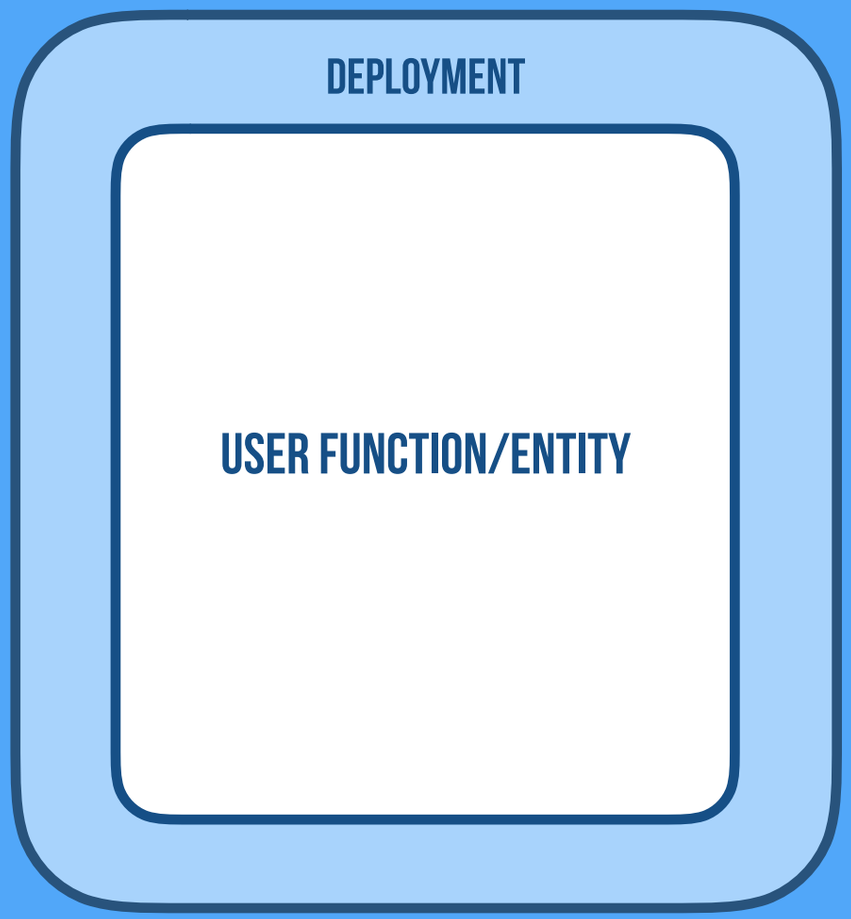
Serverless CRUD

Using KeyValue



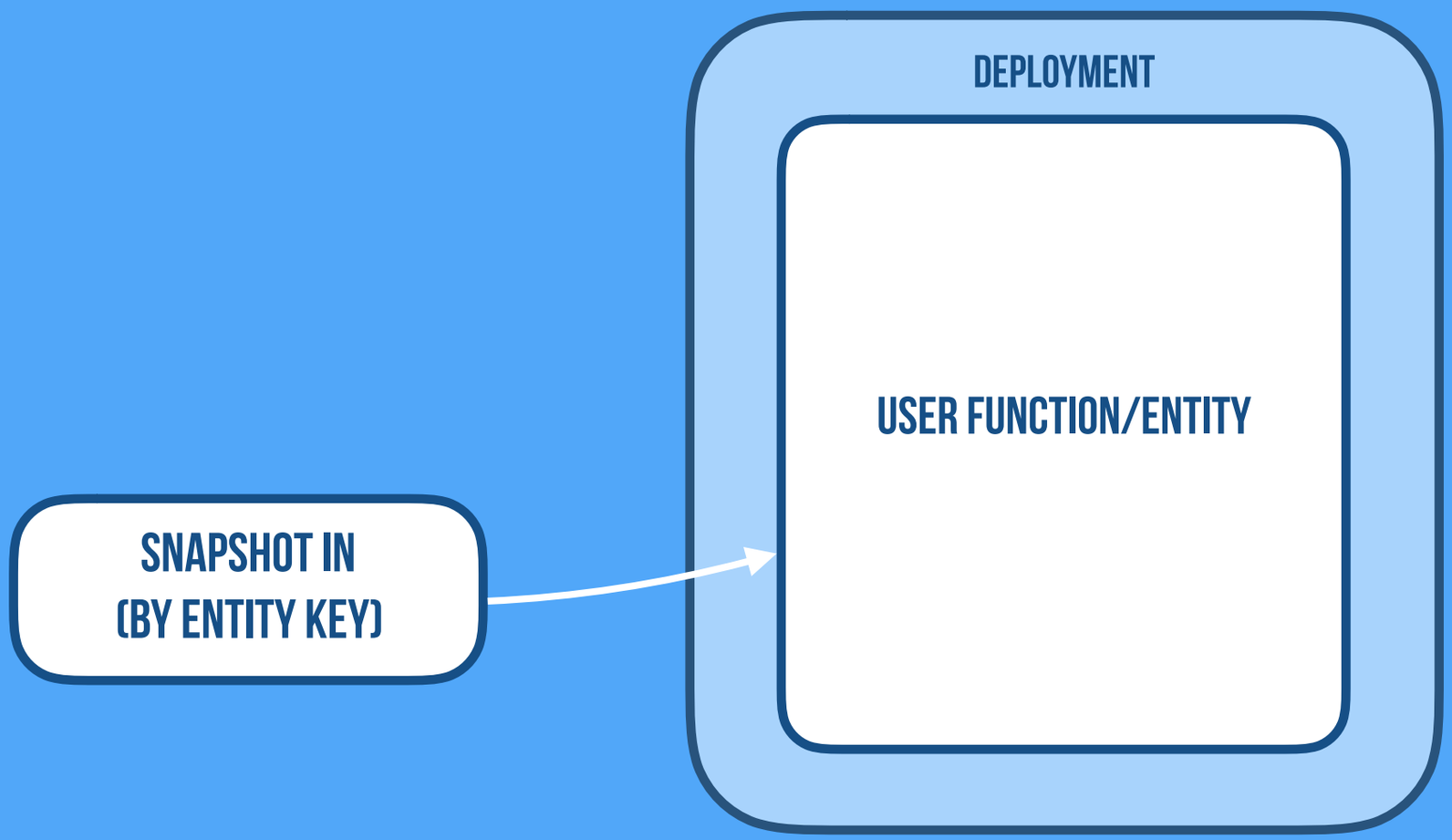
Serverless CRUD

Using KeyValue



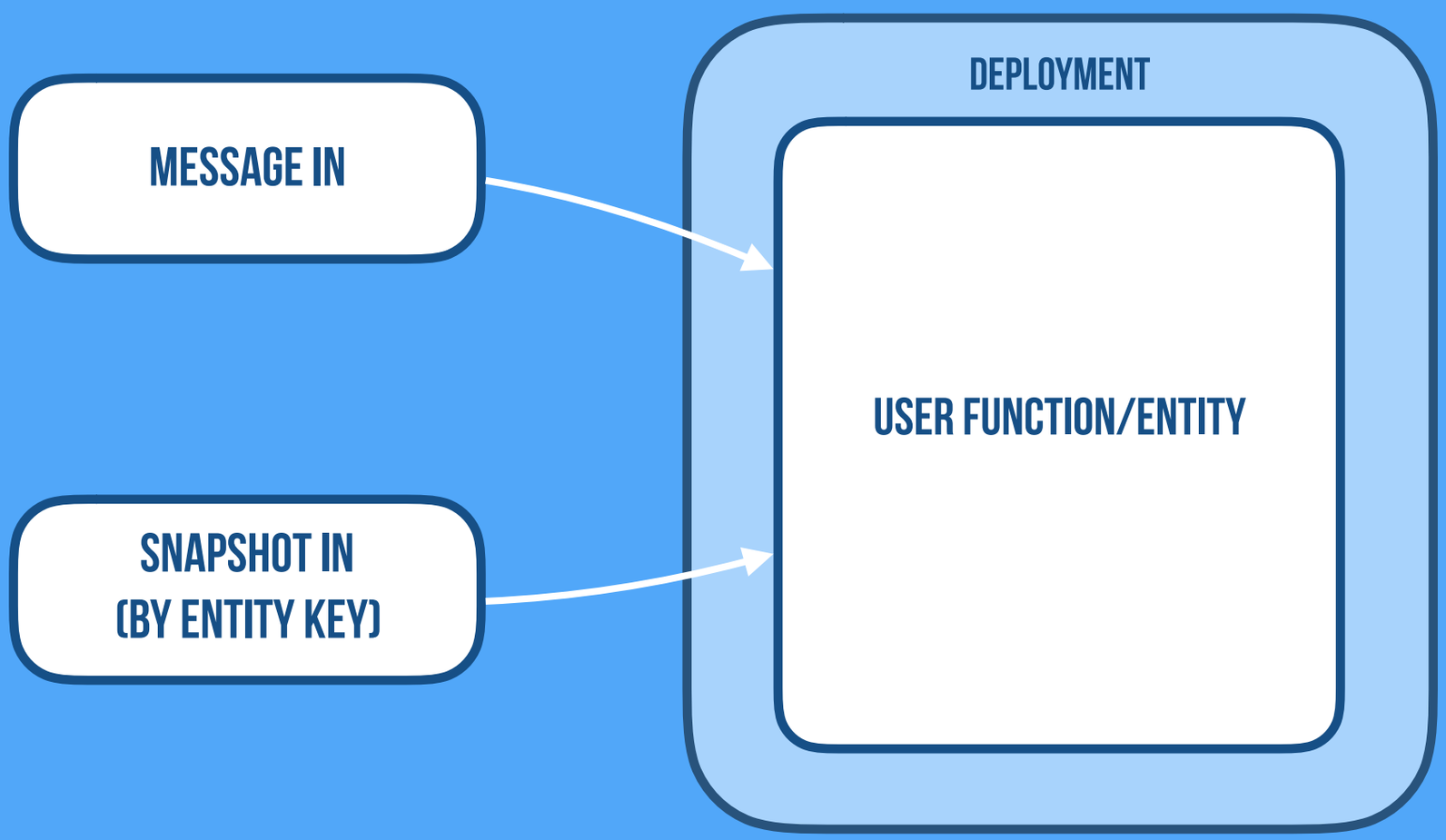
Serverless CRUD

Using KeyValue



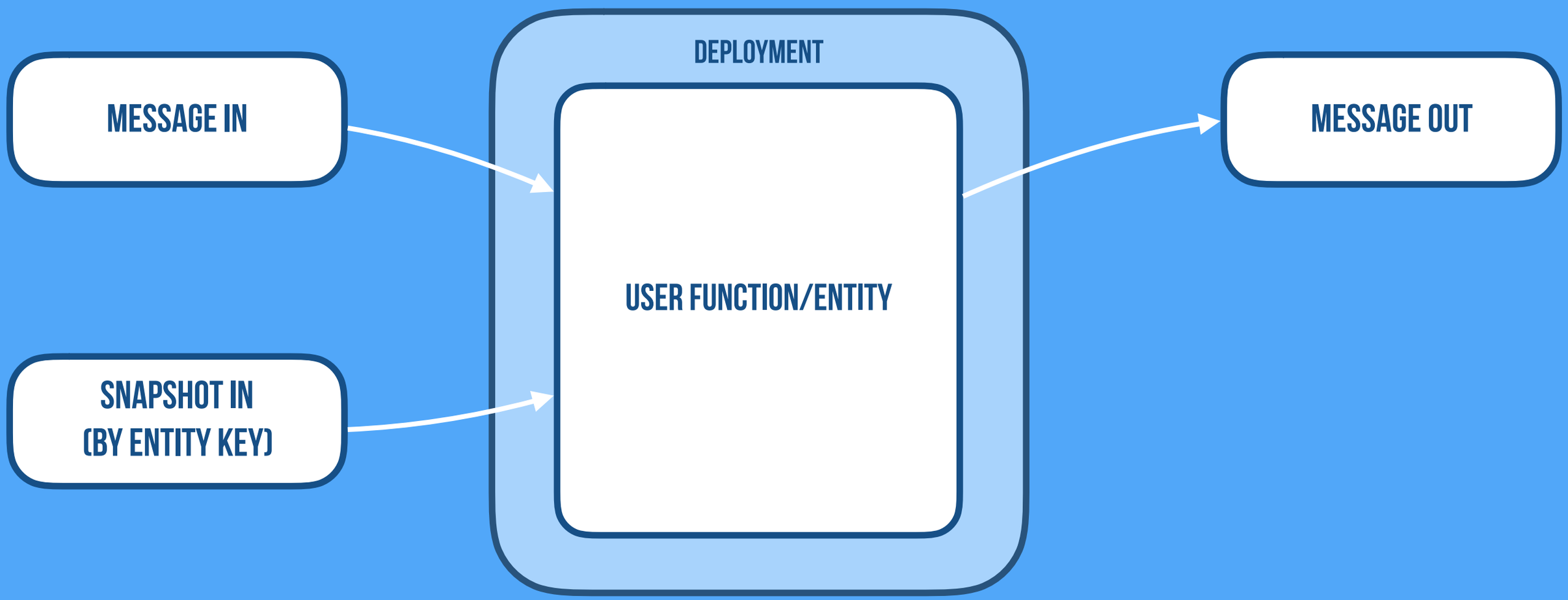
Serverless CRUD

Using KeyValue



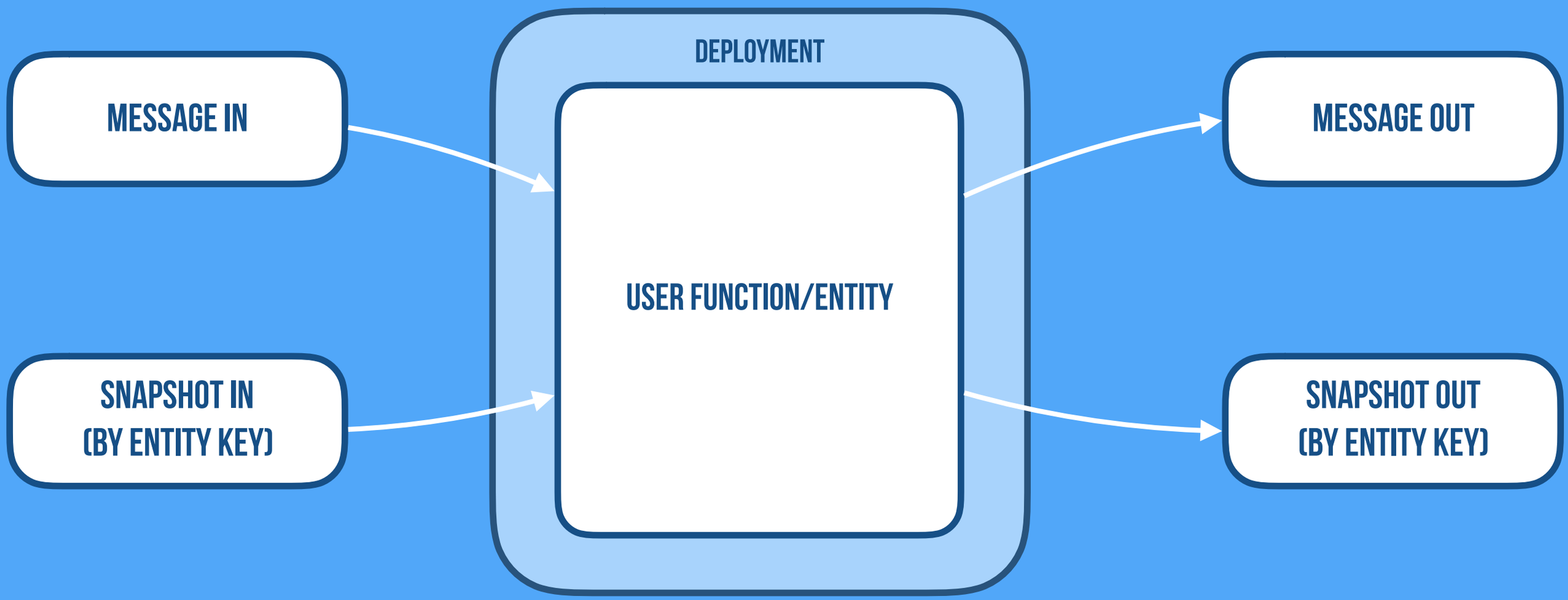
Serverless CRUD

Using KeyValue



Serverless CRUD

Using KeyValue



Example

CRDT Entity

PRESENCE FUNCTION IN A CHAT APP

github.com/cloudstateio/samples-java-chat

Protobuf Descriptor

Protobuf Descriptor

```
syntax = "proto3";
import "cloudstate/entity_key.proto";

package cloudstate.samples.chat.presence;

option java_package = "io.cloudstate.samples.chat.presence";
option java_outer_classname = "PresenceProtos";
```

Protobuf Descriptor

```
syntax = "proto3";
import "cloudstate/entity_key.proto";

package cloudstate.samples.chat.presence;

option java_package = "io.cloudstate.samples.chat.presence";
option java_outer_classname = "PresenceProtos";

message User {
  // Entity key is the unique entity/function identifier
  string name = 1 [(.cloudstate.entity_key) = true];
}

message OnlineStatus {
  bool online = 1;
}

message Empty {
}
```

Protobuf Descriptor

```
syntax = "proto3";
import "cloudstate/entity_key.proto";

package cloudstate.samples.chat.presence;

option java_package = "io.cloudstate.samples.chat.presence";
option java_outer_classname = "PresenceProtos";

message User {
  // Entity key is the unique entity/function identifier
  string name = 1 [(.cloudstate.entity_key) = true];
}

message OnlineStatus {
  bool online = 1;
}

message Empty {
}

service Presence {

  // Connect the given user
  rpc Connect(User) returns (stream Empty);

  // Monitor the online status of the given user
  rpc Monitor(User) returns (stream OnlineStatus);
}
```

CRDT Entity

CRDT Entity

```java
@CrdtEntity
public class PresenceEntity {
  private final Vote vote;         // Vote CRDT for this user. It's auto replicated
                                   // and keeps track how each node has voted
  private final String username; // Entity Key (for sharding and routing)

  public PresenceEntity(
    Optional<Vote> vote, CrdtCreationContext ctx, @EntityId String username) { … }
}
```

CRDT Entity

```
@CrdtEntity
public class PresenceEntity {
  private final Vote vote;         // Vote CRDT for this user. It's auto replicated
                                   // and keeps track how each node has voted
  private final String username; // Entity Key (for sharding and routing)

  public PresenceEntity(
    Optional<Vote> vote, CrdtCreationContext ctx, @EntityId String username) { … }


  public static void main(String... args) {
    new CloudState()
      .registerCrdtEntity(…)
      .start();
  }


}
```

CRDT Entity

```java
@CrdtEntity
public class PresenceEntity {
  private final Vote vote;         // Vote CRDT for this user. It's auto replicated
                                   // and keeps track how each node has voted
  private final String username; // Entity Key (for sharding and routing)

  public PresenceEntity(
    Optional<Vote> vote, CrdtCreationContext ctx, @EntityId String username) { … }


  public static void main(String... args) {
    new CloudState()
      .registerCrdtEntity(…)
      .start();
  }

  // Here we implement the Protobuf Service API, our business logic
  @CommandHandler
  public void connect(StreamedCommandContext<Empty> ctx) {
    vote.vote(true);
    …
  }

}
```

CRDT Entity

```java
@CrdtEntity
public class PresenceEntity {
  private final Vote vote;         // Vote CRDT for this user. It's auto replicated
                                   // and keeps track how each node has voted
  private final String username; // Entity Key (for sharding and routing)

  public PresenceEntity(
    Optional<Vote> vote, CrdtCreationContext ctx, @EntityId String username) { … }


  public static void main(String... args) {
    new CloudState()
      .registerCrdtEntity(…)
      .start();
  }

  // Here we implement the Protobuf Service API, our business logic
  @CommandHandler
  public void connect(StreamedCommandContext<Empty> ctx) {
    vote.vote(true);
    …
  }

  @CommandHandler
  public OnlineStatus monitor(StreamedCommandContext<OnlineStatus> ctx) {
    ctx.onChange(change → {
      …
    });
    …
  }
}
```

Run in Kubernetes

Run in Kubernetes

```
# Install Cloudstate
kubectl create namespace cloudstate
```

Run in Kubernetes

```
# Install Cloudstate
kubectl create namespace cloudstate

kubectl apply -n cloudstate -f https://github.com/
cloudstateio/cloudstate/releases/download/v0.4/
cloudstate-0.4.yaml
```

Run in Kubernetes

```
# Install Cloudstate
kubectl create namespace cloudstate

kubectl apply -n cloudstate -f https://github.com/
cloudstateio/cloudstate/releases/download/v0.4/
cloudstate-0.4.yaml

# Install our Presence app and Gateway
kubectl apply -f https://raw.githubusercontent.com/
cloudstateio/samples-java-chat/master/deploy/
presence.yaml
```

Run in Kubernetes

```
# Install Cloudstate
kubectl create namespace cloudstate

kubectl apply -n cloudstate -f https://github.com/
cloudstateio/cloudstate/releases/download/v0.4/
cloudstate-0.4.yaml

# Install our Presence app and Gateway
kubectl apply -f https://raw.githubusercontent.com/
cloudstateio/samples-java-chat/master/deploy/
presence.yaml

kubectl apply -f https://raw.githubusercontent.com/
cloudstateio/samples-java-chat/master/deploy/
gateway.yaml
```

Run in Kubernetes

```
# Install Cloudstate
kubectl create namespace cloudstate

kubectl apply -n cloudstate -f https://github.com/
cloudstateio/cloudstate/releases/download/v0.4/
cloudstate-0.4.yaml

# Install our Presence app and Gateway
kubectl apply -f https://raw.githubusercontent.com/
cloudstateio/samples-java-chat/master/deploy/
presence.yaml

kubectl apply -f https://raw.githubusercontent.com/
cloudstateio/samples-java-chat/master/deploy/
gateway.yaml

# Scale up the app to 3 nodes
kubectl scale deploy/presence-deployment --replicas 3
```

Join Us

Try Out
The Next Generation
Stateful Serverless

cloudstate.io