

WHITEPAPER

Building and operating a private cloud on Apache CloudStack and Ceph

A practitioner's guide to the architecture, change process, upgrades and monitoring behind a two-region KVM cloud, with the lessons production taught us.

PUBLISHED BY
ZSoftly Technologies Inc.

DATE
September 2026

VERSION
1.0

AUDIENCE
Infrastructure, platform
and operations teams

CONTENTS

Executive summary	Page 2
Reference architecture	Page 3
Storage design with Ceph	Page 4
Networking	Page 5
Changing the platform: code, releases and upgrades	Page 6
Running the platform: patching and observability	Page 7
Lessons from production	Page 8
Readiness checklist and how ZSoftly can help	Page 9
Sources	Page 10

CloudStack and Ceph facts in this paper were checked against the Apache CloudStack and Ceph documentation on 29 September 2026. The practices come from how ZSoftly builds and runs the ZSoftly Cloud Platform.

EXECUTIVE SUMMARY

Open-source parts are the easy half. Operating them is the work.

Apache CloudStack and Ceph give you a complete private cloud from open-source parts: self-service instances, isolated networks, block volumes and S3-compatible object storage on your own KVM hosts. Installing them is the short part. Running them safely for years takes a change process, an upgrade routine and monitoring that tells you the truth. This paper describes all three as ZSoftly practises them on the ZSoftly Cloud Platform.

The problem

- Many CloudStack and Ceph estates start from a manual install. Nobody can rebuild them, and every host drifts a little.
- Storage settings that look fine on day one, such as replica counts and placement rules, fail on the first host reboot.
- Upgrades and patch windows get skipped because each one is a custom project with no rollback plan.
- Monitoring is noisy or blind. Both regions page for one fault, or nothing pages when the monitoring host itself fails.

The approach

Treat the cloud as code. Ansible roles build every layer, from host networking and Ceph to the CloudStack control plane and the tenant catalogue. Numbered playbooks keep the build order readable, and each region has its own inventory. Make one change at a time, verify it on the live system, then move on. Write every upgrade and incident down, and turn each incident into a control: a playbook check, an alert or a runbook step.

Everything in code

Ansible for the platform, Terraform for workloads on top.

One change, then verify

One host at a time, with a smoke test after every change.

Replicas by policy

Ceph pool size and placement written down per device class.

Incidents become controls

Eight lessons from production on page 8.

What this paper covers

- A reference architecture for the management plane, database, KVM hosts, storage and networks.
- Ceph pool, placement and monitor design for CloudStack primary storage and RGW.
- The repository layout and release gates that keep changes safe.
- CloudStack upgrades, Ceph upgrades and host patching, step by step.
- Per-region monitoring that pages once, on symptoms, to the right team.
- A readiness checklist you can use on your own estate.

Who should read this

Infrastructure and platform engineers will use every chapter. **Operations leads** should read the change and running chapters on pages 6 and 7 and the lessons on page 8. **Engineering leaders** weighing an open-source private cloud should read this summary and the checklist on page 9.

CHAPTER 1

Reference architecture

CloudStack organises a cloud into regions, zones, pods, clusters and hosts. A zone typically corresponds to one data centre, and a cluster is a set of KVM servers. The management server needs a MySQL database, and it can run as several load-balanced nodes [1]. We run one zone per region, Montreal and Ottawa, under one management plane.

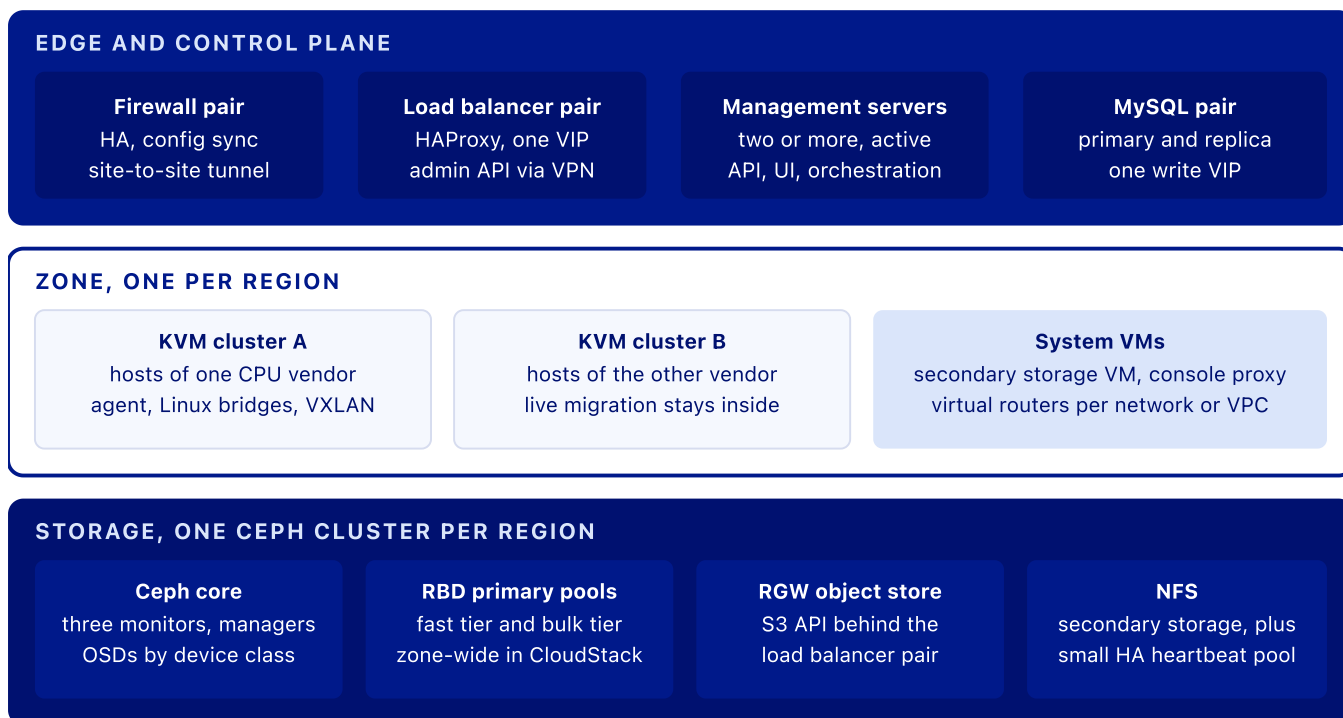


Figure 1. Reference architecture. The control plane serves both zones. Each region repeats the zone, the storage band, the edge pair and its own monitoring stack.

Component	How we run it, and why
Management servers	Two or more active nodes behind the load balancer. Each KVM agent gets an ordered list of servers so it fails over to a known peer, not to a random one.
Database	MySQL primary and replica with GTID replication. CloudStack connects only through a floating write address. A promoted node applies every received transaction before it accepts writes, and a demoted node may rejoin only when its GTID set is a subset of its peer's.
KVM hosts	Clusters split by CPU vendor, so a live migration never crosses vendors. Host tags on offerings place instances by region and CPU family.
System VMs	The secondary storage VM handles templates and snapshot copies. The console proxy has one public IP. Virtual routers serve each network or VPC [3].
Object storage	Ceph RGW registered with CloudStack as an object store, published through the load balancer pair on its own hostname per region.
Monitoring	A separate stack in each region, so the region that has the fault is never the only one watching it. Page 7 covers the design.
Infrastructure guests	Firewalls, load balancers, NFS and databases run on a separate virtualisation cluster. The ones that must survive a Ceph outage use local disks.

Table 1. Components and the design choice behind each one.

CHAPTER 2

Storage design with Ceph

On KVM, CloudStack supports NFS, local disks and Ceph RBD as primary storage, and secondary storage is always accessed over NFS [2]. On KVM, primary storage can be defined for the whole zone [1]. We use Ceph RBD for instance disks, NFS for templates, ISOs and snapshots, and Ceph RGW for object storage.

Placement: one rule per device class

OSDs set their device class to hdd, ssd or nvme when they start, and a CRUSH rule can target one class with a chosen failure domain [4]. We create one replicated rule per class with host as the failure domain, and one CloudStack primary pool per rule: a fast tier and a bulk tier. Check the class Ceph actually reports. An NVMe drive behind an LVM volume can report as ssd, and a rule that asks for nvme then matches nothing.

Replica policy, written down

Setting	When	What it means
size 3, min_size 2	Preferred	Ceph keeps three copies by default [5]. One host can fail or reboot and I/O continues.
size 2, min_size 2	A class spans only two hosts	Safe copies, but a pool pauses I/O while either host is down. Plan a third host of that class.
size 2, min_size 1	Never as a standing setting	Ceph warns that this risks data loss in production. Use it only in an emergency, and only for a short time [5].

Table 2. Pool size policy. The rule must also match the real host count, or placement groups stay undersized.

Monitors are recorded in more places than you think

Ceph recommends at least three monitors for a production cluster, and quorum needs a majority [6]. We run three per region. The monitor addresses end up in the CloudStack primary storage record, in the libvirt pool and secret on every host, and in the definition of every running instance. Changing monitors is therefore a procedure. Add the new monitors next to the old ones, wait for quorum, then update every layer one host at a time.

Deploying Ceph with cephadm

cephadm manages the full lifecycle of a Ceph cluster, from a one-node bootstrap to adding hosts and daemons through the orchestrator [7]. One phased, tagged playbook drives it: prerequisites, bootstrap, hosts, OSDs, CRUSH rules, pools, RGW and dashboard. Rules and pools are declared per region in the inventory, so a change is an inventory edit and a re-run. Two host details matter. The container security profile on monitor hosts must allow the journald socket, or monitor creation hangs. An empty crash list on a mature cluster usually means crash reporting is broken, not that nothing crashed.

Object storage with RGW

The Ceph Object Gateway offers a large subset of the Amazon S3 and OpenStack Swift APIs [8]. Each region has its own realm, zonegroup and zone, index pools on the fast tier, and at least two gateway daemons behind the load balancer pair. The zonegroup API name must match the location constraint that S3 clients and CloudStack send, or bucket creation fails.

CHAPTER 3

Networking

We use advanced zones. Tenants get isolated networks and VPCs. A VPC is a container for isolated network tiers that talk through its virtual router. Each tier has its own CIDR, and network ACLs are numbered rules evaluated from the lowest number up [9]. We disable the network and VPC offerings we do not support, so tenants cannot pick them.

Guest isolation with VXLAN

CloudStack's VXLAN support on KVM uses multicast by default, and each VNI must be unique within a zone [10]. We chose VXLAN over VLANs because the VLAN ID space would cap the number of tenant networks far below what VXLAN allows, and moving later would mean rebuilding every tenant network. One setting matters more than any other. Linux lets a host join 20 multicast groups by default, and CloudStack's documentation says to raise `net.ipv4.igmp_max_memberships` for more networks [10]. Page 8 shows what happens if you do not.

Host networking

- Native Linux bridges: a management bridge, a storage interface at jumbo MTU, a VXLAN underlay and a guest bridge.
- Host network classes, from one NIC to two LACP bonds, handled by one template. Nobody edits host network files by hand.
- Strong-host ARP settings, so a host answers only on the interface that owns an address.
- VLAN membership declared on the bridge itself, never in a hook on a member port that may run before the port joins.

CPU model and live migration

With `host-passthrough`, an instance can only migrate to a host with an exactly matching CPU. `host-model` keeps migration working across slightly different CPUs [11]. We use host-model on mixed clusters and keep kernel versions aligned inside a cluster, because a kernel difference can expose different CPU features.

Edge, load balancing and access

Layer	Pattern
Firewalls	An HA pair per region with configuration sync. Planned failover uses the firewall's maintenance mode. Rules that must exist on both nodes are created on the primary, because a sync overwrites the backup.
Between regions	A site-to-site tunnel between the firewall pairs. It replaced overlay routers that caused asymmetric routes and stale NAT state.
Load balancers	An HAProxy pair per region with one floating address, separate from the firewall. Every backend accepts internal sources only, unless it is deliberately public.
Public addresses	Routable IPv4 for tenant public IPs. Private, admin and system networks sit behind a second NAT stage.
Engineer access	A mesh VPN for people only. Site traffic never depends on it, and the admin API is not reachable from the internet.

Table 3. Edge and access patterns, repeated in each region.

CHAPTER 4

Changing the platform: code, releases and upgrades

Two layers of code describe the cloud. Ansible builds the platform itself: hosts, Ceph, firewalls, switches and CloudStack. Terraform creates workloads on top through the CloudStack API. Playbooks are named `NNN-verb-target.yml`, and the number follows the build order. Gaps leave room for new steps.

Range	Layer	Examples
0xx	Bootstrap	Base packages, hardening, access keys
1xx	Host plumbing	Bridges, bonds, storage interfaces
2xx	Ceph	Deploy, OSD changes, RGW, storage exporters
3xx	Control plane	Database, management servers, zone, system templates
4xx	Hypervisors	KVM agent, add or remove a host, node maintenance
5xx	Tenant catalogue	Offerings, limits, guest templates, VPC offerings
6xx and 9xx	Services and checks	Exporters, smoke tests, diagnostics, the full build

Table 4. Playbook number ranges in build order.

Rules that keep changes safe

- **One inventory per region.** A value that differs by region never goes in a group that exists in both regions. When inventories merge, the last one loaded wins without a warning. CI loads one region per job.
- **Secrets in a vault.** Every secret variable carries a vault prefix and never appears on a command line.
- **Review, then a manual deploy.** A pull request runs a syntax check and a dry run with a diff. Deploys need a named reviewer and never run on merge. A green dry run proves the code parses, not that the live system behaves.
- **One change, then verify.** Maintenance runs one host at a time and waits for Ceph to settle. After each change, confirm the value each host actually resolved and run the smoke tests. In a sequence of risky changes, stop after each one.
- **Diff before you prune.** Compare declared state with the live system before any cleanup deletes objects. A pruner needs a clear rule for what it owns.

Upgrading CloudStack

- 1 Check that all hosts are Up and no virtual router is waiting for an upgrade. Register the new system VM template if the release needs one [12].
- 2 Back up the `cloud` and `cloud_usage` databases on the current primary [12]. Pre-download the packages, so the outage is only the schema migration.
- 3 Stop every management server [12]. Upgrade one, let it migrate the schema, then upgrade the rest.
- 4 Upgrade the KVM agent host by host [12]. Re-apply any local script patches the package overwrites.
- 5 Patch or restart system VMs and virtual routers [12]. Restart non-redundant routers out of hours.

For Ceph, `cephadm` runs the upgrade with `ceph orch upgrade start`. Check `ceph -s` first. It upgrades managers and monitors before other daemons, and restarts each daemon only when the cluster will stay available [13]. Record every upgrade as one row in an append-only changelog.

CHAPTER 5

Running the platform: patching and observability

Patching a hypervisor host

Host patching is an Ansible role, not a checklist in someone's head. It runs one host at a time.

- 1 Silence the host's alerts in every alerting system, with an expiry.
- 2 Check Ceph health, then set `noout` so stopped OSDs do not trigger a rebalance [14].
- 3 Move virtual routers and system VMs by hand. Clear libvirt domains that CloudStack has no record of.
- 4 Put the host in maintenance mode, which live-migrates its instances to other hosts [15].
- 5 Upgrade packages and CPU microcode. Reboot when the running kernel is older than the newest installed one, a reboot flag exists, or services still use old libraries.
- 6 Verify, re-enable the host and wait for placement groups to settle. Then unset the flags, remove the silences and run a smoke test.

Two rules came from experience. Ceph flags control data movement, not I/O, so a pool with size 2 and min_size 2 still pauses while its host is down. Patch the database replica first, fail over, then patch the old primary. Windows run monthly, in tiers, with change approval in advance and a vulnerability scan afterwards.

Observability

Each region runs its own Prometheus, Alertmanager, Grafana and Loki, with pinned image versions. The regions share nothing, so a fault in one cannot blind the other.

What we collect

- Node metrics from every host and guest.
- Ceph metrics from the manager's Prometheus module [16], with every manager-capable host listed.
- CloudStack, MySQL, libvirt, HAProxy, DNS and firewall exporters.
- Text-file collectors for backups, NFS and hardware.

How alerts reach people

- Per-domain channels: storage, network, compute, platform, security.
- Critical and high page around the clock. Warnings wait for business hours.
- Unresolved critical alerts escalate to a phone call.
- Development and test targets never notify.

Page once per fault. Each region scrapes only itself and federates the other region's data for dashboards. Alert rules exclude federated series, or both regions page for one fault. The one allowed cross-region alert watches the peer's monitoring stack, because a stack cannot report its own death.

A status page shows customers the health of each service. It records state and leaves paging to Alertmanager. Every check has an owner team, and each probe tests only what its prober can actually reach, so a monitoring fault does not look like a customer outage.

Alert on the right signal. A healthy VXLAN interface reports its operational state as unknown, so we alert on its admin flag instead. When one condition has one owner, send one message.

CHAPTER 6

Lessons from production

Every incident gets a dated postmortem, and each one ends in a control: a playbook check, an alert or a runbook step. These eight shaped the practices in this paper. We describe the pattern, not the estate.

Lesson	What happened, and why	The control now
Multicast group limit	Past about 20 VXLAN networks on one host, new VXLAN devices came up administratively down. Traffic between hosts dropped while traffic on one host worked, so it looked like a broken template. The stock script did not report the failure.	Raise <code>igmp_max_memberships</code> [10]. A patched script brings the device up and fails loudly. An alert on the admin flag. New hosts ping every peer before use.
Monitor addresses drift	Old monitor addresses stayed in host storage definitions and running instance definitions. Live migrations failed, and pools were one monitor failure away from instances that would not boot.	The playbook now enforces the monitor list instead of only adding to it. A monitor change procedure updates every layer, one host at a time.
Placement rule too wide	CRUSH rules assumed more hosts than a device class had. After failed OSDs were removed, placement groups stayed undersized and I/O blocked for hours before anyone noticed.	Rules match the real host count per class. A written size policy. Alerts on Ceph health and slow operations.
Database divergence	Three separate events left the replica and primary with different histories: a failover that never repointed the old primary, a network partition, and an unattended upgrade that restarted the replica as writable.	Promotion waits for received transactions. A GTID rejoin gate blocks a divergent node and alerts. We rebuild divergent nodes and never merge them.
Stale storage mount	After primary NFS moved to a new server, the agent skipped the remount because the path was already mounted. A dead handle wedged storage tasks on several hosts.	A fleet-wide stale mount sweep and a mount check after any storage move. Read the architecture before touching storage.
Pinned CPU model	A custom CPU model taken from one host stopped an instance from starting anywhere. A kernel difference between two hosts also broke migration between them.	host-model on mixed clusters [11]. Test CPU changes on a throwaway instance. Keep kernels aligned per cluster.
Patched, not rebooted	Hosts with a pinned kernel, or already on the newest kernel with stale services, reported success without a reboot. One kept a wedged NIC for a week.	Compare the running kernel with the newest installed one, and check again after the reboot.
Silences never removed	The maintenance automation called the wrong API path to delete its silences and ignored the error. Monitoring stayed quiet for hours after each window.	Post-flight sweeps every alerting system for silences the automation created, and fails if any remain.

Table 5. Eight production lessons, anonymised, with the control that closed each one.

The common thread. Most of these failures were silent. A step logged a warning, returned success or skipped work it thought was done. Controls that check the live result, not the exit code, catch them.

CHAPTER 7

Readiness checklist

Use this list on your own CloudStack and Ceph estate. Each unchecked item is a place where an incident on page 8 can happen to you.

- | | |
|--|--|
| ☐ Can you rebuild every layer from code, per region? | ☐ Are live migrations kept within one CPU vendor and CPU mode? |
| ☐ Does each Ceph pool have a written size and min_size, and enough hosts in its device class? | ☐ Does host patching confirm the reboot actually happened? |
| ☐ Do you know every place your monitor addresses are stored? | ☐ Do maintenance silences expire and get swept afterwards? |
| ☐ Is multicast group capacity sized for the networks each host will carry? | ☐ Does one fault page once, and does someone watch the monitoring? |
| ☐ Can a demoted database node rejoin without a check on its history? | ☐ Is the last CloudStack upgrade a changelog row with a backup behind it? |

How ZSoftly can help

ZSoftly Technologies Inc. is a cloud engineering company. We build and run the ZSoftly Cloud Platform on CloudStack and Ceph, and we design, build and operate cloud platforms for clients in Canada, the United States and the United Kingdom.

Assess

Review your estate against this checklist and your incident history.

Design

Agree the zone, storage, network and change model in writing.

Build

Deliver it as Ansible roles and numbered playbooks in your repository.

Operate

Run upgrades and patch windows with your team.

Hand over

Code, runbooks and changelogs transfer to your team.

Where we usually start

- 1 Discovery call.** Your first hour with our engineers is free. Bring your architecture and your last three incidents.
- 2 Assessment.** A written report against the checklist, with the risks ranked and a plan for each.
- 3 Delivery.** We agree scope, price and acceptance tests before work starts. You own all the code.

What you keep

- Every role, playbook and inventory, in your own repositories.
- Runbooks for upgrades, host patching, monitor changes and database failover.
- An upgrade changelog and dated postmortems your team can extend.

Talk to an engineer. Book a discovery call at zcp.zsoftly.ca/contact or call +1 (647) 771-1065.

REFERENCES

Sources

All sources were checked on 29 September 2026.

1. Apache CloudStack. Concepts and terminology.
<https://docs.cloudstack.apache.org/en/latest/conceptsandterminology/concepts.html>
2. Apache CloudStack. Storage overview.
<https://docs.cloudstack.apache.org/en/latest/adminguide/storage.html>
3. Apache CloudStack. Working with system virtual machines.
<https://docs.cloudstack.apache.org/en/latest/adminguide/systemvm.html>
4. Ceph. CRUSH maps: device classes and placement rules.
<https://docs.ceph.com/en/latest/rados/operations/crush-map/>
5. Ceph. Pools: size and min_size.
<https://docs.ceph.com/en/latest/rados/operations/pools/>
6. Ceph. Monitor configuration reference.
<https://docs.ceph.com/en/latest/rados/configuration/mon-config-ref/>
7. Ceph. Cephadm.
<https://docs.ceph.com/en/latest/cephadm/>
8. Ceph. Ceph Object Gateway.
<https://docs.ceph.com/en/latest/radosgw/>
9. Apache CloudStack. Virtual Private Cloud configuration.
https://docs.cloudstack.apache.org/en/latest/adminguide/networking/virtual_private_cloud_config.html
10. Apache CloudStack. The VXLAN plugin.
<https://docs.cloudstack.apache.org/en/latest/plugins/vxlan.html>
11. Apache CloudStack. Host KVM installation, guest CPU mode.
<https://docs.cloudstack.apache.org/en/latest/installguide/hypervisor/kvm.html>
12. Apache CloudStack. Upgrade instructions from 4.22.x.
<https://docs.cloudstack.apache.org/en/latest/upgrading/upgrade/upgrade-4.22.html>
13. Ceph. Upgrading Ceph with cephadm.
<https://docs.ceph.com/en/latest/cephadm/upgrade/>
14. Ceph. Troubleshooting OSDs: stopping without rebalancing.
<https://docs.ceph.com/en/latest/rados/troubleshooting/troubleshooting-osd/>
15. Apache CloudStack. Working with hosts, maintenance mode.
<https://docs.cloudstack.apache.org/en/latest/adminguide/hosts.html>
16. Ceph. Prometheus manager module.
<https://docs.ceph.com/en/latest/mgr/prometheus/>

Notices

This paper is for general information. It does not create a warranty or a commitment from ZSoftly Technologies Inc. It describes practices, not guarantees, and it makes no availability or performance claim. Check the linked documentation for the CloudStack and Ceph releases you run. Apache CloudStack, CloudStack and Apache are trademarks of The Apache Software Foundation. Ceph and other names are trademarks of their respective owners.

© 2026 ZSoftly Technologies Inc. Version 1.0, September 2026.