CPLT

ONE MAN ARMY

04 // OPERATIONS

DR Posture Summary

> Disaster recovery standard applied to every CPLT engagement

DOCUMENT	VERSION	DATE	CONTACT
CPLT-OPS-04	3.0	2026-05-24	contact@cplt.online

DR Posture Summary

WHAT THIS DOCUMENT COVERS

This summary describes the disaster recovery standard and process applied to every CPLT engagement. It demonstrates the documentation quality, recovery posture, and operational rigor we bring to client deployments.

Every engagement is custom-scoped. The components, container counts, and specific tools vary. What doesn't vary: the documentation standard, the recovery methodology, and the handover process.

The full playbook (54 pages) contains detailed recovery procedures, verification commands, and operational runbooks from a reference implementation. It is shared under NDA during scoping conversations.

ENGAGEMENT MODEL: CUSTOM-SCOPED, STANDARD-DOCUMENTED

CPLT deployments are not off-the-shelf products. Each engagement is:

- **Custom-scoped** to client requirements (document types, volume, integrations, compliance needs)
- **Custom-built** with client-specific components (MCP servers, databases, inference models, application layer)
- **Documented to the same standard** regardless of component selection

What varies by engagement:

- Container count and composition
- Specific MCP servers or tool integrations
- Database tier (which databases, sizing, schema)
- Inference layer (which models, local vs. provider mix)
- Application layer (LibreChat, custom UI, internal tools, etc.)

What doesn't vary:

- Documentation standard (operator-grade runbooks, decision records, verification tooling)
- Recovery methodology (decision trees, not checklists; verification at every stage)
- Backup architecture (layered strategy with independent failure domains)

- Handover process (full source code, IaC, runbooks; zero retention after engagement)

REFERENCE IMPLEMENTATION: THE DOCUMENTATION STANDARD IN PRACTICE

The DR Playbook documents a reference implementation that demonstrates the CPLT standard:

Reference deployment:

- Container orchestration: Docker Compose (not Kubernetes — right scale for on-prem)
- Multiple independent Compose projects with explicit decoupling
- Inference layer: LiteLLM routing 6+ providers, local model serving via llama.cpp
- Tool integration: 13 MCP servers (filesystem, web search, database, etc.) with host-network isolation
- Application: LibreChat (self-hosted ChatGPT alternative) with custom patches
- Database tier: MongoDB (document store), PostgreSQL (vector DB via pgvector), Redis (cache), Meilisearch (search)

Key architectural principles:

- All components independently restartable — no cascade failures
- Cross-project dependencies explicitly decoupled
- Single source of truth (SSOT) compiler renders all configuration from one tree
- Pre-flight checks refuse to render on drift
- Every container, service, and configuration is documented with:
  - Purpose and responsibility
  - Dependencies (what it needs, what needs it)
  - Verification method (how to confirm it's working)
  - Recovery procedure (how to fix it when it breaks)

The specific container count, tool selections, and configurations vary by engagement. The documentation standard does not.

RECOVERY TIME OBJECTIVES

RTOs are measured from detection to verified recovery. Verification is mandatory — recovery is not complete until automated checks pass.

SCENARIO	TARGET RTO	TESTED
Single container failure	< 5 minutes	✓
Multi-container service degradation	< 15 minutes	✓
Full application stack restart	< 30 minutes	✓
Data drive failure (restore from backup)	< 2 hours	✓
Complete bare-metal rebuild	< 2 hours	✓

Note on restart vs. rebuild: The restart target (30 min) assumes a healthy system being restarted in order. The bare-metal rebuild target (2 hours) assumes starting from nothing with automated scripts and verified backups. Rebuild is faster than it sounds because it's scripted end-to-end; restart is slower because it includes manual verification at each stage.

Actual RTOs depend on deployment size, backup strategy, and hardware. These targets serve as a baseline for scoping conversations.

BACKUP ARCHITECTURE

Three-layer strategy, each with independent cadence and retention:

Layer 1 — Local backups (high-frequency)

- All stateful services (databases, application state, configuration)
- Retention: 7-30 days (engagement-specific)
- Storage: Local volume, separate from application data

Layer 2 — Offsite backups (daily)

- All Layer 1 artifacts
- Encryption at rest
- Retention: 90 days
- Storage: Object storage (S3-compatible, client-specified region)

Layer 3 — Configuration & artifact version control (continuous)

- All application configuration (YAML, environment, patches)
- All infrastructure definitions (Compose files, systemd units, scripts)
- All custom-built components (Dockerfiles, patches, integrations)
- Sanitized configuration templates

- Retention: Permanent (git history)
- Storage: Version-controlled repository

Backup schedules, retention policies, and storage targets are engagement-specific. The layered strategy and independent failure domains are standard.

RECOVERY DECISION TREE

Recovery follows a diagnostic decision tree, not a linear checklist:

```
Something is wrong
|
├─ Is the application UI reachable?
|  ├─ Yes → Specific feature broken?
|  |  ├─ Yes → Isolate: which container/service?
|  |  |  └─ Check logs, restart, verify
|  |  └─ No → Performance degradation?
|  |     └─ Check inference units, resource limits
|  |
|  └─ No → Is any container running?
|     ├─ Yes → Multi-container failure?
|     |  └─ Check dependencies, restart stack
|     └─ No → Host unresponsive?
|        ├─ Yes → Boot drive failure?
|        |  ├─ Yes → Full bare-metal rebuild
|        |  └─ No → Data drive failure?
|        |     ├─ Yes → Restore from backup
|        |     └─ No → Configuration drift?
|        |        └─ Re-apply from version control
|        └─ No → Secrets compromise?
|           └─ Rotate all credentials
```

Each branch terminates in a specific recovery procedure with:

- Explicit verification steps (not just "check if it works")
- Pre-flight conditions (must pass before proceeding)
- Post-recovery validation (automated checks)

The decision tree structure is standard. The specific containers, services, and verification commands are engagement-specific.

VERIFICATION AND STALE-CHECK

The playbook includes automated verification tooling:

Pre-recovery checks:

- Container count matches expected inventory
- Configuration files match documented state
- Backup freshness (within engagement-specific threshold)
- Offsite backup reachability
- Secrets bundle integrity

Post-recovery checks:

- All services reachable and responding
- Database connections established
- Inference units accepting requests
- Tool integrations functional
- Performance within expected bounds

Drift detection:

- Stale-check script runs before any recovery procedure
- Compares current state against documented state
- Fails hard if drift exceeds tolerance (forces playbook update before execution)
- Warns on soft drift (e.g., retired containers, updated versions)

The tooling is generated from the engagement's source of truth. When components change, the verification suite is regenerated — not manually updated.

RECOVERY SCENARIOS COVERED

The full playbook covers seven recovery scenarios in detail:

1. **Single container failure** — Isolate, restart, verify
2. **Multi-container service degradation** — Dependency analysis, ordered restart
3. **Stack-wide recovery** — Full shutdown/startup sequence
4. **Data drive failure** — Restore from local backup
5. **Full bare-metal rebuild** — OS reinstall through application restore
6. **Secrets compromise** — Credential rotation, access revocation
7. **Data loss event** — Restore from offsite backup, verify completeness

Each scenario includes:

- Diagnostic decision tree (not linear checklist)

- Explicit pre-flight checks (must pass before proceeding)
- Step-by-step recovery with verification at each stage
- Post-recovery validation (automated, not manual)
- Common failure modes and workarounds

The scenarios are standard. The specific procedures, commands, and verification methods are engagement-specific.

DOCUMENTATION STANDARD

Every CPLT engagement ships with documentation at this standard:

Operator-grade runbooks:

- Step-by-step procedures with explicit verification
- Decision trees, not linear checklists
- Pre-flight checks before destructive operations
- Post-recovery validation (automated, not manual)

Decision records:

- Why each component was chosen
- What alternatives were rejected
- Tradeoffs accepted
- Migration paths if requirements change

Verification tooling:

- Automated checks for documented state
- Drift detection before procedures execute
- Post-recovery validation scripts

Recovery procedures:

- Tested and verified (not theoretical)
- RTO targets measured from detection to verification
- Layered backup with independent failure domains

Source of truth (SSOT) compiler:

- Single configuration tree renders all containers, services, scripts

- Pre-flight checks refuse to render on drift
- Verification tooling generated from the same source
- When configuration changes, documentation is regenerated — not manually updated

---

HOW TO ACCESS THE FULL PLAYBOOK

The full 54-page DR Playbook is available during scoping conversations:

1. **Submit a scope request** — cplt.online/contact
2. **Initial conversation** — We'll discuss your requirements and stack
3. **NDA execution** — Standard mutual NDA
4. **Playbook access** — Full document shared under NDA

The playbook is part of every CPLT engagement deliverable, not a standalone product.

WHAT'S NOT IN THIS SUMMARY

This document intentionally omits:

- Specific container names and configurations
- Exact backup schedules and retention policies
- Detailed recovery commands and scripts
- File system paths and permissions
- Systemd unit definitions
- Database connection strings and credentials
- Provider-specific configuration
- Exact hardware specifications
- Verification script contents
- SSOT compiler structure and source tree

These details are in the full playbook, shared under NDA during scoping conversations.

This document is public. Share freely. The full playbook is not.