Trantor®

PANORAMICA TECNICA DELLA PIATTAFORMA · LINEA V6

TVirt©

La piattaforma completa, **funzione per funzione.**

Architettura cluster-first, sicurezza, macchine virtuali, container, Ceph gestito, rete e servizi di rete, compatibilità VMware, osservabilità, API e resilienza: il riferimento tecnico per chi valuta la piattaforma, con il dettaglio che una due diligence richiede.

EL 10 · KVM / QEMU | PODMAN · LXC | API VSPHERE-COMPATIBILE | OPENAPI 3.1 | SELINUX ENFORCING | FIPS-READY

ARCHITETTURA

Cluster-first, senza engine

01

Ogni nodo è un peer autosufficiente: nessun management server, nessun ruolo speciale. Un nodo singolo è già un cluster di uno, con lo stesso store reale.

- **Store distribuito (YugabyteDB)** per tutta la configurazione; ogni nodo mantiene una replica PostgreSQL locale in sola lettura, applicata come snapshot atomici con generation watermark.
- **Corsia d'emergenza:** sul nodo proprietario start / stop / kill / reset / pause / resume nascono come task locali e vengono caricati a posteriori.
- **Proximity, la catena di decisione:** gate rigidi (membership, manutenzione, attestazioni, freschezza heartbeat, capacità RAM empirica, reti, regole hard) → preferenze soft → strategia configurabile (leastUsedResources, leastVms, random), deterministica a parità di dati.
- **Manutenzione con drain:** migra le VM dove esiste un candidato, altrimenti stop graceful; un gate selettivo lascia claimabili solo teardown ed emergenze.
- **Controllo diretto dei runtime:** QEMU via QMP e Podman via REST, senza libvirt: meno strati, comportamento verificabile.
- **Siti air-gapped:** installazione e aggiornamento con bundle RPM trasferiti via SSH.
- **Albero di scope unico:** gli scope gerarchici sono la primitiva di tenancy — ancoraggio RBAC, datacenter virtuali, contenimento ancestor-only imposto all'admission.
- **Regola store-down:** le VM continuano, i login verificano sulla replica (staleness esplicita), le scritture di configurazione rispondono 503 — mai comportamenti ambigui.
- **Autorità VM a tre passi:** assegnazione durevole più claim volatile; i trasferimenti degradano a zero esecutori, mai a due; rejoin barrier al riavvio.
- **Join con CA pinning:** token monouso a vita breve con fingerprint della CA; il database del nuovo nodo è seminato da uno snapshot completo prima del primo avvio.
- **Proxy one-hop ?node=** per console, upload e stato del nodo, instradato sul registro replicato — funziona anche a store giù; fra i membri, JWT di identità nodo.
- **Host EL10 standard, x86-64 e ARM (aarch64):** AlmaLinux 10 di default o RHEL 10 con la tua subscription; deployment da ISO + kickstart, dnf + tvirt-setup, o remoto via SSH con piani --dry-run.
- **Rimozione pulita con report:** il nodo torna un sistema EL10 standard e supportabile.
- **Convergenza visibile:** configGeneration, floor di flotta minReplicatedGeneration e watermark per nodo; a store giù le letture cadono sulla replica con stale: true esplicito.

Il motore dei task

- **Un task in esecuzione È il write lease:** le operazioni in conflitto sono rifiutate; un assegnatario irraggiungibile lascia il task in limbo esplicito — mai reclaim a timeout.
- **Watchdog del motore:** budget di runtime per tipo, cancellazione cooperativa poi abbandono con scrittura terminale fenced; ogni dialogo QEMU è time-bounded.
- **Idempotenza:** Idempotency-Key con replay a 24 ore sulle azioni; idempotenza dei task come requisito del motore.
- **Catene di cleanup atomiche:** ereditano il lock del padre e sono auto-rigeneranti: un oggetto non resta mai mezzo pulito e sbloccato.
- **Macrotask controllabili:** stop al passo fallito, resume-from-step, abort con undo dichiarati eseguiti in ordine inverso.
- **Mezze opere flaggate ovunque:** chi materializza una directory scrive un flag .build prima di ogni contenuto: un build incompleto è riconoscibile, mai adottabile.

NIENTE MAGIA, SOLO CONTRATTI

Niente reclaim a timeout, niente cleanup nascosti, niente adozioni automatiche: ogni comportamento del cluster è dichiarato nel contratto API e osservabile nei watermark di convergenza. Quando qualcosa non torna, la piattaforma lo dice ad alta voce — non lo nasconde dietro un retry.

SICUREZZA

Sicurezza by design, verificabile

02

Confinamento, cifratura e controllo degli accessi sono il default della piattaforma, non un profilo da attivare.

Isolamento e cifratura

- **SELinux enforcing** su tutta la piattaforma; ogni VM confinata: utente non privilegiato dedicato, sandbox seccomp, etichetta MCS univoca, cgroup dedicato.
- **Cifratura dischi qcow2-LUKS** con chiavi nel keystore di nodo; dati di piattaforma su LUKS2 con chiave ancorata al TPM; OSD Ceph cifrati a riposo (default sicuro).
- **Confidential computing** AMD SEV-SNP e Intel TDX: memoria delle VM cifrata in esecuzione; appliance hardened con measured attestation.
- **Modalità FIPS** su base RHEL 10; mTLS sullo stream di migrazione; privd come root-helper dedicato e minimale.
- **Firmware a tre livelli** bios / uefi / uefi-secure: Secure Boot applicato e vTPM (swtpm) per VM, immutabile per la vita della macchina.
- **Container rootless di default;** secret cifrati a riposo; credenziali di registry distribuite dal control plane, mai auth.json per nodo.
- **Firewall per VM:** nftables sul tap della vNIC, toggle e default policy per interfaccia, macro di servizio, security group riusabili a livello cluster, alias risolti dall'IPAM; le regole vivono nello store e seguono la live migration.
- **CA di cluster:** join con CA pinning (fingerprint nel token monouso) e chiamate interne fra membri autenticate con asserzioni JWT di identità nodo.

Identità e controllo degli accessi

- **Realm di cluster** con login verificato sulla replica (funziona a store giù); root@local come break-glass a zero dipendenze, verificato via PAM.
- **Ruoli e grant deleganti** su selettori di scope, con quote per scope e grant a gruppi; le catene di delega decadono con l'autorità del delegante.
- **SSO OIDC e MFA TOTP;** igiene di sessione: token opachi revocabili, idle expiry, lockout e policy password replicate.
- **Keystore tipizzato e write-only:** Credential usernamePassword / iscsiChap / cephx, mai restituite; le cephx nate da una proposta Ceph muoiono con il loro storage (cascade tracciato), quelle portate dall'operatore mai.
- **RBAC deny-by-default,** un permesso per ogni azione dichiarato per endpoint; le liste filtrano al visibile: l'invisibile è indistinguibile dall'inesistente.
- **Sudo mode** con ri-autenticazione recente per gli endpoint distruttivi; app token con permessi restringibili, segreto mostrato una volta, revoca puntuale o in blocco.
- **Audit unico VM + container:** append-only, hash concatenati (tamper-evident), diff before→after RFC 6902 con soli riferimenti ai segreti, export syslog / webhook.
- **Secret di piattaforma** a envelope encryption; eventi di sicurezza (login falliti, elevazioni, revoche) nel changelog con severità e visibilità RBAC.

PENSATA PER GLI AMBIENTI REGOLAMENTATI

Difesa, PA e ambienti classificati: deployment e aggiornamenti air-gapped, FIPS su RHEL 10, audit tamper-evident esportabile, cifratura dal disco della VM all'OSD Ceph, e una superficie d'attacco ridotta dal controllo diretto dei runtime. La sicurezza è un requisito dell'architettura, non una checklist commerciale.

COMPUTE

Macchine virtuali

03

Ciclo di vita esplicito, hardware determinato dal profilo, ogni azione auditata.

Ciclo di vita e hardware

- **Lifecycle esplicito** start / stop / reset / pause / resume; stop ACPI con fallback forzato a timeout; azioni bulk su selezioni di massa (delete con keepDisks).
- **CPU e memoria:** topologia socket / core / thread, pinning e NUMA; limiti live via cgroup v2, ballooning, memory hotplug; rate limit per disco e per NIC.
- **Config salvata vs applicata:** le modifiche si salvano subito e si applicano al ciclo successivo; PendingRestart elenca i campi divergenti; live apply per media, cavo NIC e grow del disco.
- **TVirt Guest Tools:** ISO driver + agent per Windows come asset di piattaforma, montata di default alla creazione dei profili windows; versioni per nodo riportate a heartbeat, auto-eject prima della migrazione.
- **Restart policy** always / onFailure / never su stop involontari classificati; avvio al boot del nodo con ordine e ritardi; flag protected.
- **Template e QuickDeploy:** blueprint congelati con deploy a linked clone e provisioning a manifest (rete, identità, join AD offline, attivazione KMS, step custom via agent).
- **Profili hardware** windows / linux / legacy: un campo determina l'hardware virtuale, virtio-only sui profili moderni, set legacy in quarantena dedicata.
- **Dispositivi a slot** (dischi 1–4, cdrom 1–3, NIC 1–6) con bootOrder anche PXE; hotplug di dispositivi e CD reale via QMP sui profili moderni; passthrough PCIe VFIO (IOMMU-strict).
- **Nodo di avvio deliberato:** start accetta un nodo esplicito, validato dagli stessi gate (rifiuto col vincolo nominato, mai correzioni silenziose); il censimento di piazzamento per nodo guida il picker.
- **cloud-init** (userData / networkConfig) e **guest agent** standard: quiesce, report IP / OS, shutdown graceful, guest-exec.
- **Console VNC** su WebSocket con ticket monouso, anche cross-node; screenshot PNG on-demand; metadati (varstore UEFI, stato vTPM) su storage designato.
- **Pool di baseline CPU:** domini di migrazione e piazzamento calcolati sul modello comune più basso (bp-all di default).

Snapshot, cloni, migrazione, alta disponibilità

- **Snapshot esterni qcow2** di tutti i dischi insieme, anche della RAM (il revert riprende la VM in esecuzione); albero con parent e marcatore current; quiesce automatico con flag onesto.
- **Live migration pre-copy:** auto-converge, 60 s di setup-stall, deadline 20 min; ogni fallimento fa rollback con la VM ancora sull'origine; switchover con fencing sullo store — mai due esecutori.
- **Attestazione automatica del nodo silente** (la manutenzione non è mai «morte»), solo storage condiviso — le VM su storage locale sono saltate e verbalizzate — e zombie fence al ritorno del nodo: il doppio avvio è impossibile.
- **Hardening del fencing:** watchdog self-fencing, power fencing Redfish / IPMI opzionale e admission control chiudono la finestra del nodo vivo-ma-isolato.
- **Cloni linked e full** (sorgente protetta 409 finché esistono overlay); MAC e identità vTPM sempre nuove; snapshot thin-pool su iSCSI sotto lease.
- **Proximity HA sempre attiva:** flag ha per VM; restart sullo stesso nodo con backoff esponenziale e resa onesta al crash-loop; failover del nodo morto tramite decider eletto a lease.
- **Regole Proximity** di affinità e anti-affinità su insiemi di VM, hard (422 nominato) o soft (preferenza con warning); stato satisfied / violated in tempo reale.
- **Gate di storage condiviso** e drain di manutenzione integrato; le VM in pausa migrano in pausa.

OGNI AZIONE, UN AUDIT

Non esiste un campo «power»: accensioni, spegnimenti, migrazioni, snapshot e aperture di console sono azioni distinte, auditate una a una e disponibili in dry-run. La storia operativa di ogni macchina è ricostruibile, sempre.

COMPUTE

Container, app e orchestrazione

04

Container OCI e system container LXC nella stessa piattaforma delle VM: stesso scheduler, stessa rete, stesso storage, stesso audit — e un'anagrafica immagini che non finge che i tag siano immutabili.

Workload e runtime

- **Podman supervisionato dal worker,** senza systemd né quadlet né secondi supervisori: lo steady state è riconciliato dietro hash gate — riavvia i morti, parcheggia gli estranei, raccoglie gli orfani.
- **App nate ferme, cancellate deliberatamente:** definisci e poi avvii; il delete con dati persistenti rifiuta senza purge esplicito e le directory superstiti sono elencate per percorso.
- **System container LXC:** kind distinto con lifecycle da VM — template, snapshot, cloni — su scheduler, rete e storage condivisi.
- **ImageDrift:** quando un digest pinnato esce dal suo branch (il retag del vendor che un runtime OCI standard non può nemmeno notare), la condizione si accende.
- **Cache di cluster con proprietà visibile** e GC su pressione disco (i digest pinnati sono protetti); delete guard che nominano i consumatori e chiedono purge per gli orfani.
- **Un solo tipo di workload replicato** — niente matrioska Pod / ReplicaSet / Deployment — con probe, rolling update e rollback sempre referenziabile a digest; reschedule dopo il fencing del nodo.
- **Persistenza scelta per volume e per root:** l'operatore decide cosa sopravvive alla ricreazione e dove vive; rete L3 su rete logica o overlay con IP statico dichiarato (o allocato dall'IPAM).
- **Immagini = digest:** il riferimento è name:branch@selector — il tag è un branch (puntatore mobile), il digest è il commit; la spec normalizzata porta sempre branch@digest, il lockfile.
- **Registry come anagrafica:** Docker Hub, GHCR, ECR, Quay e self-hosted, credenziali nel keystore; verifica con probe reale, refresh stile git fetch con reflog per branch ed errori sticky.
- **Build come primitiva:** OCIBuild e BuildProfile con pod rootless non fidati, canali air-gapped, FROM risolto dall'anagrafica, record di build append-only; registry interno di piattaforma con RBAC di progetto, push e promozione.

Compatibilità

- **Socket Docker Engine API:** la docker CLI ufficiale funziona intonsa (DOCKER_HOST / context); Portainer, pipeline CI e Testcontainers; docker logs, exec e attach nel perimetro RBAC del Project.
- **Import di manifest Kubernetes:** validazione strict, defaulting fedele, reject espliciti con alternativa — mai drop silenziosi; conversioni OpenShift (DeploymentConfig, ImageStream).
- **Tag nativi al posto delle label:** vocabolari enum o free-text, cardinalità singola o multipla, permessi distinti per vocabolario e assegnazione, risolti su ciò che il chiamante può vedere.
- **Compose v2** secondo la specification, interpolazione .env compresa; gli stack Swarm entrano dalla stessa porta via chiave deploy.
- **kubectl ufficiale** su un profilo API documentato e versionato; matrice di compatibilità pubblica, testata a corpus in CI; enclave k3s satellite per operator e CRD.
- **Export dei workload** in formato k8s o Compose, con perdita dichiarata nel report; niente demone Docker privilegiato per nodo.

LA DISCIPLINA DELLA COMPATIBILITÀ

Il profilo API nativo è dichiarato per ciò che è: un sottoinsieme documentato dell'API Kubernetes, mai un claim di conformità. Ciò che non è supportato viene rifiutato con un messaggio mirato e la lista dei reject è parte della specifica: la trasparenza sulla conversione è la prova comportamentale del principio anti lock-in.

DATI

Storage e Ceph gestito

05

Storage

- **Driver** dir, NFS, CIFS, iSCSI (LUN → VG → thin LV), Ceph RBD nativo via QEMU e CephFS kernel-mounted; credenziale cephx obbligatoria dal keystore, mai su argv.
- **Mobilità fra storage:** move / clone dei dischi (id e snapshot conservati, offline-only) e move / copy delle immagini; il confine iSCSI converte intere catene qcow2 ↔ thin-LV con replay parent-first, riferimenti preservati.
- **Impegno di provisioning:** provisionedGiB accanto a capacità / usato / libero — il potenziale configurato, che può legittimamente superare la capacità: l'overcommit è il senso del thin provisioning.
- **Capacità osservata con observedAt** (lo stantio degrada a unknown) e riserva di allocazione che rifiuta gli overcommit non dichiarati.
- **Protocolli estesi:** FibreChannel, FCoE, famiglia NVMe (TCP / FC / RDMA / RoCEv2) e object storage S3-compatibile per immagini e backup.
- **Namespace portabile:** un'unica directory trantor-tvirt-v6/ auto-contenuta con oggetti id-keyed e metadata.json — ripunta il percorso e ri-adotta.
- **Migrazione storage live** dei dischi di una VM accesa; replica asincrona ZFS; mutual-CHAP iSCSI dedicato.
- **iSCSI senza sanlock / lvmlockd:** il write lease dei task serializza le mutazioni LVM su tutta la flotta; attivazione LV esclusiva sul nodo di autorità.
- **Dischi** qcow2 / raw, sorgenti blank / clone / image; resize live grow-only; qcow2-LUKS; content typing all'admission; snapshot chirurgici anche a livello di singolo disco.

Ceph gestito, iperconvergente

- **Cluster multipli e indipendenti:** CephCluster è una collezione — ogni cluster con fsid, quorum mon, realm cephx, registro dispositivi e lease propri; un nodo può avere ruoli in più cluster.
- **Registro OSD con esclusività di flotta:** dispositivi identificati by-path + seriale, un disco appartiene a un solo cluster (409 che nomina il proprietario), wipe solo con wipe: true esplicito.
- **Pool e CephFS first-class:** createPool / deletePool con conferma e finestra limitata, MDS gestiti con guardia sull'ultimo, filesystem che nascono e muoiono con i loro pool dedicati.
- **Posizione fisica dei server → gerarchia CRUSH:** rack, fila, sala, datacenter dichiarati una volta sul nodo; la topologia si applica da sola a ogni cluster interessato, idempotente.
- **Esploratore oggetti e snapshot:** pool → immagini RBD → dettaglio (uso reale via rbd du, watcher live, mappa OSD campionata) su un canale di query read-only e lease-free.
- **Consegnato dalla ISO, legato alla piattaforma:** RPM EL10 nativi preinstallati, demoni supervisionati dal worker — niente cephadm, niente container, niente install a runtime.
- **Crittografia a riposo per OSD** (default sicuro: sempre attiva), chiavi nel config-key store dei mon; ogni disco porta la sua verità frozen cifrato / in chiaro.
- **Consumo proposto, mai auto-creato:** dalla riga del pool o del filesystem nasce lo Storage pronto, con identità cephx limitata al pool coniata nel keystore e il legame che protegge destroy e removeFs.
- **Editor CRUSH guidato** («ogni replica su un server / rack diverso») più crushmap raw dietro flag, compilata e testata con crushtool prima di ogni scrittura, con sudo e conferma.
- **Auto-guarigione nativa, distruzione attestata:** backfill e auto-out di Ceph restano attivi (deviazione dichiarata, solo per dati replicati); mark-lost, purge e riuso dei dispositivi restano dietro attestazione dell'operatore.

LO STORAGE SEGUE LE STESSE REGOLE DI TUTTO IL RESTO

Niente adozioni automatiche, niente wipe impliciti, niente stato nascosto: ogni byte si distrugge solo per decisione esplicita e tracciata, e ogni mezza opera è flaggata e riconoscibile. Anche il Ceph gestito obbedisce al contratto della piattaforma.

RETE

Rete, fabric e servizi di rete

06

Host network e overlay

- **Documento host-network per nodo** (interfacce, VLAN, bond, bridge, hostname, static hosts): convergenza via NetworkManager con checkpoint, rollback e probe di raggiungibilità post-apply.
- **Bond active-backup e LACP** (miimon, primary, hash policy), VLAN, bridge con STP; la validazione rifiuta modifiche che isolerebbero il nodo.
- **IP-per-container** sulla stessa overlay delle VM, senza NAT est-ovest; Service con VIP e load balancing L4, DNS di progetto, NetworkPolicy; cavo virtuale per NIC e defaultPolicy per rete.
- **NIC registrate per MAC permanente:** ciò che non è registrato non viene mai toccato; l'inventario osservato (link, MTU, indirizzi) alimenta validazione e GUI.
- **Overlay VXLAN** multi-nodo unicast con FDB statiche; la partecipazione del nodo è il vincolo di placement; trunk vNIC con VLAN guest mappate e admission per overlay.
- **Ruoli di rete del cluster** management e liveMigration per membro; watermark di convergenza rete / overlay per nodo con last-error visibile.

Fabric, IPAM e servizi

- **Fabric come radice del mondo fisico:** lato logico (reti con MTU e subnet obbligatorie, gateway, DNS, IPv4 / IPv6 / dual-stack) e lato fisico (switch con posizione in vocabolario CRUSH, porte, LAGG LACP slow / fast).
- **Motore di derivazione:** dal cablaggio dichiarato, la configurazione VLAN / bond / bridge di ogni nodo si genera da sola; LLDP passivo come allarme di divergenza.
- **Servizi di rete come demoni bare-metal** supervisionati dal worker: RPM dalla ISO, niente container, config resa a ogni pass dal registro IPAM dietro hash gate — restart solo su cambiamento reale.
- **DNS autoritativo (PowerDNS), NAT (nftables), load balancer / reverse proxy (Envoy)** e dual-stack IPv6; WireGuard, NTP, DHCP relay, PXE / netboot e ACME completano la suite.
- **Firewall per VM ancorato all'IPAM:** «allow from Rete1» risolve la subnet della rete logica; alias e ipset seguono il registro, le regole seguono la migrazione.
- **VLAN dichiarate per porta** (untagged → una rete, tagged → altre), mai sulla rete logica; il cablaggio del nodo dichiara quale NIC raggiunge quale porta.
- **IPAM generico che assorbe da ovunque:** Subnet è un oggetto di registro; la discovery calcola un censimento vivo da nodi, fabric, overlay e Ceph e propone i candidati all'adozione — mai doppioni sullo stesso CIDR.
- **DHCP (Kea):** pool derivati da subnet meno gateway meno riserve, option routers dal gateway, binding di interfaccia; una subnet è servita da un'istanza sola in tutta la flotta (409 che nomina il titolare); fase osservata con onestà sullo stale.
- **HA dei servizi by default:** fencing e ri-piazzamento attraverso il decider Proximity generalizzato.

LA RETE È DICHIARATA, LA DERIVA È VISIBILE

Dal cablaggio fisico alla policy per vNIC, ogni livello è configurazione dichiarata che i worker convergono con checkpoint e probe: una modifica che isolerebbe un nodo viene rifiutata prima, non scoperta dopo. E ciò che resta indietro si vede, con il suo perché.

ECOSISTEMA

Compatibilità VMware nativa

07

VMWCompat parla i protocolli di VMware contro l'API nativa: un traduttore, mai un secondo percorso di scrittura. Il modello canonico resta quello di TVirt; gli oggetti VMware sono proiezioni. Verificato nel gate di test con un govc reale.

- **Quattro superfici, un motore:** SOAP /sdk, il binding VI/JSON, la moderna /api e la legacy /rest convergono su un solo layer vim25 — SOAP è un puro codec, mai due implementazioni.
- **Ciclo di vita completo:** PowerOn / PowerOff / Reset / ShutdownGuest / PowerOnMultiVM, più CreateVM_Task, CloneVM_Task, ReconfigVM_Task, RelocateVM_Task e Destroy_Task — govc e Terraform creano, clonano, riconfigurano, migrano e distruggono VM su TVirt.
- **PropertyCollector fedele:** motore TraversalSpec con profondità rispettata, collector per istanza, WaitForUpdatesEx con versioni monotone e diff, ContainerView, paging e missingSet con la semantica VIM verificata empiricamente.
- **App token come password:** govc, Terraform o un tool di backup si autenticano con un token applicativo, senza credenziali reali.
- **Inventario proiettato:** Datacenter sintetico, ClusterComputeResource ← cluster, HostSystem ← nodi, VirtualMachine ← VM, Datastore ← storage, Network ← overlay, Folder tipizzate ← scope; MoID persistenti, mai derivati dai nomi, mai riusati.
- **Dispositivi con onestà:** ciò che TVirt non modella è tollerato quando è cosmetico e rifiutato NotSupported per nome quando è semantico — mai finzioni.
- **Task durevoli e RBAC proiettato:** i task compat sopravvivono al riavvio del processo (GC e tombstone); AuthorizationManager proietta ruoli e permessi in sola lettura da ciò che la sessione può vedere.
- **Statistiche host, tagging /rest su tag nativi, MoID federati e spostamenti fra folder** completano la superficie sui domini corrispondenti della piattaforma.

LA PROIEZIONE, OGGETTO PER OGGETTO

Oggetto vSphere	Sorgente TVirt
Datacenter	sintetico, radice dell'inventario proiettato
ClusterComputeResource	il cluster TVirt
HostSystem	i nodi del cluster
VirtualMachine	le VM, con MoID persistenti
Datastore	gli storage
Network	le reti overlay
Folder (vm / network / datastore)	gli scope, proiettati come folder tipizzate

PERCHÉ CONTA

Gli strumenti, gli script e le integrazioni costruiti in anni di vSphere continuano a funzionare mentre migri: la compatibilità è il ponte, l'API nativa è la destinazione. E poiché il traduttore non è mai un secondo percorso di scrittura, RBAC e audit restano quelli della piattaforma.

OPERAZIONI

Osservabilità, API, automazione

08

Osservabilità

- **Catalogo contention-first:** cpu.ready (l'analogo di %RDY), pressione PSI di CPU / memoria / I/O, balloon, host-swap del VMM, latenza dischi derivata, contatori per NIC.
- **Streaming SSE a 2 s** per la dashboard aperta e snapshot di fleet in una chiamata: ordina l'intera flotta per ready%.
- **Osservazioni oneste:** ogni campo osservato porta il suo timestamp; oltre la cadenza di raccolta degrada a unknown.
- **Prometheus nativo;** dati raw a 10 s (~26 ore) e rollup a 1 minuto avg+max (7 giorni), con risoluzione scelta dal server.
- **AlertRule integrate** (ready%, swap host, squeeze del balloon, latenza disco, pressione I/O), regolabili e filtrabili per scope.
- **Log locali-first** in NDJSON dietro watermark; changelog semantico pollabile con severità e visibilità RBAC; probe system/health e system/ready non autenticate.

API e automazione

- **OpenAPI 3.1 contract-first** servito dal backend con estensioni x-tvirt (permission, async, sudo, filterable); un test CI confronta il contratto servito con il repository.
- **Convenzioni uniformi:** paginazione cursor, grammatica filtri tipata, ?scope= ad albero, ?sort=, errori RFC 9457 con codici stabili; filtrabilità dichiarata campo per campo.
- **Ansible e Terraform:** collection ufficiale con riconciliazione esterna e purge esplicito; provider Terraform / OpenTofu.
- **Deployment remoto** trantor-tvirt-deploy: install / join / update via SSH con gate di coerenza e piani --dry-run.
- **Una API, tre client:** console web e CLI sono client puri; il client TypeScript generato (nome→id, follower dei task) è l'unico codice HTTP ammesso.
- **Dry-run ovunque** (?dryRun=true) con admission completa; «tvirt apply» accetta formato nativo, YAML Kubernetes e Compose con auto-detection.
- **Webhook in uscita,** client mount per consumatori fuori cluster e gestione UPS integrata.
- **Eventi e task pollabili:** ogni operazione asincrona è una risorsa con log NDJSON seguibile in tempo reale, resume-from-step compreso.

Immagini e contenuti

- **Contenuti verificati:** ISO e cloud image entrano solo con sha256 dichiarato; il server calcola l'hash sugli upload — media non verificati non esistono.
- **Kind iso e disk:** media d'installazione per i cdrom e cloud image consumabili come sorgenti disco per le VM cloud-seeded.
- **Upload chunked e riprendibile** con risincronizzazione dell'offset; contenuti immutabili da ready e protetti 409 finché referenziati.
- **Download di ritorno** dei contenuti una volta ready, per verifiche e riuso fuori piattaforma.

TUTTO CIÒ CHE FA LA CONSOLE, LO FA UNO SCRIPT

Non esistono percorsi privilegiati: la console web e la CLI usano la stessa API pubblica documentata, con lo stesso RBAC e lo stesso audit. Ogni funzione descritta in questo documento è quindi automatizzabile e verificabile in dry-run prima di toccare la produzione — la differenza fra una demo e un ambiente operabile. È anche il motivo per cui la compatibilità VMware può essere un puro traduttore: sotto c'è già tutto.

RESILIENZA

Backup, disastri e uscita

09

Backup e ripristino

- **Backup di configurazione portabile:** un singolo artefatto .tvbk (manifest in chiaro + un JSON per famiglia) scaricabile e ricaricabile ovunque — nasce per vivere fuori dal cluster.
- **Restore selettivo** per famiglia e per oggetto, in modalità merge o overwrite, con esplorazione del contenuto; un backup paracadute è scattato prima di toccare qualsiasi riga.
- **Checkpoint automatico pre-modifica:** «ho cambiato Proximity e me ne pento» diventa una proprietà del sistema, non un incidente.
- **Masterpass per i segreti,** mai persistita: senza, il restore procede con placeholder espliciti; i run schedulati e paracadute lo dichiarano dalla nascita.
- **Guardie parlanti:** foreign key camminate contro stato selezione con ogni mancanza elencata; schedulazioni hourly / daily / weekly con retention che non tocca mai i backup manuali o paracadute.
- **Data plane per VM e container:** repository CAS con dedup, zstd e AES-256-GCM client-side, incrementali CBT / export-diff RBD, consistenza crash o applicativa, immutabilità, restore file-level, repliche 3-2-1 e restore-from-zero.

Attestazioni e ripristino d'emergenza

- **attestLost / attestDown:** dichiarazioni sudo dell'operatore per storage e nodi persi — falliscono i task rilasciando i lease, non distruggono mai byte; i delete diventano record-only.
- **Restore d'emergenza con epoch fencing:** la replica di un superstite è promossa a nuovo store e la nuova epoch recinta i nodi evicted o «resuscitati».
- **Scan & adopt:** la realtà su disco classificata contro lo store, conflitti flaggati e build incompleti riconoscibili; l'operatore decide per oggetto — mai adozioni automatiche.
- **Export / import NDJSON** dell'intera configurazione (segreti come riferimenti), reconcile o replace sudo-protetta, import ordinato per dipendenze, --preserve-uids per il DR.

L'USCITA, PER ISCRITTO

Asset	Formato / protocollo	Procedura di uscita
Dischi e macchine virtuali	QCOW2 · dominio QEMU/KVM, virtio, guest agent	Attacco diretto a qualunque KVM; export XML / OVF
Immagini container	Registry OCI, digest pinnati	skopeo sync verso qualunque registry: i digest restano validi
Workload container	Risorse dichiarative native	tvirt export --format k8s | compose, con perdita dichiarata nel report
Volumi	Ceph, driver CSI standard	Stessi formati e procedure per VM e container
Rete	VXLAN / EVPN / BGP	Interoperabile con apparati di terze parti
Stato e configurazione	YAML dichiarativo	Versionabile nel Git del cliente, non solo nello store TVirt

Documentazione completa e PoC su richiesta.

Contratto OpenAPI, matrice di compatibilità e exit path integrale sono disponibili per la valutazione tecnica.

trantor.it · info@trantor.it