ANVCS — AI-Native Version Control System (Core Spec v0.4)

Subtitle: Offline-First, Disconnected CRDT Mode, Secure Remote Connectivity, and Enterprise Resilience

> This v0.4 release consolidates the core ANVCS design and **integrates the new CRDT-disconnected (offline) mode**, plus CLI/UX flows, security, and testability guarantees for air-gapped environments. It is intended to replace earlier versions as the **main document**.

---

0) Executive Summary

ANVCS is a **Git-compatible, AI-native version control system** that combines **CRDT live collaboration**, a **commit DAG**, **semantic merge**, and **rich AI provenance** (prompts, model versions, tests, artifacts). New in v0.4: a formal **offline/CRDT-disconnected mode** that maintains **full parity** with connected workflows, ensuring developers and agents can work **without network access**, commit locally, and **rejoin** later without losing context or integrity.

Core value - **Git immutability + compatibility** preserved (content-addressed commits; push/pull; CI unaffected) - **CRDT live edits** across agents with conflict-free convergence - **Semantic merges** at commit boundaries (AST/test/LLM-assisted) - **Provenance** captured per commit (prompt, model, tests, artifacts) - **Offline-first parity**: connected and disconnected experiences behave **the same** - **Pluggable artifact storage** (S3 versioning/IPFS/NFS), exact version pinning

---

1) Problem Statement (recap)

- Git was optimized for human-paced teams, not 24/7 swarms of agents. Merge conflicts, history opacity, and cross-repo drift become severe.
- Non-text assets (models, datasets) lack reproducibility without external pinning.
- Enterprises need **offline/air-gapped** workflows while retaining auditability and consistency.

---

2) Goals & Non-Goals

Goals - Preserve Git **immutability**, **content addressability**, and **interop**. - Enable **multi-agent** concurrency with low friction. - Make **intent traceable**: prompts, models, test outcomes per commit. - Support **any file type** via **version-aware storage adapters** (S3 VersionId, IPFS CID, etc.). - Provide **offline-first parity**: commits and metadata identical on/offline. - Secure, observable, testable in **air-gapped** and **cloud** settings.

Non-Goals (v0.4) - Replacing Git hosting; blockchain consensus layers; automatic resolution of all semantic conflicts.

3) Core Principles

1. **Git-Compatibility First** — Dual-write to `.git/` on every commit; Git notes/structured messages for AI metadata.
2. **Immutability & Reproducibility** — Append-only DAG; content-hash; pinned artifacts `{key, version_id, sha256}`.
3. **CRDT for Edits, DAG for Versions** — Ops converge live; commits snapshot coherent states.
4. **Provenance as First-Class** — Capture prompt/model/tests/env/artifacts for every commit.
5. **Semantic Merge** — AST/test-guided merges at DAG boundaries.
6. **Offline-First Equivalence** — Connected and disconnected modes **produce identical artifacts and metadata**.
7. **Pluggable Everything** — Storage, transports, merge strategies, policies, identity.

4) Conceptual Model

- **CRDT Room** — Operation-based per-file logs; edits stream among participants when connected.
- **Commit (Semantic Snapshot)** — Agents/devs take snapshots at task boundaries; creates DAG node + Git commit with AI metadata.
- **DAG** — Directed acyclic graph with multi-parent merges; branches/streams are just **named pointers**.
- **Agent Context** — Structured metadata: prompt, model, tests, coverage, env, artifacts.
- **Artifact Store** — Offloads large binaries via adapters (S3/IPFS/NFS) with exact version pinning.

5) Architecture Overview

```
┌──────────────────────────────────────────────────────────────────────────┐
│              IDEs / Agents / CLI (local or Codespaces / remote)          │
│  VS Code ext (CRDT cursors) • Agent runners • ai-vcs CLI • Hooks & CI    │
└──────────────────────────────────────────────────────────────────────────┘
               │ CRDT ops (WS/WebRTC/gRPC) • Offline journal when disconnected
┌──────────────▼────────────┐    ┌─────────────────────────────┐
│  CRDT Collaboration Core  │    │     Commit DAG Manager      │
│ (op logs; offline journal)│    │  (graph store, refs/streams)│
└──────────────┬────────────┘    └──────────────┬──────────────┘
               │                                │
        live buffers / ops                 DAG snapshots
               │                                │
┌──────────────▼────────────┐    ┌──────────────▼──────────────┐
│  Semantic Merge Engine    │    │  Provenance & Metadata Store│
│ (AST/test/LLM strategies) │    │ (prompts, models, tests, env)│
└──────────────┬────────────┘    └──────────────┬──────────────┘
```

```
              │                                  │
         merged diffs                    commit annotations
              │                                  │
┌─────────────▼─────────────┐      ┌─────────────▼─────────────┐
│   Git Compatibility Layer  │      │   Artifact Storage Adapters │
│  (.git objects/notes/refs) │      │ (S3 ver., IPFS, NFS, local) │
└─────────────┬─────────────┘      └─────────────┬─────────────┘
              │                                  │
       push/pull over Git                upload/download blobs
              │                                  │
       Git remotes / GitHub           S3/MinIO/IPFS/NFS endpoints
```

6) Data Model

Commit object (concept)

```
commit_id: <hash>
parents: [<hash>, ...]
author: { name, email, time }
message: |
  Summary line

  ai-meta:
    prompt: "..."
    model: "..."
    tests_passed: true
    test_report_ref: ".ai/meta/<id>/tests.json"
    coverage: 87.1
    env: { python: "3.12", node: "24", os: "ubuntu-24.04" }
    agent_id: "agent-foo"
    artifacts:
      - type: model
        storage: s3
        bucket: anvcs-artifacts
        region: eu-west-1
        key: models/agent_v1.pt
        version_id: 3HL4IxNZA...
        sha256: 4f34...9a1c
        size: 104857600
```

CRDT ops (per file) — operation journal (insert/delete/replace) with lamport/vector clocks and stable op IDs; compacted over time.

DAG store — append-only nodes + edges; refs map; cross-repo meta-commit manifests (optional).

---

7) CRDT-Disconnected (Offline) Mode — Local-First & Air-Gapped Resilience

Design objective: When CRDT connectivity is unavailable, **nothing breaks**. Edits apply to a **local CRDT journal**, and every `ai-vcs commit` produces the **same** DAG node + Git commit + AI metadata as if online.

Key guarantees - **Parity:** Connected vs disconnected commits are **identical in structure** (Git commit + `.ai/meta` schema + artifact pins). - **Dual-write:** Each commit always writes to `.git` (objects/refs) and `.ai/` locally. No daemon required. - **No special commands:** Continue using `ai-vcs commit`. If disconnected, networking is skipped; commit completes locally. - **Rejoin later:** `ai-vcs rejoin-room` (or reconnect) publishes buffered ops/commits, merges DAGs, hydrates peers. - **Air-gapped ready:** Works fully offline with local storage (MinIO/NFS) and private Git remotes; CI can run on isolated networks.

Workflow 1) Developer/agent edits files → ops appended to **local CRDT journal**. 2) `ai-vcs commit` → snapshot current state → **DAG node** + **Git commit** + **AI metadata** are created locally (identical schema). 3) Artifacts matched by `blobs.track` are stored via **local adapter** (e.g., MinIO) and pinned by `{key, version_id, sha256}`. 4) On reconnect: publish ops; pull peers' ops; CRDT ensures convergence; semantic merge resolves overlapping edits at DAG boundaries.

Visibility & merge when rejoining - Offline branches appear as separate DAG heads; semantic merge can combine them. - All provenance (prompts/tests) is preserved for review.

Testability - Provide offline test scenarios: disable network; ensure `commit`, `log`, `graph`, `checkout` work; artifact hydration uses only local endpoints; rejoin test validates convergence and merge.

---

8) Semantic Merge (recap)

- **AST-aware structured diff/merge** (renames/moves/extractions understood);
- **Guards** with typechecks/tests/linters;
- **LLM-assisted proposals** for tricky overlaps (gated by tests);
- **Non-code** drivers for JSON/YAML, notebooks, IDLs.

---

9) Storage Adapters (Any File Type)

- **Interface**: `upload() → {key, version_id, sha256, size}`, `download(key, version_id)`, `exists()`, `delete_version()`.
- **S3 Versioning**: Stable key; commit pins `VersionId`; exact retrieval on checkout; supports MinIO/Spaces.

- **Local/Hybrid modes**: Cache under `.ai/blobs/`; defer sync; resilient retries; proxy/env awareness.

Config (`ai-vcs.config.yaml`)

```
storage: { mode: hybrid, primary: s3, fallback: local }
adapters:
  s3:
    bucket: ${AI_VCS_S3_BUCKET}
    region: ${AWS_REGION}
    endpoint_url: ${S3_ENDPOINT_URL:-}
    use_presigned: true
blobs:
  track: ["*.pt","*.onnx","datasets/**/*.csv","**/*.ipynb","**/*.png","**/
*.mp4"]
  upload_on_commit: true
  compress: true
```

10) Git Interop Strategy

- **Dual-write** every commit to Git (tree/parents) + `.ai/meta` (or Git notes);
- **Refs** map DAG heads to Git branches;
- **Push/Pull** use normal Git remotes; metadata rides via notes or repo files.

11) Security, Identity, Governance

- **Commit signing** (Ed25519/GPG); per-agent keypairs; short-lived tokens; OIDC/STS for storage.
- **Scopes**: `read|write|commit|merge`; path allow/deny policies.
- **Audit**: immutable logs of snapshot/merge decisions; provenance captured per commit.

12) CLI, API & VS Code UX

CLI (selected)

```
ai-vcs init
ai-vcs collab start|join <room>     # optional; not required to commit
ai-vcs add <paths>
ai-vcs commit -m "feat: X"
  --prompt "Implement X" --model gpt-4o --test-report ./.ai/meta/tests.json
ai-vcs merge <src> into <dst> [--auto --test-check]
ai-vcs graph --since 2w --filter agent:foo
```

```
ai-vcs rejoin-room               # publish buffered ops/commits when back
online
ai-vcs push|pull origin main
```

VS Code extension - **Live cursors & diff heatmap** (when connected) - **Preview → Confirm** for snapshots (semantic summary, risk flags) - **Playback timeline** (op journal), **DAG view**, **Artifacts pane** - **Offline badge** with graceful degradation (local only)

Agent Streaming API (summary) - WebSocket/WebRTC/gRPC ops; `snapshot` to persist commit; `artifact` announcements; acks/backpressure; auth scopes.

---

13) Deployment Topologies (Secure & Flexible)

Topology	When	Pros	Cons
Local-only	Solo/offline	Zero infra, fully offline	No live collab
Ephemeral relay (devcontainer)	Small teams/ Codespaces	Easy, low-latency	Process lifecycle mgmt
WebRTC P2P	Cross-site agents	E2E, low relay cost	NAT/TURN complexity
Managed relay cluster	Enterprises	SSO, quotas, audit	Operate service

Connectivity: TLS/DTLS/mTLS; JWT/OIDC/SSH-sig; ICE (STUN/TURN); proxy envs respected.

---

14) Observability & Debuggability

- **Metrics**: ops/sec, latency, queue depth, merge success, artifact hydration.
- **Session journal**: join/leave, batch IDs, snapshot/merge attempts, test outcomes.
- **Time-travel replay**: rebuild worktree at op index/timestamp.
- **Tombstone GC** for CRDT stores; bounded logs.

---

15) Multi-Repo & Cross-Modality (optional)

- **Meta-commit groups**: sign/pin coordinated commits across repos; CI validates group.
- **Modality-aware diffs**: JSON/YAML (schema), notebooks (cell graph), IDLs (symbols), media manifests.

---

16) Test Matrix (incl. Air-Gapped)

- **Offline unit/integration**: no network; ensure `commit/log/graph/checkout` pass; artifacts served by local endpoints only.
- **Rejoin convergence**: create divergent offline branches; rejoin; assert CRDT convergence + semantic merge correctness.
- **Codespaces + proxy**: ensure adapter respects `HTTPS_PROXY/NO_PROXY`.
- **S3 versioning**: verify `VersionId` pinning, exact hydration on checkout.

---

17) Roadmap

- v0.5: cross-repo meta-commit manifests; policy DSL; extended modality drivers.
- v0.6: P2P hardening; delta-sync optimizations; fine-grained blame w/ provenance UI.

---

18) Appendix — Interfaces & Schemas (extract)

BlobStore (version-aware)

```
class BlobStore(ABC):
    def upload(local_path: str, key: str) -> Dict: ...
    def download(key: str, version_id: Optional[str], dest_path: str) ->
None: ...
    def exists(key: str, version_id: Optional[str] = None) -> bool: ...
    def delete_version(key: str, version_id: str) -> None: ...
```

S3VersionedBlobStore — returns `{key, version_id, sha256, size}`; supports MinIO/Spaces via `endpoint_url`.

Agent Streaming (WS) — envelope

```
{ "type": "op|ack|snapshot|artifact|presence|heartbeat|error", "room": "...",
"agent": "...", "seq": 512, "data": { ... } }
```

Commit metadata (JSON)

```
{
  "commit_id": "abc123...",
  "ai_meta": {
    "prompt": "Implement validation",
    "model": "gpt-4o-2025-10",
    "tests_passed": true,
    "coverage": 91.2,
    "agent_id": "agent-validate",
    "artifacts": [
      {"storage": "s3", "bucket": "anvcs-artifacts", "region": "eu-west-1",
"key": "models/validator.pt", "version_id": "3HL4IxNZ...", "sha256":
"4f34...9a1c", "size": 104857600 }
    ]
  }
}
```

Summary — v0.4 elevates ANVCS to a **resilient, offline-first** system: identical commit semantics on/offline, secure remote connectivity, and enterprise-grade observability — all while retaining Git compatibility and adding AI-native collaboration.