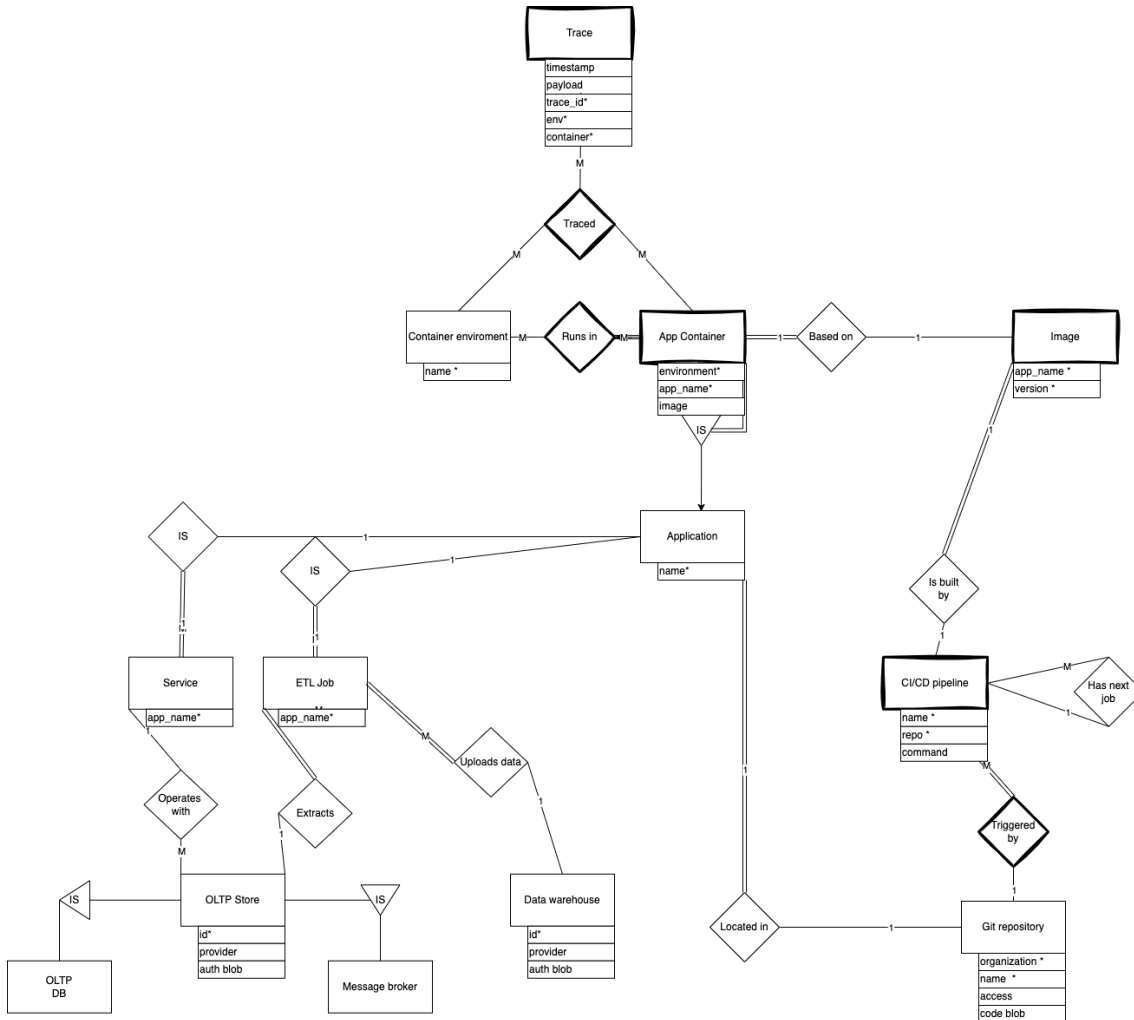


# Entity relationship diagram

Simple cloud native application ontology.



## Description

### Application

- Application - generic type of program.
- Service - is an Application, designed to run as a server. **Can** use oltp store.
- ETL job - is an Application, designed to run as a daemon to process data(streaming or batching). **Must** extract data from oltp storage, process it and upload to data warehouse.
- OLTP storage - generic type of OLTP storage.
- OLTP db - represents any oltp database(cockroachDB, postgresSQL, etc.) which should be used for instant domain operations.
- Message broker - represents any message broker(kafka, nats, rabbitMQ, etc.).

## Code management

- Git repository - entity where application code is located.
- CI/CD pipeline - job, triggered by interval or changes in Git repository. Pipeline can trigger other pipelines, even in different repo's (so has relationship with itself).

*CI/CD pipeline is weak entity, its primary key is composite and contains repo\_id.*

## Delivery

- Container environment - system for automating software deployment, scaling, and management.
- App container - subtype of application (is a). Represents running container from [OCI](#), runs on **Container environment**. Sometimes it's necessary to run application replica on different environments. Uses image artifact to bootstrap.

*App container is a weak entity, as its primary key depends on app\_name and environment\_name.*

- Image - represents image from [OCI](#). Just an application gathered with all deps. Built by CI/CD pipeline.

*Image is a weak entity, because its primary key consists of external app\_name and internal version.*

## Tracing

- Trace - is a distributed log of data, produced by multiple containers at multiple environments, connected by trace\_id.

*Trace is a weak entity, because its primary key consists of external app\_container\_id and environment\_id and internal trace\_id.*

It's necessary to accurately determine from which container and environment trace comes.

Since there are no mappings from "App container" to "Container environment" and one trace can engage "App containers" from different "Container environments" - Trace has ternary relation with both "App containers" and "Container environments". (container\_id, env\_id, trace\_id)

*Why not just a few binary?*

1. Let's assume that we have three relationships. Existed(env, container) and new: (env, trace\_id) and (container, trace\_id) - The "trace fact" is stored in two tables which leads to violation of data integrity.