MEMBANGUN PRIVATE CLOUD CLUSTER MULTI-NODE MENGGUNAKAN PROXMOX VE UNTUK IMPLEMENTASI HIGH AVAILABILITY DAN LIVE MIGRATION

Diajukan untuk memenuhi tugas mata kuliah Sistem Operasi

Dosen Pengampu :

Ferdi Chahyadi, S.Kom., M.Cs

Disusun oleh :

Mohd Ikhsan Sadilah (2401020133)

Muhammad Khaerul Sukandar (2401020131)

Rahmat Sucipto Halawa (2401020130)

Christian Aprilio Sihite (2401020151)

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS TEKNIK DAN TEKNOLOGI KEMARITIMAN

UNIVERSITAS MARITIM RAJA ALI HAJI

2025

ABSTRAK

Kebutuhan akan infrastruktur TI yang tangguh (*resilient*) dan fleksibel telah mendorong pergeseran dari server fisik tunggal menuju teknologi virtualisasi terklaster. Namun, adopsi *public cloud* sering terhambat oleh isu kedaulatan data dan biaya operasional yang tinggi. Penelitian ini bertujuan untuk merancang dan mengimplementasikan infrastruktur *Private Cloud* menggunakan platform *open-source* Proxmox VE. Penelitian berfokus pada pembangunan *cluster* multi-node untuk menguji efektivitas fitur *High Availability* (HA) dalam memitigasi kegagalan perangkat keras, serta menganalisis kinerja *Live Migration* menggunakan metode *Iterative Pre-Copy*. Metodologi penelitian mencakup perancangan topologi jaringan terpisah, implementasi penyimpanan terdistribusi (Ceph/ZFS), dan pengujian skenario kegagalan (*failover*). Hasil yang diharapkan adalah sebuah arsitektur sistem yang mampu menjamin kontinuitas layanan dengan *downtime* minimal, serta tersedianya data empiris mengenai latensi jaringan dan penggunaan sumber daya CPU/RAM selama proses migrasi berlangsung.

BAB I
PENDAHULUAN

1.1 Latar Belakang

Dalam arsitektur sistem tradisional, satu server fisik menjalankan satu sistem operasi dan serangkaian aplikasi. Model ini memiliki kelemahan fatal berupa *Single Point of Failure* (SPOF); jika perangkat keras mengalami kerusakan, seluruh layanan akan berhenti total hingga perbaikan fisik selesai. Selain itu, inefisiensi penggunaan sumber daya (*resource underutilization*) sering terjadi ketika server berspesifikasi tinggi hanya menangani beban kerja ringan.

Teknologi *Cloud Computing* menawarkan solusi melalui abstraksi sumber daya, namun organisasi sering kali membutuhkan kontrol penuh terhadap infrastruktur mereka karena alasan regulasi dan keamanan. *Private Cloud* menjadi solusi optimal, namun implementasinya menuntut pemahaman mendalam tentang orkestrasi *cluster*.

Proxmox Virtual Environment (Proxmox VE) adalah platform manajemen server berbasis Debian Linux yang mengintegrasikan *hypervisor* KVM dan kontainer LXC. Fitur krusial yang ditawarkan adalah kemampuan menggabungkan beberapa server fisik menjadi satu kesatuan (*clustering*) untuk mencapai *High Availability* (HA). Penelitian ini difokuskan untuk membuktikan secara teknis bagaimana mekanisme sinkronisasi *state* memori pada *Live Migration* dan mekanisme *Fencing* pada HA bekerja dalam menjaga ketersediaan layanan di lingkungan *Private Cloud*.

1.2 Rumusan Masalah

Berdasarkan latar belakang di atas, rumusan masalah dalam penelitian ini adalah:

1. Bagaimana merancang topologi jaringan *cluster* Proxmox multi-node yang memenuhi syarat *Quorum* untuk mencegah kondisi *Split-Brain*?
2. Bagaimana performa dan latensi yang dihasilkan saat melakukan *Live Migration* mesin virtual antar node dalam kondisi beban kerja yang berbeda?
3. Bagaimana mekanisme *High Availability* (HA) merespons kegagalan node secara tiba-tiba, dan berapa lama waktu pemulihan (*Recovery Time Objective*) yang dibutuhkan?
4. Apa perbandingan efektivitas antara penyimpanan lokal dengan replikasi (ZFS Replication) dibandingkan penyimpanan terdistribusi (Ceph) dalam mendukung arsitektur ini?

1.3 Tujuan Penelitian

Tujuan dalam penelitian ini adalah sebagai berikut:

1. Implementasi Infrastruktur: Membangun *cluster* Proxmox VE yang terdiri dari minimal tiga node fisik/virtual yang terintegrasi.
2. Analisis Live Migration: Mengukur dampak *Live Migration* terhadap ketersediaan layanan (ping response, packet loss).
3. Pengujian Failover: Memverifikasi fungsi otomatisasi HA saat terjadi pemutusan daya atau jaringan pada node utama.
4. Dokumentasi Teknis: Menyediakan laporan mendalam mengenai konfigurasi *storage backend* dan manajemen *resource* sistem operasi.

1.4 Manfaat Penelitian

Manfaat dari penelitian ini adalah sebagai berikut:

1. Bagi Akademisi: Memberikan referensi teknis mengenai implementasi sistem terdistribusi dan perilaku *hypervisor* KVM.
2. Bagi Industri/Praktisi: Memberikan gambaran arsitektur *low-cost high-availability* yang dapat diterapkan pada lingkungan produksi skala kecil hingga menengah (SME).

BAB II
TINJAUAN PUSTAKA DAN LANDASAN TEORI

2.1 Tinjauan Pustaka

Perkembangan teknologi virtualisasi telah mendorong banyak penelitian mengenai efisiensi server dan ketersediaan layanan. Berdasarkan penelusuran literatur, terdapat beberapa penelitian terdahulu yang relevan dengan topik ini.

Penelitian pertama dilakukan oleh Sudyana dan Ali (2014) dalam *Jurnal Sains dan Teknologi Informasi (SATIN)*. Penelitian ini berfokus pada efisiensi sumber daya dengan menggabungkan tujuh server logis ke dalam tiga server fisik menggunakan Proxmox VE. Hasil penelitian menunjukkan bahwa fitur *Live Migration* efektif digunakan untuk memindahkan server virtual tanpa *downtime* saat dilakukan pemeliharaan (*maintenance*) perangkat keras secara terencana. Namun, penelitian ini masih terbatas pada migrasi manual dan belum menerapkan mekanisme pemulihan otomatis yang komprehensif.

Penelitian selanjutnya yang lebih mutakhir dilakukan oleh Febriansyah dan Prapanca (2024) dalam *Journal of Informatics and Computer Science (JINACS)*. Penelitian ini mengembangkan konsep sebelumnya dengan mengimplementasikan penyimpanan terdistribusi Ceph untuk mendukung *High Availability* (HA). Hasil pengujian menunjukkan bahwa integrasi Ceph pada Proxmox mampu mencapai tingkat ketersediaan layanan (*uptime*) hingga di atas 97% untuk VM dan 94% untuk kontainer, dengan mekanisme *failover* yang berjalan otomatis tanpa intervensi manusia.

Berlandaskan pada kedua penelitian tersebut, tugas ini akan menggabungkan konsep efisiensi migrasi dari Sudyana (2014) dan arsitektur penyimpanan tangguh dari Febriansyah (2024) untuk membangun simulasi *Private Cloud Cluster* yang tidak hanya efisien, tetapi juga tahan terhadap kegagalan perangkat keras (*Fault Tolerant*).

2.2 Landasan Teori

2.2.1 Konsep Private Cloud dan Virtualisasi

Private Cloud adalah model penerapan komputasi awan di mana infrastruktur didedikasikan khusus untuk satu organisasi. Model ini menawarkan kontrol penuh atas data dan keamanan dibandingkan *Public Cloud*, namun membutuhkan manajemen infrastruktur yang mandiri. Teknologi inti yang memungkinkan hal ini adalah virtualisasi.

Dalam penelitian ini, digunakan KVM (*Kernel-based Virtual Machine*). KVM adalah teknologi virtualisasi tipe-1 (*bare-metal*) yang mengubah kernel Linux menjadi *hypervisor*. Berbeda dengan emulator yang menerjemahkan setiap instruksi perangkat keras melalui perangkat

lunak, KVM memungkinkan sistem operasi tamu (*guest OS*) mengakses fitur perangkat keras CPU (seperti Intel VT-x atau AMD-V) secara langsung. Hal ini menghasilkan performa yang jauh lebih tinggi dan latensi yang lebih rendah.

2.2.2 Arsitektur Cluster Proxmox VE

Proxmox VE menggunakan arsitektur *cluster* berbasis Corosync Cluster Engine untuk komunikasi antar *node*. Corosync bertugas memastikan setiap *node* mengetahui status *node* lainnya (hidup/mati) dengan latensi sangat rendah menggunakan protokol *Totem Single Ring*. Protokol ini menjamin urutan pesan yang konsisten di seluruh anggota *cluster*.

Selain itu, Proxmox menggunakan pmxcfs (*Proxmox Cluster File System*), sebuah sistem berkas berbasis basis data yang mereplikasi file konfigurasi (yang berada di direktori /etc/pve/) ke seluruh *node* secara *real-time*. Hal ini memungkinkan manajemen terpusat, di mana perubahan konfigurasi pada satu *node* akan langsung direplikasi ke seluruh *node* dalam hitungan milidetik.

2.2.3 High Availability (HA) dan Quorum

High Availability bertujuan meminimalisir waktu henti layanan (*downtime*) akibat kegagalan perangkat keras. Mekanisme utama yang menjamin keberhasilan HA dalam Proxmox adalah Quorum dan Fencing.

Quorum adalah sistem pemungutan suara untuk validasi status *cluster*. Rumus minimal suara untuk mencapai kuorum adalah (Total Node / 2) + 1. Pada *cluster* 3-node, minimal dibutuhkan 2 *node* aktif agar sistem tetap berjalan dalam mode Baca/Tulis. Jika kuorum tidak tercapai (misalnya 2 node mati), *cluster* akan masuk ke mode *Read-Only* untuk mencegah kerusakan data.

Fencing (atau sering disebut STONITH – *Shoot The Other Node In The Head*) adalah mekanisme keamanan kritis. Jika sebuah *node* terdeteksi macet atau jaringan terputus (*Split-Brain*), mekanisme ini akan mematikan paksa *node* tersebut (biasanya melalui perintah ke *Watchdog Timer* atau IPMI) sebelum *node* lain mengambil alih VM-nya. Ini vital untuk mencegah dua *node* berbeda menulis ke disk penyimpanan yang sama secara bersamaan yang dapat menyebabkan korupsi data permanen.

2.2.4 Live Migration: Algoritma Pre-Copy

Live Migration adalah proses memindahkan VM yang sedang berjalan antar host fisik tanpa memutus koneksi pengguna. Algoritma yang umum digunakan adalah Iterative Pre-Copy, yang terdiri dari tiga fase utama:

1. Fase Inisialisasi: Sistem mengalokasikan memori di *node* tujuan sesuai ukuran VM.

2. Fase Iteratif: Seluruh halaman memori (*memory pages*) disalin ke tujuan. Selama proses penyalinan ini, VM tetap berjalan dan terus mengubah data di memori. Halaman memori yang berubah selama transfer ditandai sebagai *dirty pages*. Sistem akan menyalin ulang *dirty pages* tersebut berulang kali hingga laju perubahan memori lebih kecil daripada laju transfer data.
3. Fase Stop-and-Copy: Ketika sisa *dirty pages* sudah sangat sedikit, VM dihentikan sesaat (dalam hitungan milidetik), sisa data terakhir dikirimkan, dan status CPU dipindahkan. VM kemudian langsung diaktifkan di *node* tujuan.

2.2.5 Penyimpanan Terdistribusi (Ceph)

Agar fitur HA dan Migrasi dapat berjalan, data VM tidak boleh tersimpan di disk lokal satu server saja. Penelitian ini menggunakan Ceph, yaitu sistem penyimpanan berbasis objek (*Object Storage*).

Ceph memecah data menjadi objek-objek kecil dan menyebarkannya ke berbagai disk di seluruh *node* menggunakan algoritma CRUSH (*Controlled Replication Under Scalable Hashing*). Algoritma ini memungkinkan klien menghitung lokasi data secara mandiri tanpa memerlukan server metadata pusat, sehingga menghilangkan hambatan kinerja (*bottleneck*). Jika satu disk atau satu *node* rusak, data tetap aman karena salinannya (replika) tersedia di *node* lain.

2.3 Kerangka Pemikiran

Penelitian ini didasari oleh permasalahan risiko kegagalan server fisik (*Single Point of Failure*) yang dapat menghentikan layanan total. Solusi yang ditawarkan adalah implementasi *Cluster Multi-Node* dengan penyimpanan terdistribusi.

Alur kerjanya dimulai dengan menggabungkan tiga server fisik/virtual yang telah diinstalasi Proxmox VE ke dalam satu *Cluster*. Tahap selanjutnya adalah konfigurasi penyimpanan Ceph untuk memastikan replikasi data berjalan antar *node*. Setelah infrastruktur siap, dilakukan aktivasi fitur HA dan pengujian melalui simulasi migrasi VM (*Live Migration*) serta simulasi pemutusan daya mendadak (*Failover*). Output yang diharapkan adalah terbentuknya sistem *Private Cloud* yang mampu melakukan pemulihan otomatis dengan *downtime* minimal.

BAB III
METODOLOGI PENELITIAN

3.1 Desain Lingkungan Eksperimen

Penelitian ini menggunakan pendekatan eksperimental dengan topologi sebagai berikut:
Spesifikasi Hardware (Per Node):

- CPU: Minimal 4 Core (mendukung VT-x/AMD-V).
- RAM: 16 GB DDR4.
- Network: 2x NIC (Network Interface Card).
  - *NIC 1 (eth0):* Management & Public Network (Akses User).
  - *NIC 2 (eth1):* Cluster Sync & Storage Network (Jalur khusus replikasi data agar tidak mengganggu trafik user).
- Storage: 1x SSD untuk OS (Boot), 1x SSD/HDD besar untuk Data OSD/ZFS.

3.2 Tahapan Pelaksanaan

Penelitian dibagi menjadi 5 fase utama:

1. Fase Instalasi dan Konfigurasi Dasar
   - Instalasi Proxmox VE 8.x pada ketiga node.
   - Konfigurasi *Static IP* dan *Host file*.
   - Pengaturan NTP (*Network Time Protocol*) agar waktu tersinkronisasi presisi (krusial untuk *clustering*).
2. Fase Pembentukan Cluster
   - Inisialisasi Cluster pada Node Master.
   - Menambahkan Node 2 dan Node 3 ke dalam cluster menggunakan *Cluster Join Key*.
   - Verifikasi status Quorum menggunakan perintah pvecm status.
3. Fase Implementasi Storage
   - Skenario A (ZFS Replication): Membuat ZFS Pool lokal di tiap node, lalu mengatur jadwal replikasi via GUI.
   - Skenario B (Ceph): Menginstal paket Ceph, mengkonfigurasi Monitor (MON) dan Manager (MGR) di tiap node, serta menambahkan OSD (*Object Storage Daemon*).
4. Fase Pengujian (Testing)
   - Uji Live Migration: Memindahkan VM Ubuntu yang sedang menjalankan streaming video atau transfer file besar. Mengamati apakah terjadi putus koneksi.
   - Uji HA Failover: Mencabut kabel LAN atau mematikan paksa (*hard shutdown*) node tempat VM berjalan. Menghitung waktu stopwatch hingga VM kembali bisa di-ping.

5. Fase Analisis Data
    - Mengambil data log dari /var/log/syslog dan /var/log/pve/tasks/.
    - Menganalisis grafik penggunaan CPU/RAM selama migrasi.

3.3 Parameter Keberhasilan

Parameter	Indikator Keberhasilan	Target Capaian
Stabilitas Cluster	Status Quorum tetap "OK" selama operasi normal	100% Uptime Cluster
Live Migration	VM berpindah tanpa *restart*	Downtime < 1 detik
HA Failover	VM menyala otomatis di node lain saat node asal mati	Recovery Time < 2 menit
Data Integrity	Tidak ada data hilang setelah *failover*	Zero Data Loss

BAB IV
ANALISIS RISIKO DAN MITIGASI

Dalam pelaksanaan proyek ini, beberapa risiko teknis telah diidentifikasi beserta rencana mitigasinya:

1. Risiko: *Network Partitioning* (Kabel putus sebagian).
    - *Dampak:* Node saling menganggap node lain mati, memicu *Split-Brain*.
    - *Mitigasi:* Menggunakan *redundant link* (LACP Bonding) untuk jalur komunikasi cluster.
2. Risiko: Keterbatasan Sumber Daya Hardware.
    - *Dampak:* VM berjalan lambat atau *hang* saat migrasi karena RAM penuh.
    - *Mitigasi:* Menggunakan fitur *KSM (Kernel Same-page Merging)* untuk deduplikasi RAM dan membatasi jumlah VM yang aktif bersamaan.
3. Risiko: Kegagalan Instalasi Ceph.
    - *Dampak:* Penyimpanan terdistribusi tidak berfungsi.
    - *Mitigasi:* Menyiapkan skenario cadangan menggunakan NFS (*Network File System*) pada NAS terpisah jika Ceph gagal diimplementasikan.

BAB V
JADWAL KEGIATAN

Kegiatan penelitian direncanakan berlangsung selama 5 minggu efektif:

1. **Minggu 11: Perancangan & Studi Literatur**
   - Finalisasi proposal.
   - Mempelajari dokumentasi resmi Proxmox dan Ceph.
   - Persiapan ISO dan alat bantu instalasi.
2. **Minggu 12: Instalasi Infrastruktur**
   - Instalasi OS pada semua node.
   - Konfigurasi jaringan (Switching, VLAN).
   - Pembentukan Cluster Proxmox.
3. **Minggu 13: Konfigurasi Storage & VM**
   - Setup Ceph Cluster / ZFS Replication.
   - Pembuatan VM Template dan Cloning.
   - Instalasi *monitoring tools* (htop, iftop).
4. **Minggu 14: Pengujian & Eksperimen**
   - Stress Test: Migrasi bolak-balik antar node.
   - Destructive Test: Simulasi *power failure*.
   - Pencatatan hasil benchmark.
5. **Minggu 15: Analisis & Pelaporan**
   - Rekapitulasi data log.
   - Penyusunan Laporan Akhir.
   - Pembuatan materi presentasi.

DAFTAR PUSTAKA

Febriansyah, A. A., & Prapanca, A. (2024). Simulasi implementasi high availability server menggunakan Ceph pada Proxmox. *JINACS: Journal of Informatics and Computer Science, 6*(1), 27-34.

Kivity, A., Kamay, Y., Laor, D., Lublin, U., & Liguori, A. (2007). KVM: the Linux kernel-based virtual machine. *Proceedings of the Linux Symposium, 1*, 225-230. Ottawa, Ontario, Canada.

Proxmox Server Solutions GmbH. (2024). *Proxmox VE administration guide: Release 8.1*. Vienna: Proxmox Server Solutions. Diakses dari https://pve.proxmox.com/pve-docs/pve-admin-guide.pdf

Sudyana, D., & Ali, E. (2014). Virtualisasi server dengan Proxmox untuk pengoptimalisasian penggunaan resource server pada UPT Teknologi dan Komunikasi Pendidikan. *Jurnal SATIN - Sains dan Teknologi Informasi, 3*(2), 100-106.

Tanenbaum, A. S., & Van Steen, M. (2017). *Distributed systems: Principles and paradigms* (3rd ed.). CreateSpace Independent Publishing Platform.

The Ceph Foundation. (2024). *Ceph architecture: CRUSH map and OSD management*. Diakses dari https://docs.ceph.com/en/latest/architecture/

Weil, S. A., Brandt, S. A., Miller, E. L., Long, D. D., & Maltzahn, C. (2006). Ceph: A scalable, high-performance distributed file system. *Proceedings of the 7th Symposium on Operating Systems Design and Implementation (OSDI)*, 307-320. Seattle, WA.