Plan: Port CI/CD Infrastructure from memories-mcp

Contents

Problem **2**

What to Port (and What Not To) **3**
Porting 4
NOT Porting 4

Slices **4**

Not Doing **7**

Architecture Notes **8**

> ***Instructions for implementing agents:*** *- Work through slices in order - Use* `/build [slice description]` *to implement each slice — this includes code review and security assessment per slice - After EACH slice: check it off (* `- [x]` *) in this file and run* `~/.claude/md2pdf.sh` *on this file to update the PDF - If a slice is larger than expected, split it further before building - If you discover new slices needed, add them with checkboxes and regenerate the PDF*

Scope: Feature

Problem

The `memories-mcp` project had comprehensive CI/CD infrastructure set up for open source — linting, vulnerability scanning, security analysis, supply-chain hardening, dependency automation, code coverage gates, and release automation. The `tor-go` project currently has only a bare `go test ./...` workflow. Porting the valuable infrastructure will improve code quality, security posture, and development workflow for tor-go.

What to Port (and What Not To)

Porting

Source (memories)	Value
Enhanced `test.yml`	Linting, vuln scanning, coverage gate, supply-chain hardened actions
`codeql.yml`	Automated security scanning with extended queries
`scorecard.yml`	OpenSSF supply-chain security scoring + badge
`dependabot.yml`	Automated dependency updates for Go modules and Actions
`CODEOWNERS`	Review requirements
`.gitignore` improvements	More comprehensive ignore patterns
README badge	OpenSSF Scorecard badge
`version` file + release workflow	Auto-versioned releases with signed binaries for `tor-client`
Version check workflows	PR version conflict detection

NOT Porting

Item	Reason
`install.sh`	Specific to memories daemon distribution model
`BIAS.md` / `.memory.md`	Memories-specific functionality
`docs/` V2 planning	Memories-specific architecture docs
Integration test step with mock API key	Memories-specific; tor-go has no equivalent tagged integration tests yet

Slices

- **F-1: Harden and enhance test.yml**
  - **What:** Upgrade the existing test workflow with supply-chain security best practices (pin actions to SHA, `persist-credentials: false`, `permissions: read-all`), add Go module caching, add golangci-lint, add govulncheck, add coverage reporting with a 50% minimum threshold (lower than memories' 70% since tor-go is newer), and add a LICENSE existence check.
  - **Files/areas:** `.github/workflows/test.yml`
  - **Acceptance criteria:**
    - All actions pinned to full commit SHAs
    - `persist-credentials: false` on checkout
    - `permissions: read-all` at workflow level
    - Go modules cached via `actions/setup-go` cache
    - golangci-lint runs against `./...`
    - govulncheck runs against `./...`
    - `go test -coverprofile=coverage.out -covermode=atomic ./...` runs
    - Coverage gate script fails the build if overall coverage (excluding `cmd/`) drops below 50%
    - LICENSE file existence check passes
    - Workflow triggers remain push to main + PRs to main
  - **Dependencies:** None
  - **Risks:** golangci-lint may flag existing code issues that need fixing first. Mitigation: if there are many lint issues, add a separate slice to fix them before enabling the lint step.
- **F-2: Add CodeQL security scanning workflow**
  - **What:** Add the CodeQL analysis workflow running on push, PR, and weekly schedule with `security-extended,security-and-quality` queries for Go. Pin all actions to SHAs and follow least-privilege permissions.
  - **Files/areas:** `.github/workflows/codeql.yml`
  - **Acceptance criteria:**
    - Workflow triggers on push to main, PRs to main, and weekly cron (Mondays 06:00 UTC)
    - `permissions: read-all` default, job-level `actions: read`, `contents: read`, `security-events: write`
    - Language matrix includes `go`
    - Query suite is `security-extended,security-and-quality`
    - All actions pinned to full commit SHAs
    - SARIF results uploaded to GitHub Security tab
  - **Dependencies:** None
  - **Risks:** CodeQL may flag existing issues. These should be triaged separately, not block the workflow addition.
- **F-3: Add OpenSSF Scorecard workflow + README badge**
  - **What:** Add the OpenSSF Scorecard workflow running on push to main, weekly, and manual dispatch. Add the Scorecard badge to the README alongside existing badges.
  - **Files/areas:** `.github/workflows/scorecard.yml`, `README.md`
  - **Acceptance criteria:**

- Workflow triggers on push to main, weekly cron (Mondays 07:00 UTC), and `workflow_dispatch`
- `permissions: read-all` default, job-level `security-events: write`, `id-token: write`
- Results published publicly (`publish_results: true`)
- SARIF uploaded to GitHub Security tab
- All actions pinned to full commit SHAs
- README.md has OpenSSF Scorecard badge linking to `scorecard.dev/viewer/?uri=github.com/cvsouth/t`

– **Dependencies:** None
– **Risks:** Low. Scorecard is read-only analysis.

- **F-4: Add Dependabot configuration**
  – **What:** Add Dependabot config for automated weekly dependency updates covering both Go modules (at repo root, since `go.mod` is at root unlike memories where it was in `src/`) and GitHub Actions.
  – **Files/areas:** `.github/dependabot.yml`
  – **Acceptance criteria:**
    - `gomod` ecosystem configured at `/` directory, weekly Monday 09:00, labeled `dependencies` + `go`, commit prefix `deps`, max 5 open PRs
    - `github-actions` ecosystem configured at `/` directory, weekly Monday 09:00, labeled `dependencies` + `github-actions`, commit prefix `ci`, max 5 open PRs
  – **Dependencies:** None
  – **Risks:** None. Dependabot PRs are informational and require manual merge.
- **F-5: Add CODEOWNERS**
  – **What:** Add a CODEOWNERS file assigning `@cvsouth` as the default reviewer for all files.
  – **Files/areas:** `.github/CODEOWNERS`
  – **Acceptance criteria:**
    - File contains `* @cvsouth`
    - GitHub recognizes it (correct path under `.github/`)
  – **Dependencies:** None
  – **Risks:** None.
- **F-6: Improve .gitignore**
  – **What:** Expand the existing `.gitignore` with patterns from memories that are relevant to tor-go: Go test artifacts (`*.test`, `*.out`, `coverage*.out`), release artifacts (`release/`, `dist/`, `*.tar.gz`, `*.zip`, `checksums.txt`), OS files (`.DS_Store`, `Thumbs.db`), editor swap files (`*.swp`, `*.swo`), config/personal files, scratch scripts, and temp dirs.
  – **Files/areas:** `.gitignore`
  – **Acceptance criteria:**
    - All relevant patterns from memories `.gitignore` are present
    - Existing tor-go patterns preserved
    - No patterns that would ignore tracked files (verify with `git status` after change)
  – **Dependencies:** None
  – **Risks:** None.

- **F-7: Add version file and release workflow**
  - **What:** Create a `version` file (starting at `0.1`) and add the release workflow that auto-increments patch versions, builds cross-platform binaries for `cmd/tor-client`, signs them with Cosign, generates SLSA provenance attestations, and creates GitHub releases. Also add the version-check and retrigger-version-checks workflows for PR conflict detection.
  - **Files/areas:** `version`, `.github/workflows/release.yml`, `.github/workflows/version-check.yml`, `.github/workflows/retrigger-version-checks.yml`
  - **Acceptance criteria:**
    - `version` file contains `0.1`
    - Release workflow waits for Test and CodeQL checks to pass before proceeding
    - Builds `tor-client` binary for `linux/amd64`, `linux/arm64`, `darwin/amd64`, `darwin/arm64`, `windows/amd64` with CGO disabled and stripped symbols
    - Version string injected via ldflags (requires adding a `Version` var to `cmd/tor-client/main.go` or similar)
    - All binaries signed with Cosign keyless signing
    - SLSA provenance attestations generated for all binaries
    - `checksums.txt` generated and signed
    - GitHub release created with auto-generated notes and all artifacts uploaded
    - Version-check workflow validates PR version bumps
    - Retrigger workflow re-runs version checks when `version` file changes on main
    - All actions pinned to full commit SHAs
  - **Dependencies:** F-1 (test.yml must exist for the release workflow to wait on it), F-2 (CodeQL must exist for the release workflow to wait on it)
  - **Risks:** The build path for tor-client needs to be correct (`./cmd/tor-client`). The ldflags version injection requires a variable in the Go code to receive it — this is a small code change. Cosign keyless signing requires the repo to have OIDC configured with GitHub (automatic for public repos, may need enabling for private repos).
  - **Not doing:** No `install.sh` — tor-go users install via `go install` or download release binaries directly.

Not Doing

- **install.sh**: tor-go doesn't need a daemon installer. Users install via `go install` or download binaries from releases.
- **Integration test step**: memories had a specific integration test tag with mock API keys. tor-go doesn't have tagged integration tests yet. Can be added later if needed.
- **CONTRIBUTING.md / CODE_OF_CONDUCT.md**: memories didn't have these either. Not in scope.
- **Custom golangci-lint config**: memories used defaults. We'll do the same — can be customized later if needed.

Architecture Notes

- All GitHub Actions must be pinned to full commit SHAs (not mutable version tags) for supply-chain security.
- `persist-credentials: false` on all checkout steps to prevent token leakage.
- `permissions: read-all` as workflow default; elevated permissions only at job level where needed.
- The `go.mod` is at the repo root in tor-go (unlike memories where it was in `src/`), so all paths in workflows should reference `/` or `./` rather than `src/`.