

cs-scripts

Manual

Build and utility scripts for the napp-it CS tools - OmniOS/illumos and Solaris builds, Ollama library parser, ZFS pool benchmark

Version 1.0 · October 13, 2026

Contents

1. Overview	3
2. illumos / OmniOS builds	3
3. Solaris builds	3
4. ZFS pool benchmark	3
5. Ollama library parser	4
5.1 Flags	4
5.2 Output record	4
6. License	4

1. Overview

cs-scripts collects the build and utility scripts of the napp-it CS project family: the OmniOS/illumos and Solaris build scripts for the Rust and Go cs-tools, the standalone Ollama library parser used by the napp-it CS tools download page, and the standalone ZFS pool benchmark.

Repository layout:

```
illumos/
  rustfs_omnios_1.0_release.sh   RustFS build on OmniOS/illumos
  cs-imageindex_omnios_1a.sh    cs-imageindex build on OmniOS/illumos
  build_llamacpp_omnios.sh      llama.cpp llama-server on OmniOS
  build.rc.sh                   cargo install rustfs-cli + ipadm buffer fix
  needed_ip_modification_for_rustfs.txt
  build_restic_omnios.txt       manual restic build notes (Go)
solaris/
  build_rclone_solaris.sh       rclone from source on Solaris 11.4
  build_restic_solaris.sh       restic from source on Solaris 11.4
ollama/
  ollama-library.pl             Ollama library catalog parser
benchmark/
  benchmark_worker.pl           standalone ZFS pool benchmark
```

2. illumos / OmniOS builds

There is no GitHub Actions runner for illumos, so illumos binaries are built by hand on real OmniOS hardware with these scripts and uploaded to the matching GitHub release.

Script	Purpose
rustfs_omnios_1.0_release.sh	Canonical RustFS build (renamed from rustfs_omnios_1a.sh on 2026-09-20; the history of the 2a5..2a12 line is in the script header)
cs-imageindex_omnios_1a.sh	cs-imageindex build, including the bundled static LGPL ffmpeg; the result is self-contained
build_llamacpp_omnios.sh	llama.cpp llama-server as a local OpenAI-compatible LLM; 10 illumos patches, verified on r151058
build.rc.sh	cargo install rustfs-cli plus the ipadm buffer fix
build_restic_omnios.txt	manual restic build notes (Go)

The files illumos/readme.txt and needed_ip_modification_for_rustfs.txt describe the details and the network buffer modification RustFS needs.

3. Solaris builds

build_rclone_solaris.sh and build_restic_solaris.sh build rclone and restic from source on Solaris 11.4. Run them as a normal user on a host with the Go toolchain installed.

4. ZFS pool benchmark

benchmark/benchmark_worker.pl is a standalone pool benchmark for comparing pools ("what does THIS pool really deliver?") on any ZFS host: TrueNAS CORE/SCALE, FreeBSD, illumos/Solaris, OpenZFS on Linux and Windows. It is pure Perl and needs only perl, zfs/zpool, df and optionally smartctl - no fio. The test medium is a scratch dataset that is created on the pool under test and destroyed again at the end.

It is the same file, with the same name and content, as the napp-it worker in data/menues/_lib/scripts/bench/benchmark_worker.pl.

```
perl benchmark/benchmark_worker.pl profile=quick pool=tank
perl benchmark/benchmark_worker.pl profile=mailserver syncwrite=yes load=balanced
perl benchmark/benchmark_worker.pl check=yes pool=tank
perl benchmark/benchmark_worker.pl help=yes
```

- profile=quick takes about 1-2 minutes.
- check=yes is a dry run without any I/O.
- The numbers are cache-safe by design: primarycache=metadata (data out of the ARC, metadata still cached), sync=always for the sync-write phase, a concurrent zpool iostat -v sample per phase, and recordsize=4K for the 4k test file.
- Every result is tagged storage-bound, cache, cache-influenced or tool-limited.
- The log ends with a BENCH_DONE marker.

Details and the measured limitations are in benchmark/readme.txt.

5. Ollama library parser

ollama/ollama-library.pl reads the public Ollama model library (ollama.com/library) via curl and returns the complete model and tag catalog on demand: model names, per-tag download size, context window, input capabilities (text/image) and quantization. A GUI can use it to offer a filtered model selection for ollama pull.

The script is loaded from the repository on demand (raw.githubusercontent.com) and cached locally. The interface contract is pinned in the script header (CONTRACT cs-ollama-catalog-v1), so the calling GUI stays stable even when Ollama changes its website markup - only the fetch/parse internals below the contract header may change.

```
perl ollama/ollama-library.pl --list --json
perl ollama/ollama-library.pl --json --vision --param-max 14 --size-max 10
perl ollama/ollama-library.pl --json --vision --newer 12
perl ollama/ollama-library.pl --version
```

5.1 Flags

Flag	Meaning
--list	list models (name, capabilities, parameter sizes)
--tags [--model NAME]	per-tag catalog (all models or one model)
--json	machine-readable JSON (default; --tsv for text)
--vision	only models capable of image input
--newer MONTHS	only models updated within MONTHS (approximate)
--param-max B	only models with at most B billion parameters
--size-max GB	only tags with at most GB download size
--cache SECS	cache TTL of the catalog (default 2592000 = 30 days)
--refresh	rebuild the catalog, ignore the cache
--version	print the contract version

5.2 Output record

One record per tag:

```
{ "model": "llama3.2-vision",
  "tag": "llama3.2-vision:11b-instruct-q4_K_M",
  "size_gb": 7.8, "context": 128000,
  "input": ["text", "image"], "quant": "q4_K_M",
  "param_b": 11, "age_days": 300 }
```

6. License

BSD 2-Clause. Copyright (c) 2026 Guenther Alka / napp-it.org. Project: napp-it / csweb-gui, maintained by gea-napp-it, AI-assisted by Claude (Anthropic).