Option 1: Follow these steps to obtain heap profiles

0. Install gperftools if needed (skip this step if using the development docker), e.g., `yum install -y gperftools-devel gperftools-libs gperftools ghostscript.x86_64 gv.x86_64`
1. Compile with gperf tools: `cmake -DUSE_GPERFTOOLS=1 ../foundationdb -G Ninja; ninja` (~~may need to comment out `target_compile_definitions(gperftools PUBLIC USE_GPERFTOOLS)` in `cmake/FindGperftools.cmake`~~).
2. Run with gperftools enabled: `HEAPPROFILE=/tmp/fdbserver fdbserver [args...]`
3. Profile the heap profile: `pprof-symbolize gperf-build/bin/fdbserver /tmp/fdbserver.0065.heap`

Note that the profiling runs are at least 10X slower than the runs without profiling.

See a sample profile here.

Option 2: Use Valgrind tool massif

See massif manual.

1. Compile with Valgrind, e.g., `cmake -S ${HOME}/src/foundationdb -B ${HOME}/build_output -D USE_CCACHE=ON -D USE_WERROR=ON -D USE_VALGRIND=ON -G Ninja && ninja -C ${HOME}/build_output -j 80 fdbserver`
2. Run with massif tool, e.g., `valgrind --tool=massif ./build_output/bin/fdbserver -r simulation --crash --logsize 1024MB -f ./foundationdb/tests/fast/ConfigureLocked.toml -s 93093841 -b on`

Trace events of `GetMagazineSample` and `HugeArenaSample`

- `GetMagazineSample` logs when the fast allocators adds more magazines, the backtraces will be reliably the problem.
- `HugeArenaSample` could point to arenas that eventually get deallocated, so it might not be a memory leak.