TRD NETWORK

TECHNICAL WHITEPAPER · V1.0 · JUNE 2026

TRD Network

Architecting the Future of AI & DePIN

Decentralized GPU compute, virtual machines, sovereign storage, and autonomous agents — on infrastructure the network owns end to end, with a cryptographically verifiable receipt for every action.

GPU Cloud | VM Instances | Rent & Earn | Storage | TRD Mind | TRD Mart

TRD Network
Dubai, UAE · trdn.io · architecting the future of AI & DePIN

Abstract

TRD Network is decentralized infrastructure for artificial intelligence. It unifies four kinds of resource — **GPU compute**, **virtual machines**, **sovereign storage**, and **autonomous agents** — into a single, two-sided network: one side rents capacity, the other supplies it. A renter spins up an H200 cluster or a full Linux VM in seconds; a provider plugs in an idle GPU and earns on verified work. Between them, a dispatch layer matches supply to demand on benchmark tier, real-time reputation, and sovereignty constraints.

What distinguishes TRD from a conventional cloud is **verifiability**. Every unit of work — a GPU-minute, a VM action, an agent task, an object write — emits a cryptographically signed **R+2 receipt** (Ed25519 over RFC 8785 canonical JSON). Arguments and results are hashed, never stored raw, so receipts are independently verifiable without leaking sensitive inputs. Billing is not a black box; it is an auditable chain.

This document describes the network's architecture, the four resource products plus the agent layer (**TRD Mind**) and its marketplace (**TRD Mart**), the credit economics that align renters and providers, and the receipt standard that makes the whole system auditable. Rates and figures are illustrative; live pricing and availability are shown in the dashboard.

Who this is for

Founders building AI products who need GPU economics they can model; operators with spare hardware who want to monetize it; and compliance teams who need provenance for AI actions. No prior knowledge of DePIN is assumed.

Contents

SECTION 1

Introduction — AI's compute problem

The cost of intelligence is the cost of compute. Training and serving modern models consumes GPUs at a scale that a handful of hyperscalers cannot supply cheaply or evenly. Capacity is concentrated, queues are long, prices carry a platform markup, and egress fees punish the very data-movement that AI workloads depend on. Meanwhile, an enormous pool of capable hardware — gaming rigs, prosumer workstations, under-utilized datacenter clusters — sits idle.

TRD Network closes that gap with a **DePIN** (Decentralized Physical Infrastructure Network) model: pool distributed hardware, match it to demand programmatically, and settle in real time. The result is GPU compute at the underlying hardware rate — no hyperscaler markup — and a new income stream for anyone with silicon to spare.

The three problems we set out to solve

- **Access.** Spin up datacenter and prosumer GPUs (RTX 3060 → NVIDIA H200) in seconds, billed by the second, with no queue.
- **Utilization.** Turn idle hardware into income by routing real, paid jobs to it — with payouts computed from verified usage.
- **Trust.** Make every transaction auditable. A receipt for every GPU-minute means the bill, and the provenance of every AI action, can be checked by anyone.

SECTION 2

Network architecture

TRD is a two-sided market with a dispatch layer in the middle. **Renters** submit workloads; **providers** contribute hardware; the **scheduler** places each workload on a matching node and opens a metered session. Each session streams telemetry and, on completion, emits a signed receipt that both parties — and any third party — can verify.

Layers

- **Providers.** Operators run a worker agent that benchmarks the machine, advertises availability, and executes scheduled jobs in isolated containers or VMs.
- **Dispatch / scheduler.** A matching service that selects a node per workload using a reverse-auction over benchmark tier, live reputation, region, and sovereignty constraints — typically in tens of milliseconds.
- **Settlement.** Real-time credit accounting; the renter is debited, the provider credited, and a receipt is chained to the provider's prior receipt.
- **Verification.** The R-Series receipt layer signs and hash-chains every action; anyone can re-verify offline.

- **Control plane.** A unified dashboard and API (api.trdn.io) expose GPU, VM, Earn, Storage and Agent operations behind one authenticated surface.

The same network underlies all products: a GPU job, a long-running VM, an object write, and an agent task are all *sessions* on the same fabric, differing only in lifetime and the resource they meter.

SECTION 3

GPU Cloud

TRD GPU Cloud gives direct access to datacenter and prosumer GPUs across the provider network. You get bare-metal performance with cloud convenience: launch a container or a full VM, attach NVMe scratch, and start training or serving in under a minute. NVLink multi-GPU nodes are available for large models; billing is per-second with idle auto-stop.

Representative catalog

GPU	VRAM	Best for	From (illustrative)
NVIDIA H200	141 GB HBM3e	Frontier training / 70B+ inference	$3.49/hr
NVIDIA H100 80GB	80 GB HBM3	LLM training & serving	$2.49/hr
NVIDIA A100 80GB	80 GB	Training, large-batch inference	$1.39/hr
NVIDIA L40S	48 GB	Inference, fine-tune, render	$0.99/hr
RTX 4090	24 GB	Fine-tune, diffusion, dev	$0.44/hr
RTX 3090	24 GB	Inference, hobby training	$0.34/hr
RTX 4070	12 GB	7–13B inference, dev	$0.24/hr
RTX 3060	12 GB	Entry inference / testing	$0.14/hr

Launch flow

A four-step launcher mirrors the dashboard: pick a **use-case** (the matching GPUs highlight), choose a **GPU**, select a **template** (PyTorch, vLLM, ComfyUI, Ollama — or bring your own image) and **cluster size** (×1–×8), then deploy. Per-second billing means you pay only for the seconds the instance runs.

SECTION 4

VM Instances

Not every workload is a batch job. TRD VM Instances are full Linux virtual machines — your kernel, your packages, your services — with root SSH, cloud-init, persistent NVMe volumes, snapshots, and private networking. Run a database, an API, a CI runner, a game server, or a GPU workload that must stay up. CPU- or GPU-backed, billed by the second.

Representative shapes

Size	vCPU	RAM	NVMe	GPU option	From
vm.small	2	4 GB	40 GB	—	$0.012/hr
vm.standard	4	16 GB	120 GB	—	$0.038/hr
vm.compute	8	32 GB	240 GB	optional L40S	$0.072/hr
vm.gpu	12	64 GB	500 GB	RTX 4090 / A100	$0.49/hr
vm.xl	32	256 GB	2 TB	multi-GPU	$1.90/hr

Configure-your-VM & Cost Guard

The VM configurator projects cost live as you change GPU, cluster size, disks and runtime. Its **Cost Guard** function lets you set a spend cap: when the projected total would exceed the cap, TRD auto-stops the VM at the cap — protecting unattended jobs from runaway cost. Specs (vCPU, RAM, storage, interconnect, NVLink) scale with cluster size.

SECTION 5

Rent & Earn — the provider side

TRD is two-sided by design. Providers install the **trd-worker** agent, run a one-time benchmark that assigns a tier rating, and join the pool with a default earn rate. From then on the scheduler routes paid jobs to the machine; each completed job emits a tamper-evident receipt, and payouts are computed from **receipt-verified usage** — not a self-reported counter. Providers set availability windows and price floors, and withdraw earnings anytime.

Earn tiers (illustrative)

Tier	Hardware class	Workload	Rate
Entry	RTX 3060 · 12 GB	~7B model class	40 cr/hr
Mid	RTX 4070 / 4090	13–34B class	95 cr/hr
High	A100 / L40S	70B class	240 cr/hr
Datacenter	H100 / H200	frontier workloads	520 cr/hr

SECTION 6

Sovereign Storage

TRD Storage is S3-compatible object storage on the decentralized network. Choose a replication tier, hold your own keys, and receive continuous integrity audits. Every object write is replicated across independent regions before the write is acknowledged, and each write carries a signed receipt with the

durability window baked in — up to a 99-year permanence commitment on the cold tier.

Replication tiers (illustrative, per GB·month)

Tier	Durability	Price
Standard	3× replication, hot reads, S3 API	$0.006
Durable	Geo-redundant, integrity audits, receipts	$0.011
Permanent	99-year window in-receipt, cryptographic erasure	$0.029

- S3-compatible API & SDKs — point existing tooling at the TRD endpoint, no rewrite.
- Continuous hash audits across replicas; drift is detected and self-healed.
- Cryptographic erasure on request, satisfying data-deletion obligations (e.g. GDPR Art. 17) while the immutable receipt of the action remains.

SECTION 7

TRD Mind — autonomous agents

TRD Mind is the agent layer: it orchestrates autonomous agents across the network's compute, storage and economy. Agents monitor data, run workflows, transact with one another, and act on a schedule — and because they run on TRD, every action an agent takes emits a verifiable R+2 receipt. Provenance for agent behavior is built in, not bolted on.

Capabilities

- **Orchestration.** Compose agents from skills and personas; run them on TRD GPUs/VMs with persistent, sovereign memory.
- **Verifiable actions.** Each tool call, memory write and settlement is signed and chained — an auditable trail of what an agent did, when, and on which inputs (hashed).
- **Sovereign memory.** Long-term agent memory with semantic search, exportable as a signed stream for backup or regulator audit.
- **Economy.** Agents hold balances and settle with each other through the credit ledger, each settlement receipted.

SECTION 8

TRD Mart — the agent marketplace

TRD Mart is the marketplace for the agent economy: a place where anyone can list an agent — a persona, a skill, or a complete workflow — for others to deploy. Buyers get agents that run on verifiable infrastructure; builders get distribution and a receipted settlement rail. Listings span **Personas** (ready-to-run agent identities), **Skills** (capabilities agents can install), and **Workflows** (multi-step automations).

TRD Mart is launching; this section describes intended scope. Every purchase and deployment settles through the same credit ledger and emits a receipt, so the provenance guarantees that apply to compute apply to the marketplace as well.

SECTION 9

Credit economics

All usage is denominated in **credits**, a unit of account used uniformly across GPU, VM, storage, agents and the marketplace. Renters spend credits; providers earn them; the network retains a transparent spread. There are no subscriptions, no minimum commitments, and no markup on the underlying hardware rate — the catalog rate is the rate.

How a transaction settles

- A renter starts a session; credits are metered per second of real usage.
- On completion the provider is credited the bulk of the spend; the network retains a small spread to operate dispatch, settlement and verification.
- Both legs are written to the credit ledger and the action is sealed with an R+2 receipt.
- Providers withdraw earned credits on their own schedule.

Because settlement is receipt-anchored, the economics are auditable end to end: the sum a provider is owed is derivable from signed receipts, not from a dashboard figure that must be taken on faith.

SECTION 10

R+2 receipts & verifiability

The receipt standard is the spine of the network. Each action produces an **R+2 receipt**: a JSON object carrying the agent's public key, an action type, hashed action data, a timestamp, a per-chain nonce, and a pointer to the previous receipt — signed with Ed25519 over RFC 8785 (JCS) canonical JSON. Receipts form a per-agent hash chain.

Properties

- **Privacy-preserving.** Arguments and results are hashed, never stored raw, so a receipt proves an action occurred without exposing its contents.
- **Independently verifiable.** Anyone can recompute the canonical form, check the signature against the public key, and follow the chain pointer — offline, with no call back to TRD.
- **Tamper-evident.** Flip any field and the signature fails; break the chain and the mismatch is detectable.
- **Exportable.** A range of receipts can be bundled into a Merkle audit export (with an optional signed PDF certificate) for regulators or counterparties.

Illustrative receipt body (signature omitted):

```
{ "spec_version": "r2/v0.1", "agent_pubkey": "<base64url>", "action_type":
"gpu/session", "action_data": { "args_hash": "sha256:…" }, "occurred_at":
"2026-06-03T09:42:25Z", "prev_receipt_cid": "sha256:…", "nonce": "<96-bit>" }
```

SECTION 11

Security, sovereignty & roadmap

Security & sovereignty

- **Isolation.** Workloads run in isolated containers or VMs on provider nodes; tenants do not share process space.
- **Key custody.** Signing keys back the receipt layer; production custody moves to HSM-backed providers without changing receipt logic.
- **Jurisdictional routing.** Sovereignty constraints can pin workloads to regions, supporting data-residency requirements.
- **Your keys, your data.** Storage is sovereign by default — customers hold keys; cryptographic erasure is supported.

Roadmap (indicative)

- Deeper dashboard wiring of live network telemetry across all products.
- Expansion of TRD Mart from launch listings to an open marketplace with receipted settlement.
- Hardening of the verification path: HSM key custody, multi-instance rate limiting, and an external security audit.
- A reference zero-knowledge proving path for selective disclosure once the trusted-setup prerequisites are met.

SECTION 12

Dispatch & scheduling — the reverse auction

The dispatch layer is the heart of the network: for each incoming workload it must choose, in tens of milliseconds, the provider node that best satisfies the renter's constraints at the lowest verifiable cost. TRD models this as a **reverse auction** — providers continuously advertise capacity and a floor price; the scheduler runs a single-round sealed selection per workload rather than a live bidding war, which keeps placement fast and deterministic.

Inputs to a match

- **Requirements.** GPU model/VRAM, cluster size, vCPU/RAM, region, and template (the renter's request).
- **Benchmark tier.** Each provider node is rated by the one-time `trd-worker` benchmark; the tier bounds which workloads it may accept.

- **Reputation.** A rolling score from completion rate, uptime, and receipt-verified delivery — providers that fail jobs decay quickly.
- **Sovereignty constraints.** Hard filters that pin a workload to permitted regions/jurisdictions for data-residency.
- **Price floor.** The provider's minimum acceptable rate; the catalog rate is derived from the winning floor plus the network spread.

Selection, step by step

- **Filter.** Drop nodes that fail any hard constraint (capability, region, availability window).
- **Score.** Rank survivors by a weighted function of price floor, reputation, and locality to the renter's data.
- **Place.** Award the workload to the top-ranked node and open a metered session; ties break on reputation, then random.
- **Fallback.** If the awarded node fails to accept within a deadline, the scheduler re-awards to the next candidate — the renter never sees the failure.

Placement targets the low tens of milliseconds. Because selection is a pure function of advertised state, the same inputs yield the same placement, and the decision itself is recorded so a dispute can be reconstructed after the fact.

SECTION 13

R+2 receipt specification

This section specifies the receipt format more precisely. A receipt is a flat JSON object; it is signed over its canonical form and identified by a content hash that also serves as the chain pointer for the next receipt.

Receipt fields

Field	Type	Description
spec_version	string	Format version, e.g. "r2/v0.1".
agent_pubkey	string	base64url Ed25519 public key of the signer.
agent_id	string	Stable identifier/slug for the acting agent.
action_id	string	UUID v4 unique to this action.
action_type	string	e.g. gpu/session, vm/action, memory/store, economy/settle.
action_data	object	Action payload — sensitive values are hashed (args_hash), not raw.
occurred_at	string	RFC 3339 UTC timestamp.

Field	Type	Description
prev_receipt_cid	string|null	CID of the agent's previous receipt; null at genesis.
nonce	string	Per-chain unique 96-bit base64url value.
signature	string	base64url Ed25519 signature over the canonical unsigned body.

Canonicalization & signing

- The signable body is the receipt with `signature` removed.
- It is serialized with **RFC 8785 (JCS)** — object keys sorted, no insignificant whitespace, deterministic number/string forms — yielding identical bytes on any platform.
- The signer computes `signature = Ed25519_sign(JCS(body), secret_key)` and attaches it.

Content identifier (CID) & chaining

The CID is computed over the *unsigned* canonical body, not the signed receipt: `cid = "sha256:" + hex(SHA-256(JCS(body)))`. Each new receipt sets `prev_receipt_cid` to the agent's latest CID, forming a per-agent hash chain. Because the CID excludes the signature, an independent verifier recomputes exactly the value used for chaining.

Verification algorithm

- Remove `signature`; recompute `JCS(body)`.
- Check `Ed25519_verify(signature, JCS(body), agent_pubkey)` — fail ⇒ invalid.
- Recompute the CID and confirm it matches the referencing receipt's `prev_receipt_cid`.
- Optionally walk the chain to genesis; a Merkle root over the CIDs yields a compact audit anchor.

Illustrative verification (pseudocode):

```
body = receipt − { signature }
valid = ed25519_verify(receipt.signature, jcs(body), receipt.agent_pubkey)
cid = "sha256:" + sha256_hex(jcs(body))
```

SECTION 14

Platform API surface

All products are driven through one authenticated control plane (api.trdn.io). Reads are public where receipts are meant to be independently verifiable; writes require a bearer API key and are rate-limited. The surface below is representative of the receipt/provenance layer that underpins the products.

Method	Path	Purpose
GET	/v1/health · /version · /stats	Service status & metadata (public).
POST	/v1/auth/signup	Issue a free-tier API key (public).

Method	Path	Purpose
POST	/v1/memory/store	Store a memory; embeds + emits an R+2 receipt.
POST	/v1/memory/search	Semantic top-k search over agent memory.
GET	/v1/receipts/{id}	Fetch a signed receipt (public).
POST	/v1/receipts/verify	Verify any R+2 receipt (public).
GET	/v1/receipts/chain	Walk an agent's hash chain + Merkle root.
GET	/v1/agents · /agents/{id}	List / public identity lookup.
POST	/v1/economy/settle	Settle credits between agents; receipted.

Authentication is a bearer key (`Authorization: Bearer <key>`); keys are stored only as salted hashes. Rate limiting is per-key with standard `429` + `Retry-After` semantics.

SECTION 15

Security model & threat analysis

Trust boundaries

- **Tenant isolation.** Workloads run in isolated containers or full VMs; tenants never share process space, and providers are contractually barred from inspecting tenant workloads.
- **Key custody.** Signing keys back the receipt layer. The issuing logic is custody-agnostic, so production deployments move keys to HSM-backed providers without changing receipt semantics.
- **Privacy by hashing.** Receipts carry hashes of arguments/results, so provenance is provable without exposing inputs.

Threats & mitigations

Threat	Mitigation
Forged receipt	Ed25519 signature over JCS canonical bytes; verification is public.
Receipt tampering	Any field change breaks the signature; CID mismatch breaks the chain.
Chain re-ordering / omission	prev_receipt_cid links each receipt; gaps and re-orders are detectable.
Nonce replay	Per-chain nonce uniqueness enforced at issuance.
Provider misreport of usage	Payouts compute from receipt-verified work, not self-reported counters.
Credential theft	Bearer keys stored as hashes; rotate/revoke on suspicion; least-privilege scopes.

Out of scope for the current prototype and described as future work: external security audit, HSM key custody in production, Redis-backed rate limiting across instances, and any production zero-knowledge

proving path.

SECTION 16

Performance & comparison

The figures below are **illustrative** and describe design intent, not a measured benchmark report. A load-test report with reproducible methodology will accompany general availability.

Dimension	Centralized cloud	TRD Network
GPU price	List + platform markup	Underlying hardware rate (reverse auction)
Provisioning	Queue / quota	Seconds, per-second billing
Egress	Metered, often costly	No egress traps
Billing transparency	Invoice you must trust	Signed receipt per metered unit
Provider side	Closed	Open — anyone can supply hardware and earn
Auditability	Vendor logs	Independently verifiable R+2 chain

SECTION 17

Glossary

Term	Definition
DePIN	Decentralized Physical Infrastructure Network — pooled, independently-owned hardware coordinated by software.
R+2 receipt	A signed, hash-chained record of an action (Ed25519 + RFC 8785 JCS).
CID	Content identifier — sha256 over the unsigned canonical receipt; also the chain pointer.
JCS	JSON Canonicalization Scheme (RFC 8785) — deterministic JSON serialization for signing.
Reverse auction	Provider-side price competition; the scheduler selects the cheapest capable node.
Credit	The network's unit of account, used uniformly across all products.
Sovereignty constraint	A hard rule pinning a workload to permitted regions/jurisdictions.
SBT	Soulbound token — a non-transferable on-chain identity anchor (agent identity).

SECTION 18

References & disclaimer

- TRD Network products & dashboard — trdn.io
- R-Series receipt standard & conformance suite (open standard, operational prototype).
- RFC 8785 — JSON Canonicalization Scheme (JCS). RFC 8032 — Edwards-Curve Digital Signature Algorithm (EdDSA).

Disclaimer

This whitepaper is for information only and describes the TRD Network architecture and product direction. All prices, rates, tiers and network figures are **illustrative**; live pricing and availability are shown in the dashboard. Forward-looking items (TRD Mart general availability, external audit, zero-knowledge proving) are described as future work and may change. Nothing here is investment, legal or financial advice. © 2026 TRD Network. All rights reserved.