

# 1 DDMon — overview

## 1.1 Introduction

DDMon is a deadlock monitoring tool for Erlang and Elixir programs based on the `gen_server` behaviour. We described DDMon in the *Section 7* and *Appendix A* of the companion paper. This artifact supports the following claims and results from the paper:

- The main claim of *Section 7.1* about DDMon’s applicability to `gen_server`-based systems for deadlock detection is backed by `EXAMPLE.md`.
- The results from *Section 7.2* (including reproduction of *Figures 15 and 16* as well as *Listings 2, 3 and 4*) are addressed in `EVALUATION.md`.
- The scenario DSL described in *Appendix A.1* is documented in `SCENARIOS.md` together with examples.
- Encoding of replicated services shown in *Appendix B* is presented by example in `SCENARIOS.md` (section about the “`routing`” scenario\*\*).

Please find the referenced files and their PDF renders in the main artifact zip.

Note that mechanised proofs are not part of this artifact, but supplementary materials to the companion paper.

## 1.2 Hardware dependencies

To execute all benchmarks, the following minimal hardware prerequisites need to be met:

- 32GB of RAM
- 23GB of free disc space
- 8-core CPU (we have tested on 64-bit x86 and M1)

We also provide smaller variants of the largest tests. For those, the following should suffice:

- 16GB of RAM
- 10GB of free disc space
- 4-core CPU

## 1.3 Getting Started Guide

We recommend using Docker to conduct experiments. Please consult `EVALUATION.md` for instructions for building the docker image and for “kick the tires” evaluation.

## 1.4 Step-by-Step Instructions

Please find detailed instructions in `EVALUATION.md`, after the “kick the tires” section.

## 1.5 Reusability Guide

For reusability evaluation, we propose an showcase of how to use DDMon for monitoring an application based on the `gen_server` behaviour. The application is located in the `example-system` directory, and detailed instructions are to be found in `EXAMPLE.md`.