

CUSTOMER FEATURE GUIDE

GuestKit

Offline VM intelligence and migration assurance.

Inspect, score, and repair QCOW2, VMDK, and RAW disk images without ever powering the VM on.

by ZyvorAI Labs

What is GuestKit?

GuestKit is a pure-Rust control plane that reads a virtual machine's disk image offline, builds a normalized evidence snapshot, and answers the two questions that decide every migration: will it boot, and what must change before cutover. It ships as a scriptable CLI (guestkit), a carbon-themed TUI (guestctl), Python bindings, and a self-hosted web platform with KubeVirt integration - all sharing one engine, with no libguestfs appliance to install.

70+

CLI subcommands

6

disk formats read

0

libguestfs appliances needed

8

migration targets scored

What's inside this guide

Getting started — access & first workflows	start here
1 · Offline Disk Inspection	6 features
2 · Migration Assurance	6 features
3 · Fix Plans & Repair	6 features
4 · Security & Threat Analysis	7 features
5 · Inventory & Reporting	6 features
6 · Planning & Optimization	6 features
7 · Interactive Workspaces	7 features
8 · Guest Agent & Live Control	6 features
9 · KubeVirt & Platform	8 features
10 · Deployment & Editions	5 features
Support & contact	→

Access & first workflows

How to reach GuestKit and get your first result. Assumes it has been deployed for you.

How to access it

WEB	Self-hosted web console at <code>http://localhost:8088</code> (nginx front-end proxies <code>/api/</code> to the zyvor-api backend). Start it with <code>docker compose -f deploy/docker-compose.ghcr.yml up -d</code> .
CLI	Two binaries share one command surface: <code>guestkit</code> (scriptable CLI) and <code>guestctl</code> . Install both with <code>cargo install guestkit</code> ; run <code>guestkit --help</code> or <code>guestkit commands</code> to list all subcommands. Launch the TUI with <code>guestctl tui vm.qcow2</code> .
API	REST API served by zyvor-api behind the console's <code>/api/</code> path (e.g. <code>http://localhost:8088/api/</code>); it enqueues inspect/boot-inspect jobs onto the Redis-backed worker. Live guests are reachable host-side via <code>guestkit agent-proxy --listen 127.0.0.1:8765</code> (e.g. <code>curl http://127.0.0.1:8765/doctor</code>). Python bindings expose the engine in-process via <code>from guestkit import Guestfs</code> .
LOGIN	Packaged/web installs seed a default administrator: username <code>admin</code> , password <code>Admin@321</code> (also the default API key where applicable). Change the password, API key and <code>JWT_SECRET</code> immediately after first login and enable SSO/SAML from Settings before any network exposure.
NEEDS	A Linux host with <code>qemu-img</code> , <code>losetup</code> and <code>qemu-nbd</code> installed; mount and in-cluster boot-inspect need root or a privileged pod.

Your first workflows



Pre-flight environment check

1. Install the tools: `cargo install guestkit` (installs `guestkit` and `guestctl`).
2. Verify host tooling is present: `guestkit doctor --help` and confirm `qemu-img/losetup/qemu-nbd` are installed.
3. Confirm the disk opens and its format is detected: `guestkit detect vm.qcow2`.



Assurance-first migration (recommended)

1. Score boot readiness with root-cause: `guestkit doctor vm.vmdk --target proxmox --explain`.
2. Generate the hypervisor-aware checklist: `guestkit migrate-plan vm.vmdk --target proxmox -o json > migration-plan.json`.
3. Capture Windows-specific signals if needed: `guestkit inspect vm.vmdk --profile windows-migration -o json`.
4. Fix boot blockers offline and re-score: `guestkit repair vm.vmdk --fix boot --dry-run`, then `guestkit repair vm.vmdk --fix boot`, then `guestkit doctor vm.vmdk --target proxmox`.



Convert and cut over to KVM

1. Transcode the disk if required: `guestkit convert vm.vmdk --output vm.qcow2 --format qcow2 --compress`.
2. Inspect the converted image: `guestkit inspect vm.qcow2 --profile migration -o json > vm-inventory.json`.
3. Export a reviewable fix plan: `guestkit migrate-plan vm.qcow2 --target kvm --export plan.yaml`.
4. Apply with backups and re-verify: `guestkit plan apply plan.yaml` (roll back with `guestkit plan rollback` if needed).



Gate a golden-image pipeline

1. Run doctor in JSON with a threshold: `guestkit doctor img.qcow2 --target proxmox -o json --fail-below 80`.
2. Enforce sign-off rules as code: `guestkit policy check img.qcow2` (DSL over evidence fields or a CIS benchmark).
3. Fail the CI job on any non-zero exit so regressed images never ship.



Interactive investigation in the TUI

1. Open the dashboard: `guestctl tui vm.qcow2`.
2. Jump to Assurance with `a`, then run doctor `d`, cycle target `t`, preview the fix plan `p`, export YAML `e`.
3. Spelunk files with `guestkit explore vm.qcow2`, or compare two VMs via `guestctl tui vm.qcow2 --compare other.qcow2`.

Offline Disk Inspection

Read the full guest OS from a cold disk image - no boot, no agent, no appliance.



One-command inspect

guestkit inspect surfaces OS, distro, version, hostname, architecture, and init system from any supported disk image in a single pass.

Know exactly what a VM is before you touch it.

How · CLI: `guestkit inspect disk.qcow2` (add `-o json` for automation). In the TUI, the Summary view shows the same fields.



Six disk formats, auto-detected

Reads QCOW2, VMDK, VHD, VHDX, VDI, and RAW/IMG images, choosing loop devices or qemu-nbd automatically.

Point it at whatever your hypervisor exported - it just opens.

How · CLI: `guestkit detect disk.img` confirms the format; any command (`inspect` , `doctor` , `explore`) opens QCOW2/VMDK/VHD/VHDX/VDI/RAW directly. Add `--trace` to see the loop vs qemu-nbd path chosen.



Guest OS detection

Fingerprints Linux distributions (Fedora, Ubuntu, Debian, RHEL, CentOS, SUSE and more) plus Windows from on-disk signals.

Accurate identity without a running kernel.

How · CLI: `guestkit inspect disk.qcow2` reports OS, distro, version and init system; the same identity shows in the TUI Summary view and Python via `Guestfs.inspect_os()`.



Deep system enumeration

Extracts packages, kernels, users, SSH config, services, timers, network, DNS, LVM, fstab, runtimes, containers, certificates, and cloud-init state.

A complete inventory of the machine from bytes on disk.

How · CLI: run focused subcommands like `guestkit packages|services|users|network disk.qcow2`, or the full `guestkit inspect disk.qcow2` for everything in one pass.



Windows signal parsing

Reads SAM/SECURITY registry hives to detect BitLocker, domain join, RDP, and driver gaps for Windows guests.

See the Windows-specific blockers Linux tools miss.

How · CLI: `guestkit inspect disk.vmdk --profile windows-migration` parses SAM/SECURITY hives for BitLocker, domain join, RDP and driver gaps.



Pure-Rust engine, no libguestfs

Partition tables, filesystem signatures, and evidence schema are parsed in Rust; only host NBD/loop is used for mount.

No guestfish appliance, no fragile daemon - fewer moving parts.

How · Automatic on every command; add `--trace` (e.g. `guestkit inspect disk.qcow2 --trace`) to see the Rust parsers and the host NBD/loop mount that were used.

Inspection is always read-only: your source image is never modified.

Migration Assurance

Score boot readiness and generate hypervisor-aware fix plans before you cut over.

Target	Boot analysis	Migration rules applied
kvm / proxmox / qemu	Proxmox/KVM	VirtIO, virtio-scsi/net, VMware Tools to qemu-ga
aws / azure / gcp / cloud	Cloud	cloud-init datasource, BYOL licensing
hyperv	Hyper-V	Hyper-V-specific boot checks



Boot assurance score

guestkit doctor predicts first-boot success on a target hypervisor with a 0-100 score, ranked blockers, and warnings.

Answer 'will it boot?' before the weekend, not during it.

How · CLI: `guestkit doctor vm.qcow2 --target proxmox`. In the TUI press `d` on the Assurance tab; `t` cycles the target.



Root-cause --explain

An inference engine traces each blocker back through a causal chain so you see why a VM would fail to boot.

Fix the cause, not the symptom.

How · CLI: `guestkit doctor vm.qcow2 --target proxmox --explain` prints the causal chain behind each blocker.



Migration score + checklist

guestkit migrate-plan applies target-specific rules for VirtIO drivers, cloud-init, VMware Tools removal, BitLocker, and SELinux relabel across eight targets.

A tailored cutover checklist per destination platform.

How · CLI: `guestkit migrate-plan vm.vmdk --target proxmox` (add `-o json` to capture the checklist). Targets include kvm, proxmox, qemu, aws, azure, gcp, cloud, hyperv.



CI boot gate

`--fail-below` sets an exit-code threshold so pipelines block any image that scores under your bar, JSON still emitted.

Golden images that regress never reach production.

How · CLI: `guestkit doctor img.qcow2 --target proxmox -o json --fail-below 80` exits non-zero when the score drops below your bar while still emitting JSON.



Policy-as-code

guestkit policy check evaluates an expression DSL over evidence fields (e.g. `bootability.score >= 80`) or built-in CIS benchmarks.

Codify sign-off criteria your whole team can trust.

How · CLI: `guestkit policy check vm.qcow2` evaluates the evidence DSL (e.g. `bootability.score >= 80`) or a built-in CIS benchmark.



Forensic diff

guestkit forensic-diff compares two snapshots for config drift, suspicious persistence, and ransomware indicators.

Prove what changed between golden and drifted.

How · CLI: `guestkit forensic-diff golden.qcow2 drifted.qcow2` compares two snapshots for drift, persistence and ransomware indicators.

Fix Plans & Repair

Turn findings into reviewable, reversible, executable remediation - not blind edits.



Reviewable fix plans

Findings become a structured plan of operations (file edits, package installs, service ops, SELinux, registry edits) that you preview before anything runs.

See every change before it happens.

How · CLI: `guestkit plan preview` shows every operation before it runs. In the TUI Assurance tab press `p` to preview the generated plan.



Transactional boot repair

`guestkit repair --fix boot` converts doctor blockers into a plan, applies it with backups, then re-scores to show the delta.

Fix boot blockers offline and prove the score improved.

How · CLI: `guestkit repair vm.qcow2 --fix boot --dry-run` to preview, then `guestkit repair vm.qcow2 --fix boot`; re-run `guestkit doctor` to see the score delta.



Export to bash & Ansible

Plans export as executable shell scripts, Ansible playbooks, JSON, or YAML for change control and runbooks.

Hand ops a runbook your CAB can approve.

How · CLI: `guestkit migrate-plan vm.qcow2 --target kvm --export plan.yaml` (or export as bash/Ansible/JSON) for change control. TUI Assurance `e` exports YAML.



Backup & rollback

`guestkit plan apply` creates timestamped backups; `plan rollback` restores prior state, with dependency ordering and dry-run.

Every change has an undo button.

How · CLI: `guestkit plan apply plan.yaml` writes timestamped backups; `guestkit plan rollback` restores prior state (both support `--dry-run`).



Automated hardening

`guestkit harden` generates security-profile fixes for SSH, firewall, SELinux/AppArmor, and account posture.

Ship hardened images without hand-editing configs.

How · CLI: `guestkit harden vm.qcow2` generates SSH, firewall, SELinux/AppArmor and account-posture fixes as a reviewable plan.



Offline agent injection

`repair --inject-agent` writes a guest agent binary into the disk during migration prep, no boot required.

The VM comes up already instrumented.

How · CLI: `guestkit repair vm.qcow2 --fix boot --inject-agent --agent-binary ./target/x86_64-unknown-linux-musl/release/guestkit`, or add `--inject-agent` to `migrate-plan --export`.

Security teams generate plans; ops teams apply them - full separation of duties, in version control.

SECTION 4

Security & Threat Analysis

Audit posture, hunt for compromise, and prove compliance - all from the offline disk.



Security profile audit

guestkit inspect --profile security scores SSH exposure, UID-0 users, firewall, SELinux/AppArmor, and kernel into a risk level.

A ranked risk verdict per VM in seconds.

How · CLI: `guestkit inspect vm.qcow2 --profile security` returns a ranked risk verdict for SSH, UID-0 users, firewall, SELinux/AppArmor and kernel.



Secret & credential scan

guestkit secrets sweeps the disk for exposed credentials and keys.

Catch leaked secrets before an image ships.

How · CLI: `guestkit secrets vm.qcow2` sweeps the offline disk for exposed credentials and keys.



Malware & rootkit detection

guestkit malware scans for rootkits and known-bad artifacts offline, where in-guest malware can't hide from the scanner.

Inspect a suspect image without executing it.

How · CLI: `guestkit malware vm.qcow2` scans for rootkits and known-bad artifacts without executing the image.



CVE & patch analysis

guestkit cve maps installed packages to known vulnerabilities and missing security patches.

See the VM's exposure without a live agent.

How · CLI: `guestkit cve vm.qcow2` maps installed packages to known vulnerabilities and missing patches.



Compliance checking

guestkit compliance and audit evaluate images against security standards with detailed reporting.

Turn every VM into an audit artifact.

How · CLI: `guestkit compliance vm.qcow2` (or `guestkit audit vm.qcow2`) evaluates the image against security standards with a detailed report.



Threat hunting & IOC

guestkit threat-intel, hunt, and anomaly correlate indicators, detect anomalies, and surface suspicious persistence offline.

Forensic triage on a dead disk, safely.

How · CLI: `guestkit threat-intel vm.qcow2`, `guestkit hunt vm.qcow2`, and `guestkit anomaly vm.qcow2` correlate indicators and surface suspicious persistence.



Forensic timeline & reconstruction

guestkit timeline and reconstruct build an incident timeline from multiple on-disk sources and visualize the attack path.

Rebuild what happened without booting the evidence.

How · CLI: `guestkit timeline vm.qcow2` builds an incident timeline and `guestkit reconstruct vm.qcow2` visualizes the attack path.

Inventory & Reporting

Produce SBOMs, license reports, and shareable documents from any image.



SBOM generation

guestkit sbom emits a software bill of materials in SPDX or CycloneDX from the guest package set.

[Supply-chain inventory for every VM you run.](#)

How · CLI: `guestkit sbom vm.qcow2 --format spdx` (or `cyclonedx`) emits a software bill of materials from the guest package set.



License compliance

guestkit licenses inventories package licenses across the disk for compliance review.

[Know your license exposure before an audit asks.](#)

How · CLI: `guestkit licenses vm.qcow2 inventories` package licenses across the disk.



Self-contained HTML reports

`--export html` builds an interactive, collapsible, print-friendly report with all CSS and JS embedded.

[Email a single file to any stakeholder.](#)

How · CLI: add `--export html` to any inspect run, e.g. `guestkit inspect vm.qcow2 --export html` for a single self-contained file.



Git-friendly Markdown

`--export markdown` produces version-controllable inventory documents for VM-configuration history.

[Track infrastructure drift in your docs repo.](#)

How · CLI: `guestkit inspect vm.qcow2 --export markdown` produces a version-controllable inventory document.



Machine-readable output

Most commands accept `-o json` or `-o yaml` for automation, monitoring, and jq/yq pipelines.

[Wire GuestKit straight into your tooling.](#)

How · CLI: append `-o json` or `-o yaml` to most commands (e.g. `guestkit inspect vm.qcow2 -o json | jq`).



Fleet posture analysis

guestkit fleet analyze scans a directory of images, clusters identical OS fingerprints, and flags snowflakes and low-score blockers.

[See fleet-wide drift at a glance.](#)

How · CLI: `guestkit fleet analyze ./images/` clusters OS fingerprints and flags snowflakes and low-score images across a directory.

`--output (json/yaml/text)` and `--export (html/markdown)` are mutually exclusive - run twice for both. PDF is produced via external tools (wkhtmltopdf, headless browser).

Planning & Optimization

Reverse-engineer infrastructure-as-code, model cloud cost, and map dependencies from a disk.



Infrastructure-as-code blueprints

guestkit blueprint generates Terraform, Ansible, Kubernetes, or Docker Compose definitions from what it finds on the image.

Recreate a legacy VM as code you can redeploy.

How · CLI: `guestkit blueprint vm.qcow2 --format terraform` (also `ansible/kubernetes/compose`) regenerates the VM as redeployable code.



Cloud cost analysis

guestkit cost profiles the workload and estimates run cost plus savings opportunities across AWS, Azure, and GCP.

Price the migration before you commit to a cloud.

How · CLI: `guestkit cost vm.qcow2` estimates run cost and savings across AWS, Azure and GCP.



Dependency graph

guestkit dependencies builds a package dependency graph with conflict, circular-dependency, and impact analysis.

Understand blast radius before you change anything.

How · CLI: `guestkit dependencies vm.qcow2` builds the package dependency graph with conflict, circular and impact analysis.



Disk format conversion

guestkit convert transcodes images between the six supported formats using qemu-img.

Reformat once, migrate anywhere.

How · CLI: `guestkit convert vm.vmdk --output vm.qcow2 --format qcow2 --compress` transcodes between the six supported formats via qemu-img.



Smart recommendations

guestkit recommend and predict surface tuning and remediation guidance grounded in the evidence snapshot.

Actionable next steps, not just raw data.

How · CLI: `guestkit recommend vm.qcow2` (and `guestkit predict vm.qcow2`) surface tuning and remediation guidance from the evidence snapshot.



Performance profile

--profile performance flags swappiness, I/O scheduler, mount options, and network tuning opportunities.

Baseline and tune before cutover.

How · CLI: `guestkit inspect vm.qcow2 --profile performance` flags swappiness, I/O scheduler, mount options and network tuning.

Interactive Workspaces

A carbon TUI, a file explorer, and a shell for hands-on offline investigation.



Carbon-themed TUI

guestctl tui opens a k9s-style dashboard with grouped views, vim keys, a command palette, and glass/transparency themes.

Explore a VM visually without leaving the terminal.

How · TUI: `guestctl tui vm.qcow2` opens the k9s-style dashboard; navigate with vim keys and the command palette.



Assurance view parity

The TUI Assurance tab runs doctor, cycles targets (kvm/proxmox/aws), previews fix plans, and exports YAML - reusing the CLI engine on one mount.

Full assurance workflow, keyboard-driven.

How · TUI: in `guestctl tui vm.qcow2` press **a** for the Assurance tab, then **d** run doctor, **t** cycle target (kvm/proxmox/aws), **p** preview fix plan, **e** export YAML.



Interactive file explorer

guestkit explore browses partitions and files in place with view, info, filter, sort, and hidden-file toggles.

Grep-free spelunking through a cold disk.

How · CLI/TUI: `guestkit explore vm.qcow2` browses partitions and files with view, info, filter, sort and hidden-file toggles.



Guest shell & REPL

guestkit shell and interactive give ls/cat/grep/find over the mounted image plus a scriptable session.

Familiar Unix muscle memory on any VM.

How · CLI: `guestkit shell vm.qcow2` (or `guestkit interactive vm.qcow2`) gives ls/cat/grep/find over the mounted image plus a scriptable session.



Fleet & compare modes

--fleet browses a directory of images with a sidebar; --compare diffs two VMs side by side in the dashboard.

Spot the odd VM out across a set.

How · TUI: `guestctl tui vm.qcow2 --fleet ./images/` for a fleet sidebar; `guestctl tui vm.qcow2 --compare other.qcow2` diffs two VMs side by side.



Global search & jump

Cross-view search finds packages, boot blockers, and migration items; a grouped jump menu navigates every view.

Find any signal without knowing which tab holds it.

How · TUI: inside `guestctl tui`, use cross-view search to find packages, boot blockers or migration items, and the grouped jump menu to navigate any view.



AI copilot Q&A

guestkit ai answers natural-language questions grounded in the evidence snapshot, with pluggable LLM backends (OpenAI, Anthropic, xAI, or local Ollama).

[Ask a VM what's wrong and get an evidence-backed answer.](#)

How · CLI: `guestkit ai vm.qcow2 "why won't this boot?"` answers grounded in the evidence snapshot (build with `--features ai` ; backends: OpenAI, Anthropic, xAI, or local Ollama).

Guest Agent & Live Control

Run inside the guest - or reach it host-mediated - even when there's no guest network.



In-guest agent

guestkit agent runs like qemu-guest-agent over virtio-serial, reusing the same evidence and fix-plan schema as offline mode.

One model for cold-disk and live guests.

How · Run `guestkit agent` inside the booted guest; it serves the same evidence and fix-plan schema over the virtio-serial channel `com.zyvor.guestkit.0`.



Transport ladder

The Guest Control Fabric auto-selects the best path per VM - virtio-serial, QGA exec, QGA builtin, push cache, offline disk, or console.

Guest control that never depends on guest networking.

How · Host bridge: `guestkit agent-proxy --socket /var/lib/libvirt/qemu/channel/target/$VM/com.zyvor.guestkit.0 --listen 127.0.0.1:8765` auto-selects the best path per VM.



Snapshot quiesce

Freeze and thaw guest filesystems (fsfreeze) for application-consistent snapshots, plus soft reboot and graceful shutdown.

Clean snapshots without crash-consistency risk.

How · Agent RPC via the proxy (e.g. `curl http://127.0.0.1:8765/freeze /thaw`) issues fsfreeze, soft reboot and graceful shutdown for consistent snapshots.



Live remediation with approval

Restart failed units, collect support bundles, and run fix plans - policy-gated with JIT approval workflows.

Safe, audited guest actions at fleet scale.

How · Agent RPC / worker jobs run fix plans and restart failed units under JIT approval, e.g. `curl -s http://127.0.0.1:8765/doctor | jq .` then submit an approved plan.



mTLS & signed updates

Agents bootstrap client certs, push heartbeats over mTLS, and self-update from Ed25519-signed, SHA256-verified bundles.

A hardened, tamper-evident guest agent.

How · Agents bootstrap client certs at enrollment and self-update from Ed25519-signed, SHA256-verified bundles; configured through the agent enrollment/config, not a per-run flag.



Deep Linux health

Component scores for boot, systemd, network, DNS, storage, and security via systemd D-Bus, journald, /proc, and PSI pressure.

Root-cause the failed unit from journal correlation.

How · Agent RPC: `curl -s http://127.0.0.1:8765/doctor | jq .` returns component scores (boot, systemd, network, DNS, storage, security) from D-Bus, journald, /proc and PSI.

Deep guest intelligence targets Linux; Windows uses virtio-win and a scheduled-task updater today, with a native agent MSI scaffolded.

KubeVirt & Platform

Boot-inspect stopped VMs in-cluster and drive it all from a self-hosted web console.



Offline boot-inspect for stopped VMs

zyvor-api resolves a stopped VM's root PVC and runs guestkit boot-inspect, returning fstab validity, bootloader, and cloud-init state.

Assurance for halted VMs without booting them.

How · Web console: open a stopped VM and run Boot Inspect (zyvor-api resolves the root PVC and runs `guestkit boot-inspect`); available via the `/api/` boot-inspect endpoint.



Zeus VM Tools

A Kubernetes-native guest agent with cloud-init, QGA, ISO, and airgap install paths plus VMToolsPolicy auto-install/upgrade reconciliation.

The VMware Tools equivalent for KubeVirt.

How · Apply a `VMToolsPolicy` resource (or enable it from the web console) to auto-install/upgrade the KubeVirt guest agent via cloud-init, QGA, ISO or airgap path.



Web console

Self-hosted zyvor-ui + zyvor-api + guestkit-worker ship as public GHCR images and a Helm chart, backed by a Redis job queue.

A team-facing UI over the same engine.

How · Browse to `http://localhost:8088` and sign in with `admin / Admin@321` (change immediately). The nginx front-end proxies `/api/` to zyvor-api.



Python bindings

hypersdk-guestkit on PyPI exposes a libguestfs-style Guestfs API (100+ methods) for programmatic inspection.

Automate disk inspection from Python.

How · Python: `from guestkit import Guestfs` then `g = Guestfs(); g.add_drive("vm.qcow2"); g.launch(); g.inspect_os()` — a libguestfs-style API with 100+ methods.



hyper2kvm pipeline

Pairs with hyper2kvm for VMware-to-KVM conversion, sitting in the wider HyperSDK to GuestKit to v9s to PacketWolf flow.

One assurance gate inside a full migration pipeline.

How · Run hyper2kvm for the VMware-to-KVM conversion and call `guestkit doctor / migrate-plan` as the assurance gate in the same pipeline.



Pluggable auth

The web stack supports JWT, local login, and OIDC/SAML hooks with JWKS-verified ID tokens.

Wire the console into your existing identity.

How · Web console: go to `**Settings**` to enable OIDC/SAML (JWKS-verified ID tokens) or keep JWT/local login; rotate the seeded password and `JWT_SECRET` first.



KubeVirt manifest generation

Emits ready-to-apply DataVolume and VirtualMachine YAML with CDI import URLs, storage class, and CPU/memory sized from the migration plan.

From disk image to running KubeVirt VM in one manifest.

How · CLI: `guestkit migrate-plan vm.qcow2 --target kvm` emits ready-to-apply DataVolume and VirtualMachine YAML with CDI import URL, storage class and sized CPU/memory; also exportable from the web console.



Cloud image sources

Inspects images directly from S3, GCS, and Azure Blob URIs, resolving them to a local path on the fly.

Assess VMs where they already live in object storage.

How · CLI: point any command at an object-storage URI, e.g. `guestkit inspect s3://bucket/vm.qcow2` (also gs:// and Azure Blob), and it resolves to a local path on the fly.

In-cluster boot-inspect needs a privileged pod or node disk access plus get/list RBAC on VMs, VMIs, PVCs, and PVs. A running VM returns VM-spec heuristics - offline disk access is for stopped VMs.

Deployment & Editions

Install in one command; run the full open-source stack; scale with Enterprise support.



cargo install

cargo install guestkit installs both the guestkit CLI and guestctl TUI binaries.

[From zero to inspecting in one line.](#)

How · CLI: `cargo install guestkit` installs both the `guestkit` CLI and `guestctl` TUI binaries.



Run from GHCR

Prebuilt public images (zyvor-ui, zyvor-api, guestkit-worker) come up via docker compose with no docker login.

[Stand up the whole console in minutes.](#)

How · Docker: `docker compose -f deploy/docker-compose.ghcr.yml up -d` brings up zyvor-ui/zyvor-api/guestkit-worker at `http://localhost:8088` with no docker login.



Helm & remote deploy

A Helm chart for clusters plus scripted remote deploy for Docker hosts.

[Ship it where your fleet already lives.](#)

How · Cluster: `helm install` the chart (provisions Postgres/Redis/MinIO); for Docker hosts use the scripted remote deploy under `scripts/`.



Full open-source stack

CLI, TUI, Python bindings, assurance APIs, web console, and KubeVirt hooks are all Apache-2.0 in the repo - Enterprise adds support, not features.

[Nothing core is withheld from the open source.](#)

How · Clone the repo (`git clone https://github.com/ssahani/guestkit`) — CLI, TUI, Python bindings, assurance APIs, web console and KubeVirt hooks are all Apache-2.0.



Enterprise programs

SLA, air-gapped deployment packages, guided playbooks, and fleet automation for 100+ VM and regulated migrations.

[Backed help for VMware-exit programs at scale.](#)

How · Contact the account team at `info@zyvor.dev` for SLA, air-gapped packages, guided playbooks and fleet automation.

The default eval compose stack runs without authentication - do not expose it beyond localhost. Use the production template and turn auth on before any network exposure.

Support & next steps

Support & contact

GuestKit is developed by **ZyvorAI Labs**. For onboarding, a proof-of-concept, or a guided demo, contact your account team at **info@zyvor.dev**.

Good to know: GuestKit runs on Linux hosts and needs host tooling (losetup, qemu-nbd, and qemu-img for conversion); mount and in-cluster boot-inspect operations require root or a privileged pod. Offline disk analysis targets stopped VMs - running VMs return VM-spec heuristics or need the live guest agent. Deep guest intelligence is Linux-first; the native Windows agent MSI is scaffolded, and Windows relies on virtio-win today. LLM-assisted features require building with `--features ai`; deterministic intelligence works without it. Reporting notes: `--output` and `--export` are mutually exclusive, HTML reports truncate to 100 packages, and PDF is produced via external tools.

GuestKit

Inspect, score, and repair QCOW2, VMDK, and RAW disk images
without ever powering the VM on.

info@zyvor.dev · zyvor.dev