# Accessing an event repository

# What is an ER?

- Event Repository
- Persistence
- Historical lookup
- Downstream / upstream

# Why do we need an ER?

- Reacting on one event is easy
  - Artifact has been created > Publish artifact
  - Artifact has been published > Start tests
- Reacting on two events is not as easy
  - Requires internal storage
  - Multiple instances make internal storage harder
  - Multiple actors in the pipeline duplicates complexity

# Why do we need an ER?

- Can also be used to answer several questions
  - Which tests did we run on this build?
  - Where have we published the build?
  - Do we have confidence to release this build?

# Event repository alternatives

- Eiffel Event Repository
  - https://hub.docker.com/r/eiffelericsson/eiffel-er
  - Proprietary
  - Eiffel community event repository schema
  - RESTful
  - Works with Eiffel REMReM & Eiffel Intelligence

# Eiffel repository alternatives

- Eiffel Goer
  - https://github.com/eiffel-community/eiffel-goer
  - Open source implementation of Eiffel Event Repository
  - RESTful
  - Will work with Eiffel REMReM & Eiffel Intelligence
  - Written in Go
  - Under development (contributions are welcome!)

# Eiffel repository alternatives

- Eiffel GraphQL API
  - https://github.com/eiffel-community/eiffel-graphql-api
  - GraphQL API
  - Written in Python
  - GraphiQL
  - MongoDB queries
  - Is NOT compatible

# Which one should you use?

- Two types
  - Official Eiffel community REST APIs
  - GraphQL API
- If using ETOS
  - GraphQL API & one official Eiffel community REST API.
  - Eiffel Goer works with the same MongoDB setup as Eiffel GraphQL API
- If not using ETOS
  - Eiffel Goer or Eiffel Event Repository