DATA+AI SUMMIT 2022

How Socat and UNIX Pipes Help Data Integration

Orchestrating CLIs on K8s using native Linux tools

Davin Chia

Tech Lead, Cloud, Infrastructure and Tooling, Airbyte

ORGANIZED BY databricks

Overview

- What is Airbyte?
  - Problem Statement
- Various naive approaches
  - Fixed job pools
  - Running Docker in Docker
- Final solution
  - Socat with Sidecar container
  - Named pipes with K8s init container
  - Dynamically create Airbyte sync jobs!
- Q & A

What is Airbyte?

What are we trying to solve?

OSS framework for syncing data - EL of ELT

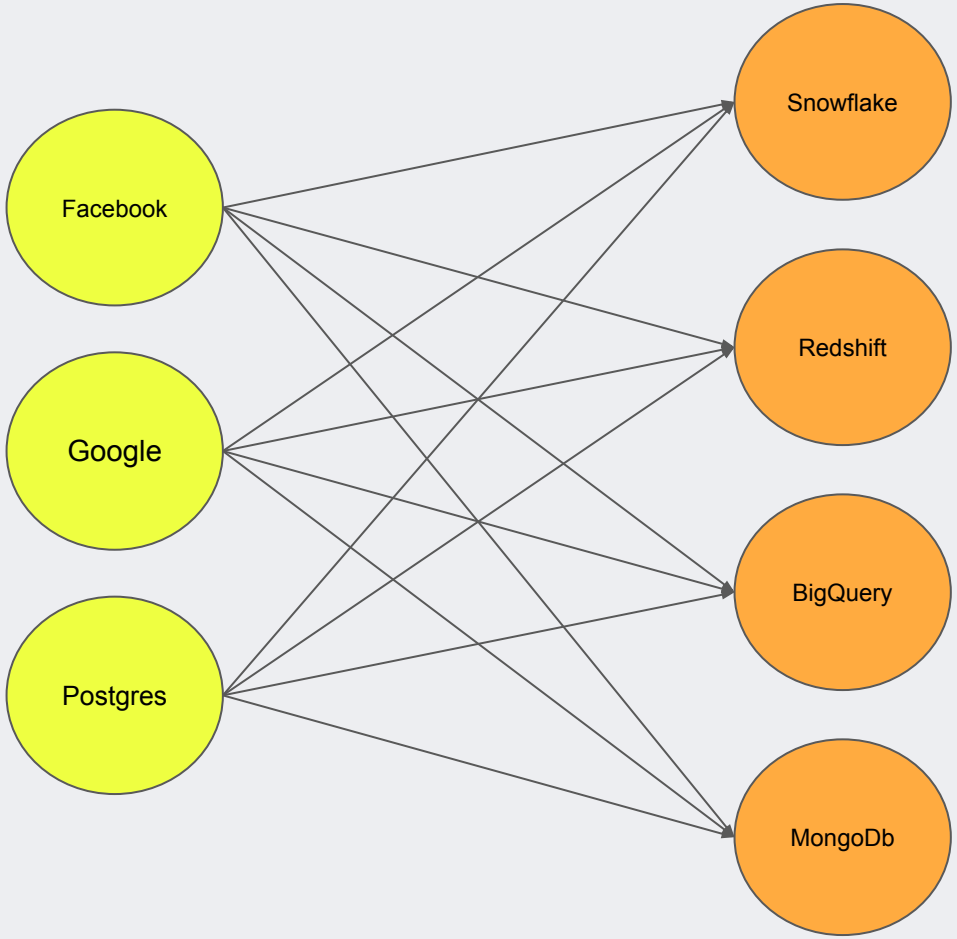
What is Airbyte?

What are we trying to solve?

OSS framework for syncing data - EL of ELT

Number of Integrations

What is Airbyte?



M Source x N Destination = M x N Connections!

What is Airbyte?

What are we trying to solve?

Crushing Tech Debt

What is Airbyte?

What are we trying to solve?

OSS framework for syncing data

Number of Integrations

Reliability of scheduling system

What is Airbyte?

What are we trying to solve?

OSS framework for syncing data

Cron Strings, Orchestrators, DAGs Frameworks, Workflow Orchestration Engines..

Large tech space with complex operational learnings

What is Airbyte?

What are we trying to solve?

OSS framework for syncing data

Number of Integrations

Reliability of scheduling system

What is Airbyte?

What are we trying to solve?

OSS framework for syncing data

Eng Team Crushed

What is Airbyte?

OSS framework for syncing data

Airbyte Protocol & Connector Development Kit - OSS Contributions

Airbyte Platform with Scheduler & UI

What is Airbyte?

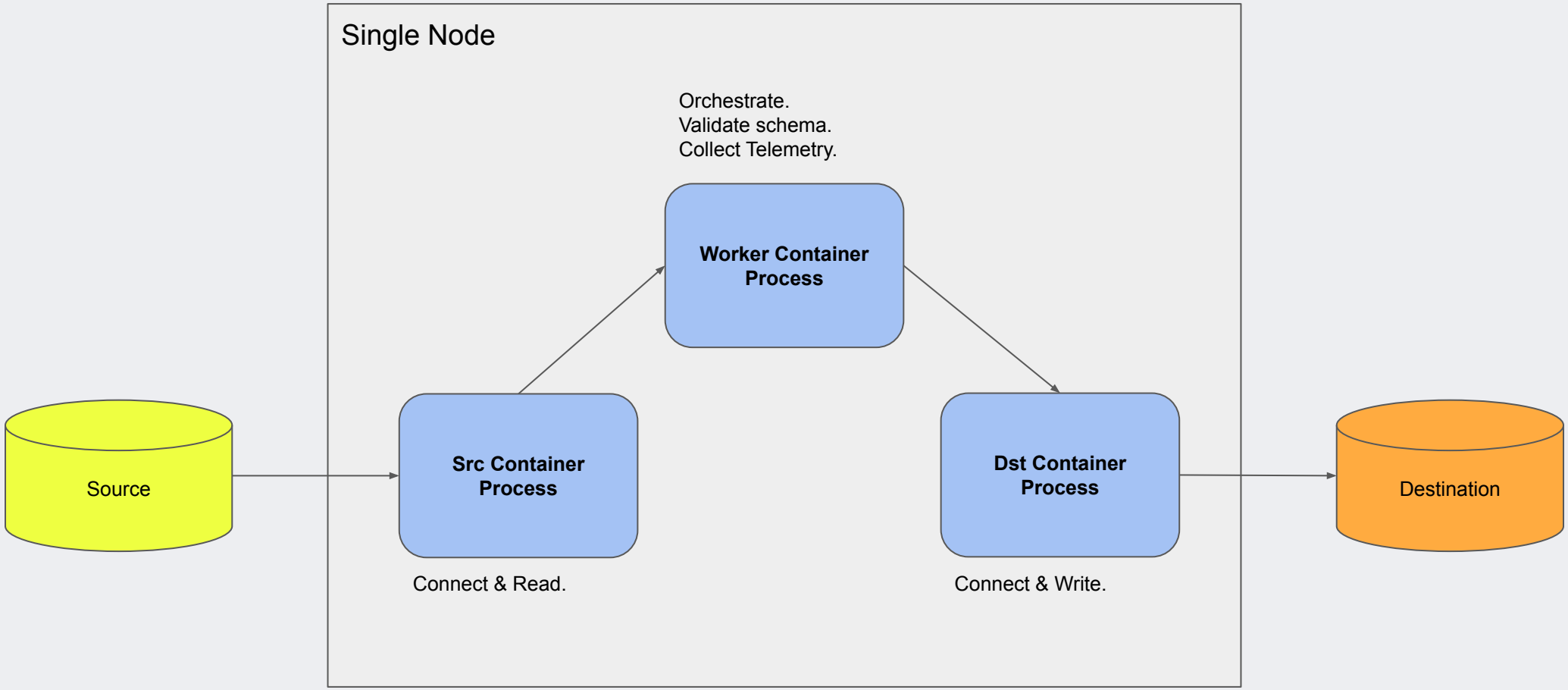
The Airbyte Protocol

Airbyte Protocol = JSON-typed container interface for integrations

Started with a Docker Deployment

What is Airbyte?

Dynamic Jobs!



What is Airbyte?

Dynamic Jobs!

```java
// note: resources are closed in the opposite order in which they are declared. thus source will be
// closed first (which is what we want).
try (destination; source) {
  destination.start(destinationConfig, jobRoot);
  source.start(sourceConfig, jobRoot);

  // note: `whenComplete` is used instead of `exceptionally` so that the original exception is still
  // thrown
  final CompletableFuture<?> destinationOutputThreadFuture = CompletableFuture.runAsync(
      getDestinationOutputRunnable(destination, cancelled, messageTracker, mdc),
      executors).whenComplete((msg, ex) -> {
        if (ex != null) {
          if (ex.getCause() instanceof DestinationException) {
            destinationRunnableFailureRef.set(FailureHelper.destinationFailure(ex, Long.valueOf(jobId), attempt));
          } else {
            destinationRunnableFailureRef.set(FailureHelper.replicationFailure(ex, Long.valueOf(jobId), attempt));
          }
        }
      });
```

Java Process Interface

What is Airbyte?

The General Data Integration Case

Date Integrations

Read CLI

Write CLI

What is Airbyte?

The General Data Integration Case

Read CLI

Write CLI

```
psql -c "Copy (Select * From geoip_v4) To STDOUT With CSV HEADER DELIMITER ',';" | gzip |
aws s3 cp - s3://my-bucket/geoip_v4_data.csv.gz
```

Scaling This

Simple and works great until it doesn’t

Users running >1k jobs a day started hitting vertical scaling limits

Kubernetes!

Problem Statement

How do we dynamically orchestrate CLIs on Kubernetes while making no assumptions on container runtimes?

Constraints

No container assumptions

Kube scaling mechanism

Docker deployment compatible

Naive Approaches

Kube Jobs

Lambda Functions

Docker-in-Docker

Pools of Job Pods

Docker-in-Docker

How

K8s Node Docker agent Access

Dynamically create container processes on the node.

Identical to Airbyte Docker

Docker-in-Docker

Analysis

Security Risk - Custom Connectors

Not supported by public Clouds

Per-node scale limit

Docker-in-Docker

When to use?

👌 = operating own Kube cluster with very stable workloads and vetted containers

Pools of Job Pods

How

Per-connector job pod pools

Pods wait to perform certain tasks

A worker reaches out to required job pods when running a sync

Multiple implementations and scaling approaches

Pools of Job Pods

Analysis

💸💸💸 = underutilised pods

1 pool for each connector = ❌ scalable

Operationally challenging

Tactical instead of a sustainable strategy

Pools of Job Pods

When to use?

👌 = very stable workloads and limited job types

The Ideal Solution

Dynamically create Kubernetes job pods and hide this behind the existing Process interface.

The Ideal Solution

Pods created only when needed

Fits well with Docker approach (existing code structure)

The Ideal Solution

Problem 0

How to dynamically create pods?

Kube API! And the many SDKs available.

The Ideal Solution

Problem 1

How do we handle typical process interaction (STDIO) between two Kube read/write job pods?

The Ideal Solution

Problem 1

The Ideal Solution

Problem 1

Create a connection to a remote ip/port

Wrap the pod's main container's STDIO in a network protocol

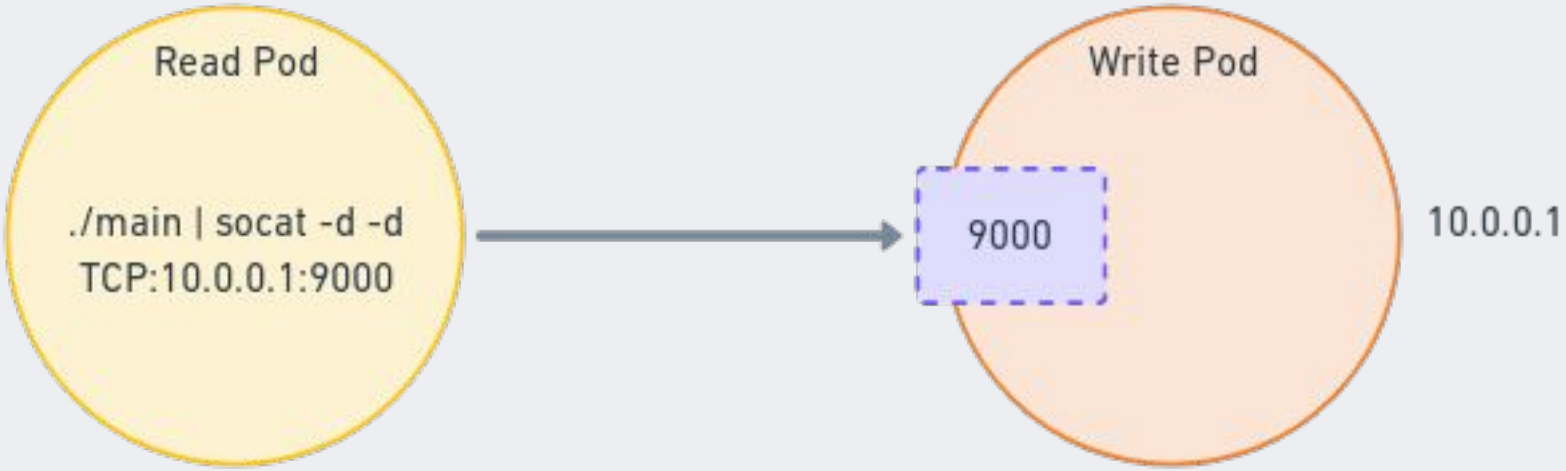
Problem 1

Socat

```
./read | socat -d -d - TCP:ip:port
```

Problem 1

Socat



The Ideal Solution

Problem 2

How do we include socat without adding runtime dependencies?

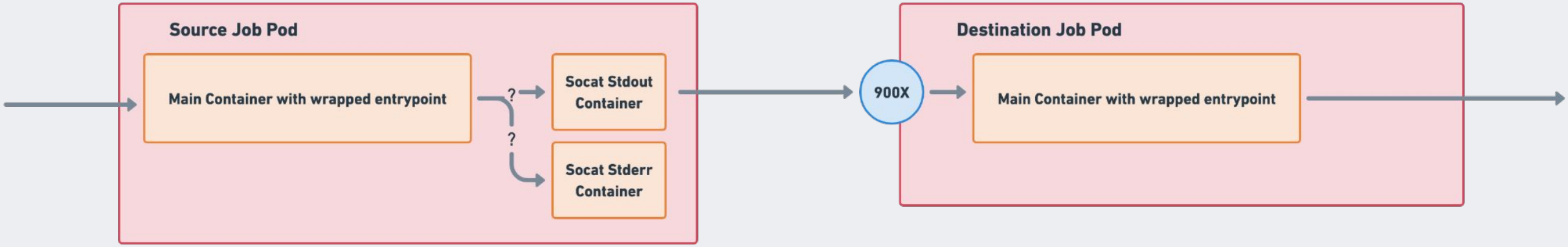
Problem 2

Socat without runtime dependencies

Sidecars!

Problem 2

Socat without runtime dependencies



The Ideal Solution

Problem 3

How do we pipe data between sidecar containers and the main container?

The Ideal Solution

Problem 3

❌ `socat_container > main_container`

Problem 3

Named Pipes

Named pipes are pipes that appear within the file system.

Name = handle for programmatic use.

Local file system access.

Problem 3

Named Pipes

```
stderr-pipe

stdout-pipe
```

Problem 3

Named Pipes

```
stderr-pipe

stdout-pipe
```

```
./read 2> stderr-pipe > stdout-pipe
```

Problem 3

Named Pipes

```
stderr-pipe

stdout-pipe
```

```
./read 2> stderr-pipe > stdout-pipe

cat stdout-pipe | socat -d - TCP:[remote-ip]:[remote-port]
```

Problem 3

Init Containers

How do we created the named pipes?

Problem 3

Init Containers

Init containers runs before all containers.

Guarantee named pipes are created.

The Ideal Solution

Coming Together

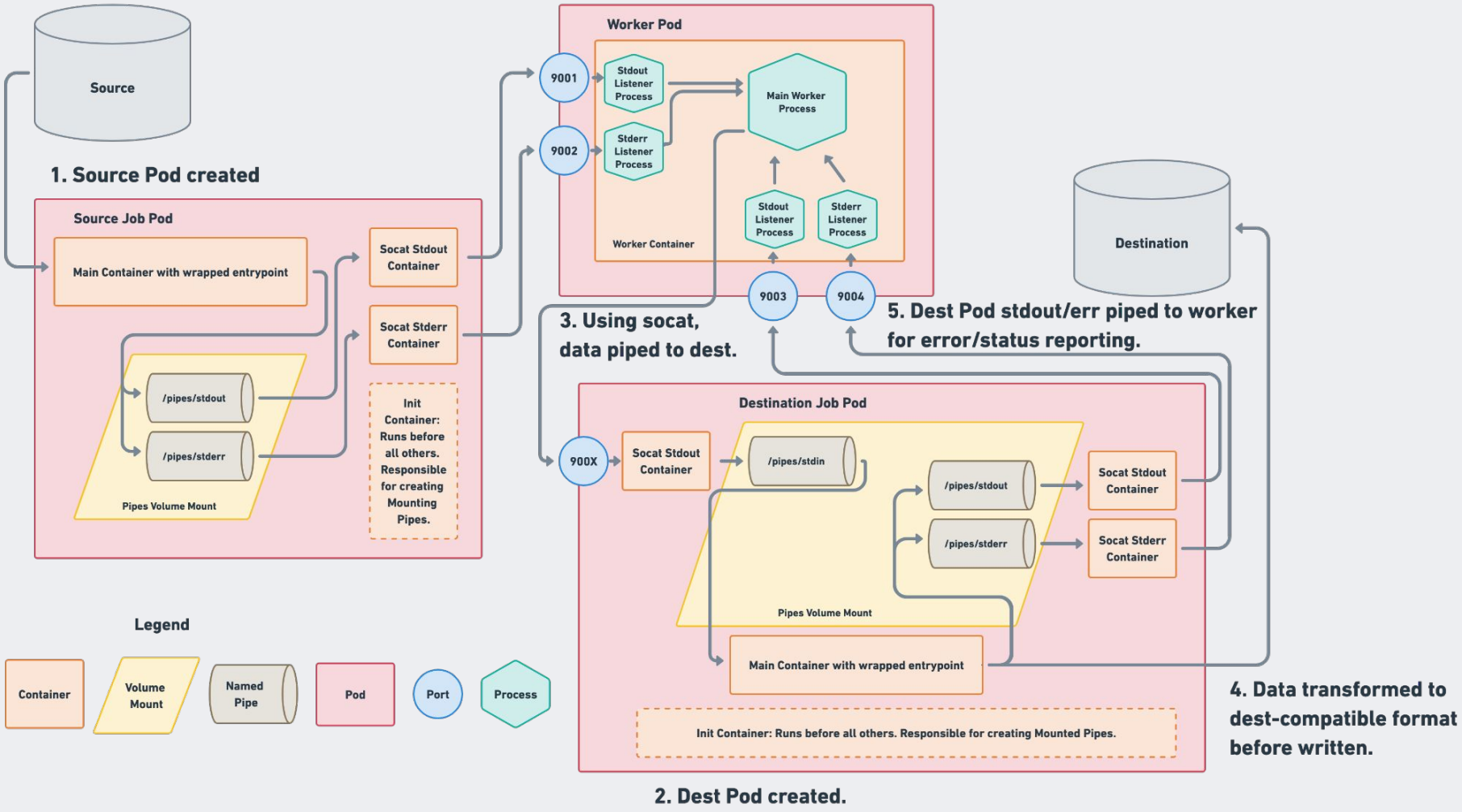
How to pipe data between Kube pods: use socat.

How to bundle socat with containers: use the sidecar pattern.

How to pipe data between our two containers: use named pipes.

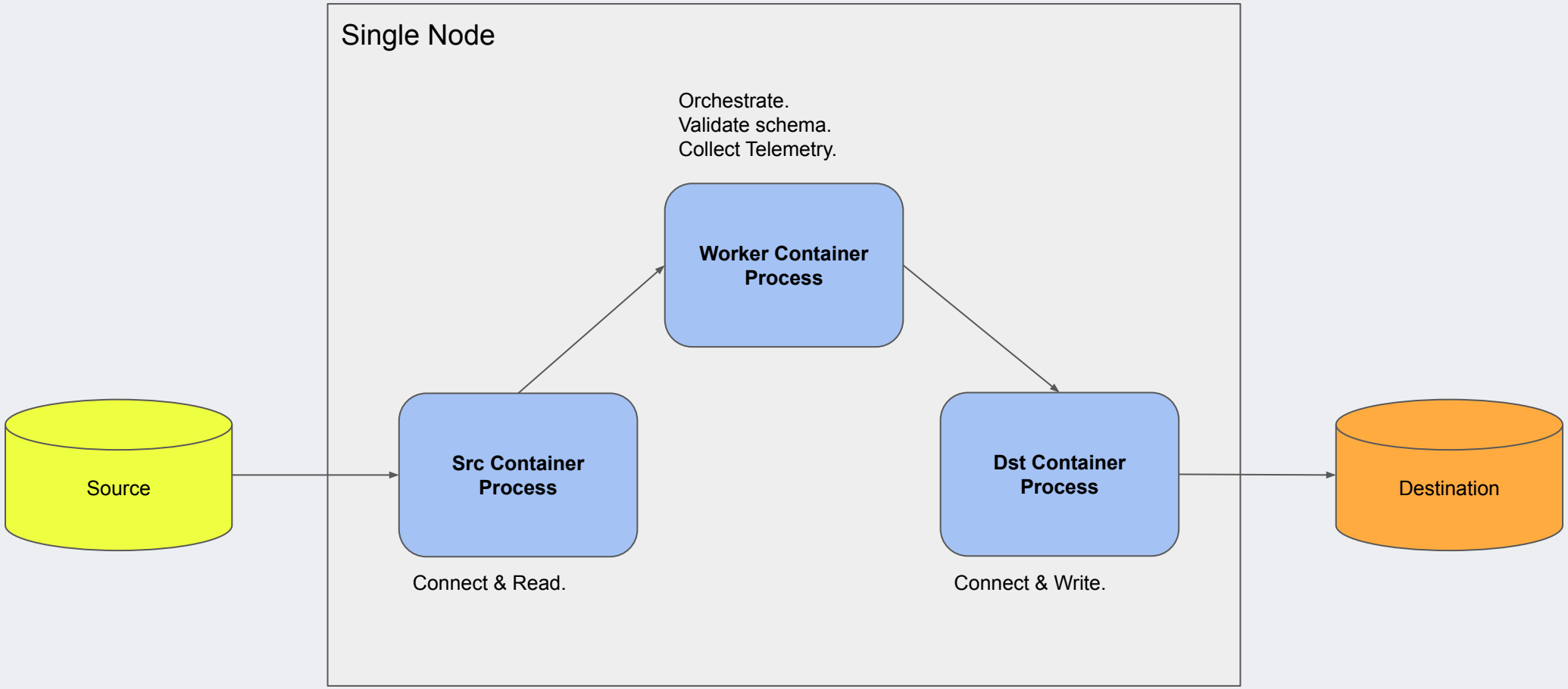
The Ideal Solution

Coming Together



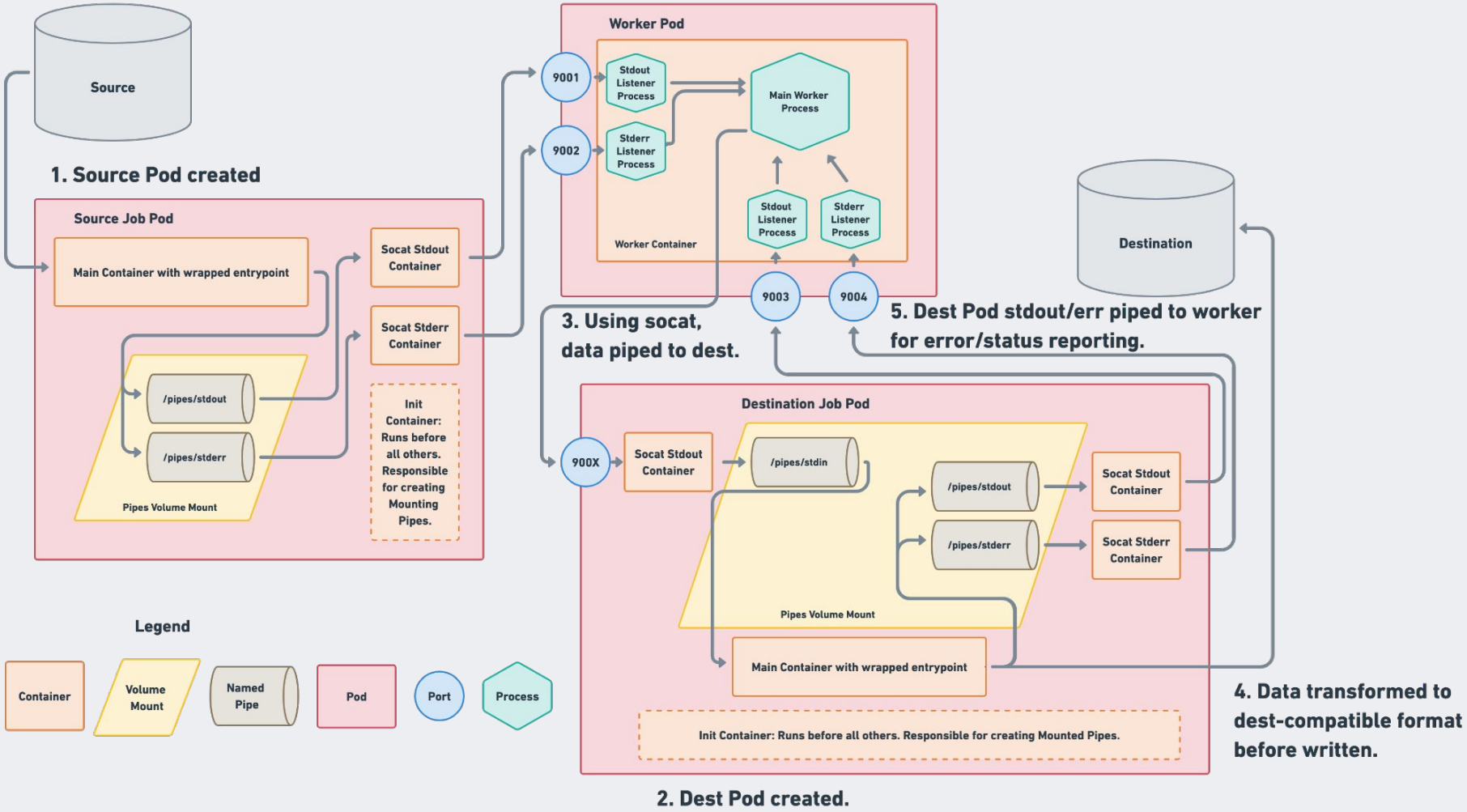
What is Airbyte?

Dynamic Jobs!



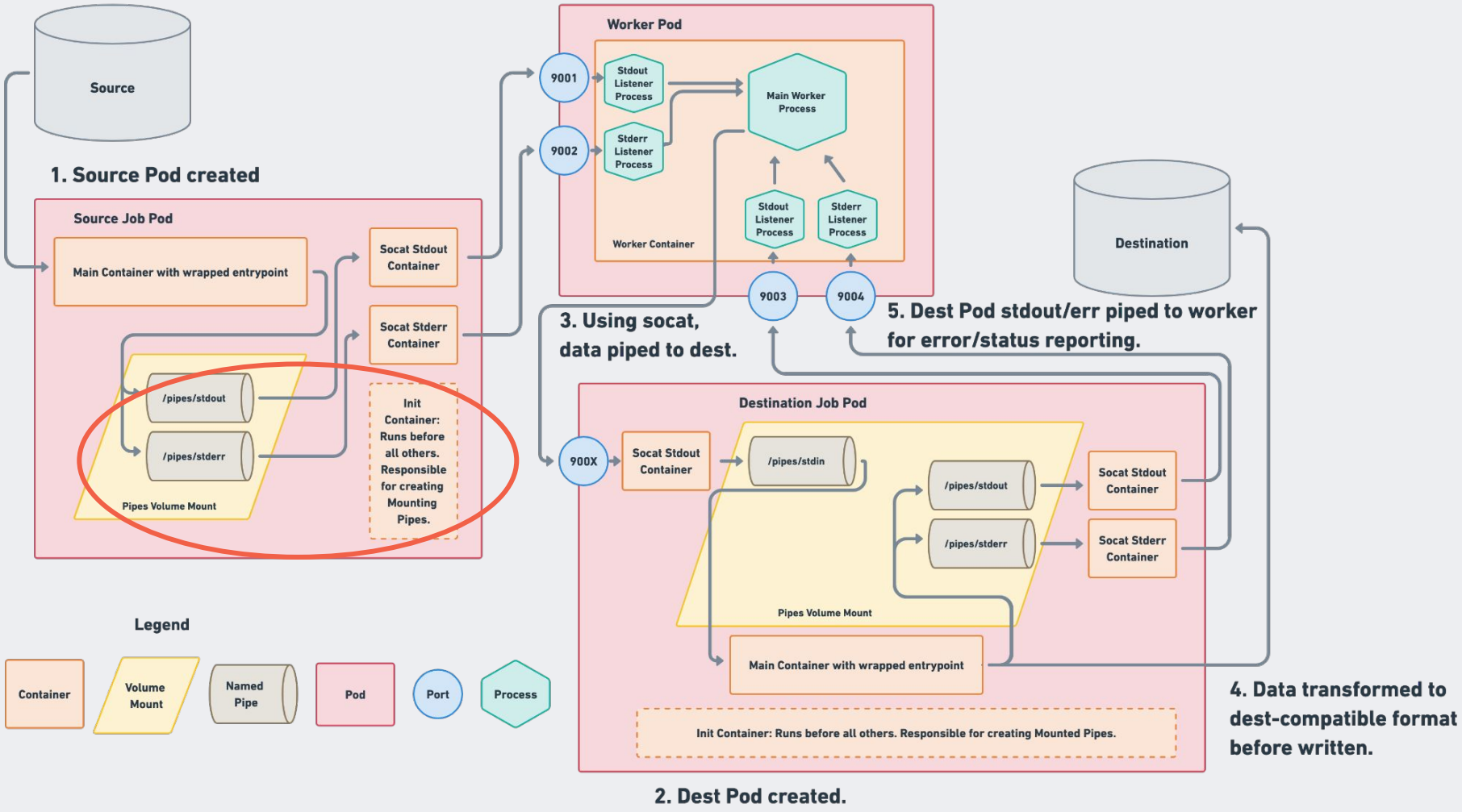
The Ideal Solution

Coming Together



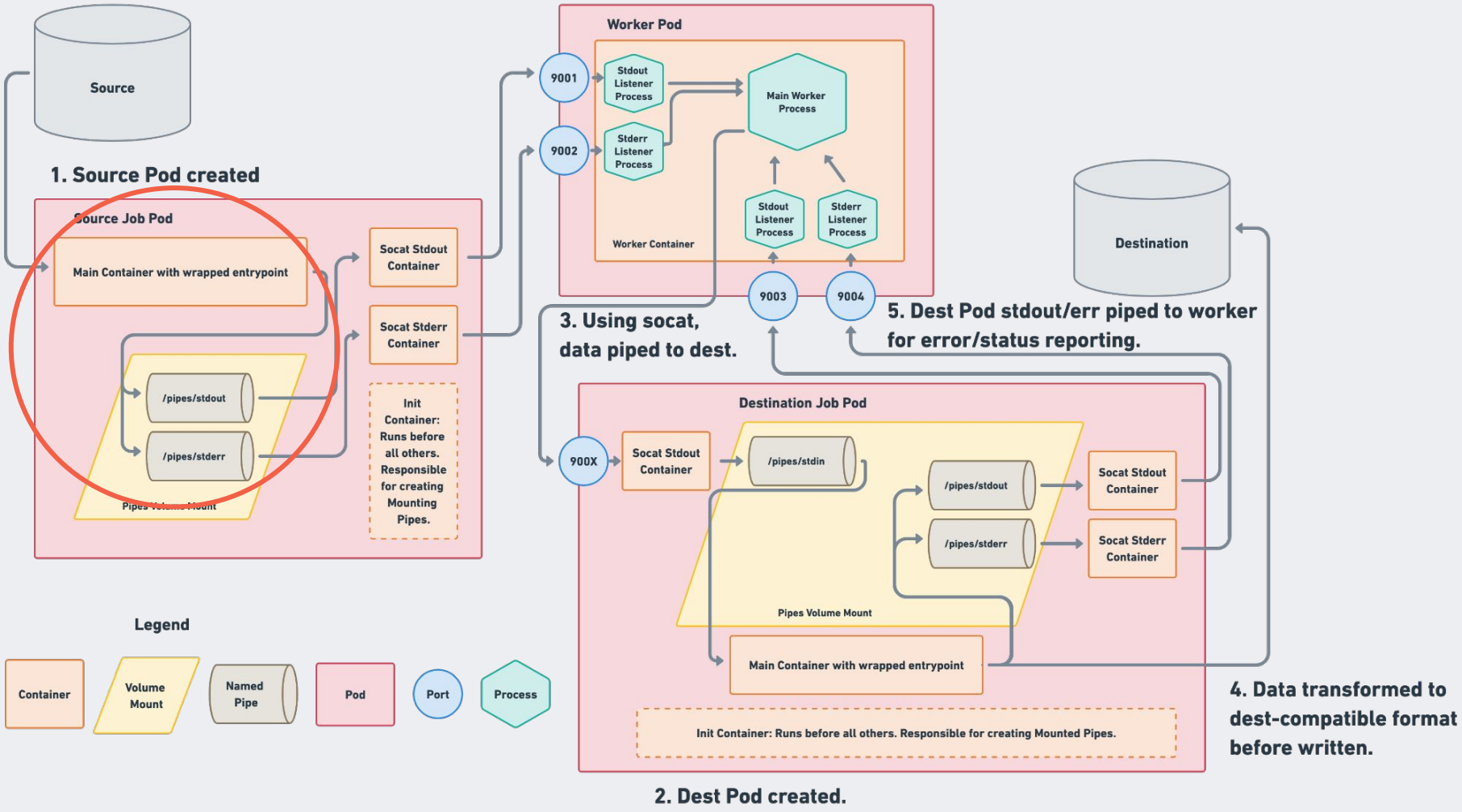
The Ideal Solution

Coming Together



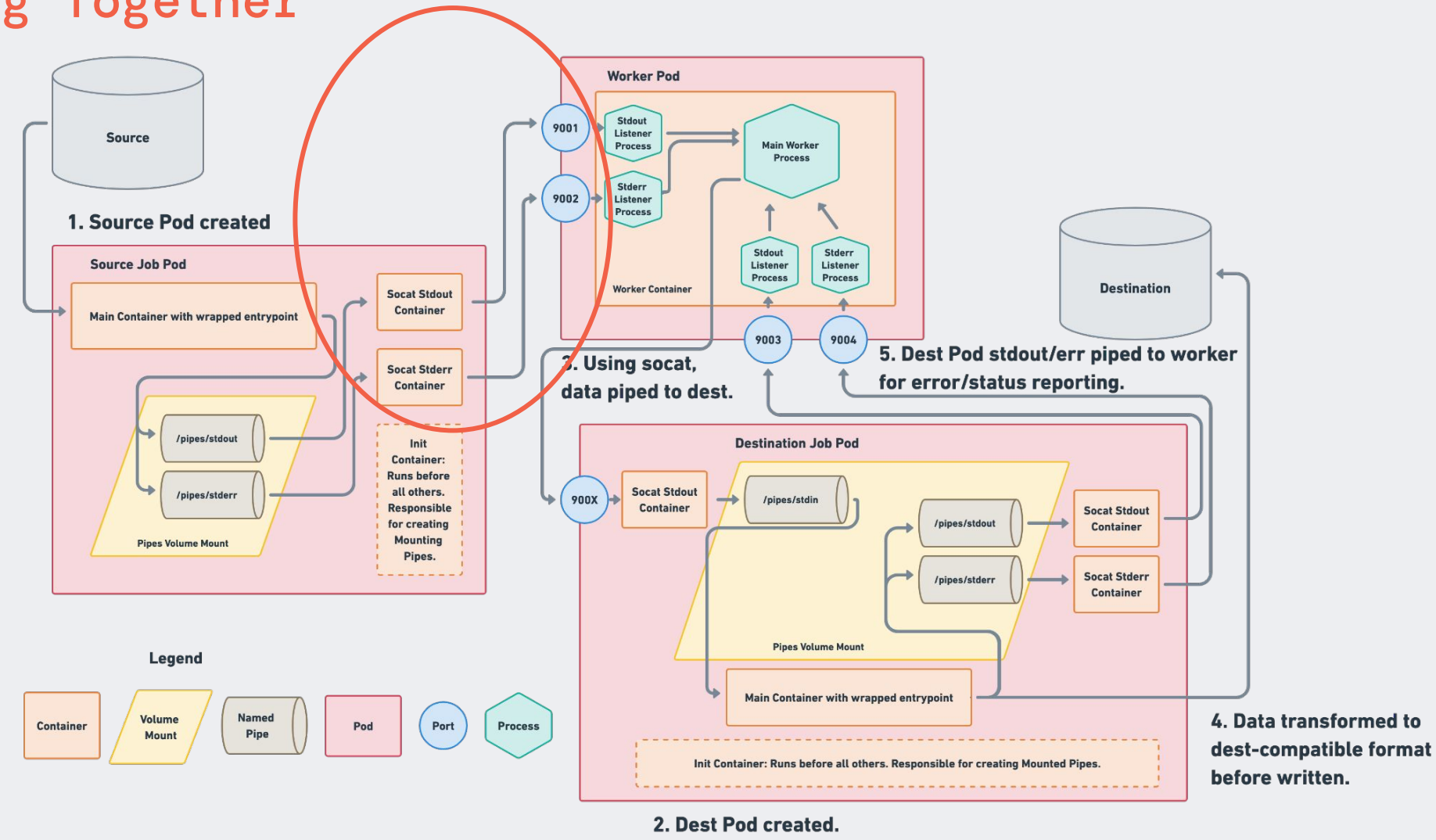
The Ideal Solution

Coming Together



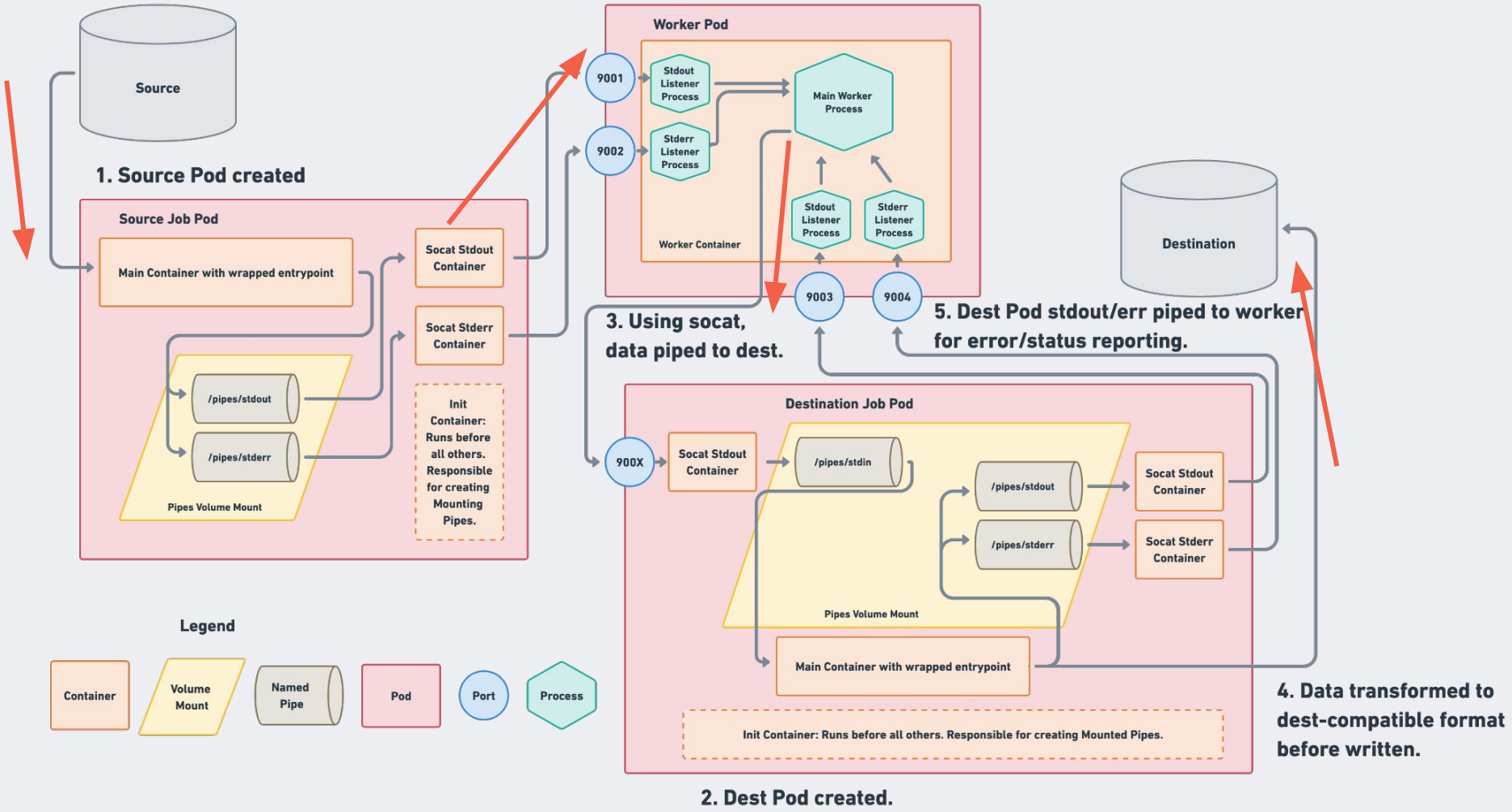
The Ideal Solution

Coming Together



The Ideal Solution

Coming Together



Demo

Gotchas

K8s pod startup tuning

Correct EOF signal propagation

Watching Kube events for Pod status

Conclusion

The world of CLIs

Socat, Linux Operators, Named Pipes and Sidecar Containers

Powers hundreds of Airbyte Kube Deployments everyday!

Questions?

DATA+AI
SUMMIT 2022

Thank you

Davin Chia
Tech Lead, Cloud, Infrastructure and Tooling, Airbyte

DATA+AI
SUMMIT 2022