Rapport Keepers of Logs

Ett projekt om robust infrastruktur, monitorering och säkerhet

Aron Wallroth
Olle Anund
Elis Ilmarson
Abdulkader Dabah
Stefan Ekström
Johannes Kvist

Proxmox

Vad är Proxmox VE?

Proxmox VE (Proxmox Virtual Environment) är en open source-plattform för serverhantering och virtualisering baserat på Debian, som kan installeras direkt på hårdvaran (“bare-metal”) likt andra operativsystem. Dess primära funktion är virtualisering genom två tekniker; LXC (Linux Containers) samt KVM (Kernel-based Virtual Machines). Denna “två-i-ett”-strategi är en av plattformens största styrkor då den tillåter administratören att köra allt från fullfjädrade operativsystem till extremt lätta, applikationsspecifika containrar på en och samma värd. Hanteringen av plattformen sker genom ett centraliserat, webbaserat gränssnitt (GUI). Detta eliminerar behovet av separata, tunga hanteringsklienter och gör tekniken tillgänglig för en bredare användarbas, allt från “homelabs” till stora företagsdatacenter.

Kärnfunktioner

KVM (full virtualisering): KVM möjliggör full virtualisering genom att nyttja hårdvaruacceleration (Intel VT-x/AMD-V). Detta tillåter gästoperativsystemet att köras med nästintill “native” prestanda och passar när en högre grad av isolering krävs.
LXC (containrar): LXC-containrar är inte kompletta VMs då de delar på värdoperativsystemets Linux-kernel och istället skapar en isolerad “userspace”-miljö. Det leder till vissa positiva egenskaper, huvudsakligen extremt låg overhead; de kan startas upp på sekunder och med minimal förbrukning av CPU och RAM. Containrar är ett mycket passande sätt att isolera enskilda applikationer som webbservrar, databaser och andra Linux-baserade tjänster.
Klustring och High Availability (HA): Flertalet proxmox-servrar (noder) kan kopplas samman i ett kluster. Detta möjliggör flertalet avancerade funktioner, bland annat “live-migration”, som ger möjligheten att flytta virtuella maskiner mellan olika fysiska servrar utan nedtid. Detta kräver dock delad lagring. HA tillåter även Proxmox att automatiskt detektera om en nod går ned och startar automatiskt om den nodens virtuella maskiner på en annan, “frisk” nod. Dessa funktioner kräver flera noder och är inte implementerade i detta projekt.
Flexibel lagring: Proxmox har inbyggt stöd för en mängd olika lagringstyper som LVM (Logical Volume Manager) och ZFS som är både ett avancerat filsystem och volymhanterare med funktioner som bland annat integritetskontroll genom checksums och snapshots.

Varför valde vi Proxmox VE?

1. **Kostnadsfritt och open source**
   Plattformen är helt kostnadsfri och låser heller inga viktiga funktioner bakom licenser.
2. **Flexibilitet**
   Möjligheten att välja mellan hela VMs och “lättviktiga” containrar tillåter administratören att effektivt allokera och använda serverns resurser.
3. **Bred support**
   Stort community, lätt att hitta resurser online

Uppbyggnad av vår miljö

Vår infrastruktur är byggd i Proxmox VE, som beskrivits i föregående avsnitt. Vi har valt att nästan uteslutande använda oss av lätta LXC-containrar istället för fulla virtuella maskiner (VMs) för att maximera effektutvinningen, ett koncept som Aron beskriver senare.

Vår kärnfilosofi var att inte bara bygga en samling isolerade tjänster, utan att skapa en **integrerad plattform**. För att uppnå detta byggde vi en webbapplikation (WordPress) som fick agera "testobjekt" för att utvärdera hur vår infrastruktur hanterar loggning, monitorering, backup och säkerhetshot.

Serverarkitektur och Funktioner

Vår Proxmox-miljö består av följande nyckelkomponenter, var och en isolerad i sin egen LXC-container:

- **Webbapplikation (WordPress):** Detta är vår primära testapplikation. För att öka robustheten och testa en mer realistisk "3 lager" arkitektur delades denna upp i två separata containers:
    - **wp-web (Apache/PHP):** En container som kör webbservern Apache och PHP. Den hanterar all logik och presentation av hemsidan.
    - **wp-db (MariaDB):** En dedikerad databas-container som enbart lagrar WordPress-datan (inlägg, användare, etc.). wp-web containern ansluter till denna över det interna nätverket.
- **Monitorering (Grafana/Zabbix):** Arons ansvarsområde. Denna container kör Zabbix-servern och Grafana, och är samlingspunkt för vår monitoreringsdata.
- **Logghantering (Graylog):** Stefans ansvarsområde och inspirationen till vårt gruppnamn. Graylog-servern tar emot, indexerar och gör alla loggar från våra andra containers sökbara.
- **Backup (Proxmox Backup Server, PBS):** Abbes ansvarsområde. Denna container kör PBS och agerar central lagringsplats för alla systemsäkerhetskopior.
- **Plattform & Säkerhet:**
    - **OPNsense (KVM):** En av få tjänster som körs som en full VM för att agera brandvägg och router för hela vår miljö.
    - **Cloudflared:** En lättvikt-container som kör Cloudflare-tunneln. Den ansluter vår interna webbserver till vår publika domän utan att vi behöver öppna några portar i brandväggen.
    - **Vaultwarden:** En internt hostad lösenordshanterare för att gruppen säkert skulle kunna dela på kritiska lösenord.
    - **Twingate:** Vår lösning för "Zero Trust"-åtkomst till Proxmox-miljön för alla gruppmedlemmar.

Tjänsternas anslutning och dataflöde

En tjänst är bara användbar om den kan interagera med resten av systemet. Vi har implementerat flera nyckelkopplingar för att binda samman vår plattform:

1. **Trafikflöde (Webb):** En besökare på vår domän träffar först Cloudflares nätverk. Trafiken skickas krypterad genom vår Cloudflared-container, routas via OPNsense-brandväggen och landar slutligen hos wp-web containern. Denna container kontaktar sedan wp-db containern för att hämta data innan den skickar tillbaka sidan till besökaren.
2. **Monitorering (Data Scraping & Agents):** För att Zabbix-servern ska kunna övervaka våra tjänster har vi installerat en **Zabbix Agent** på varje kritisk container (wp-web, wp-db, Graylog, etc.). Denna agent samlar aktivt in data och rapporterar den tillbaka till Zabbix-servern.
3. **Logghantering (Agents):** Parallellt har vi konfigurerat systemtjänsten rsyslog på våra applikations- och databasservrar. De är konfigurerade för att i realtid skicka alla sina loggfiler (Apache-loggar, PHP-fel, databasloggar) till vår centrala Graylog container för indexering.
4. **Backup-flöde:** Proxmox-hosten har konfigurerats för att ansluta till vår PBS-container. Detta tillåter hosten att schemalagt “skjuta” krypterade, inkrementella backuper av alla våra LXC containrar till PBS, vilket beskrivs mer ingående i Abbes avsnitt

Denna design, där vi separerade applikationer från databaser och kopplade dem till vårt centrala system för monitorering och loggning, skapade den dynamiska testmiljö som vi behövde för att kunna utvärdera säkerhet, feltolerans och robusthet i praktiken.

Säkerhet, tillgänglighet och kapacitet

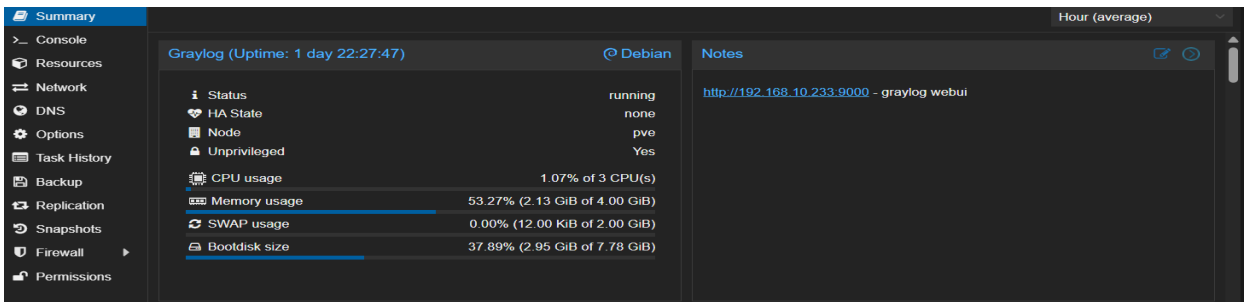
Eftersom vårt projekt inte är särskilt omfattande har vi större möjlighet att anpassa vilken tillgänglighet vi vill uppnå.

Vi har valt att använda **Cloudflare**, som fungerar som första säkerhetsbarriär och skyddar mot bland annat DDoS-attacker och bottrafik. En av dess uppgifter är att dölja serverns IP-adress, vilket ger ett ökat skydd mot externa angrepp. Vi använder även OPNsense, som fungerar som ett filter där trafiken från våra CT/VM hanteras. Här kan man även övervaka och analysera eventuella intrångsförsök mot nätverket. Vidare används Twingate Connectors, som skapar en krypterad tunnel baserad på Zero Trust-principen. Detta innebär att endast verifierade användare får tillgång till interna resurser och i vårt fall har vi tilldelats specifikt tillträde.

För att ytterligare stärka säkerheten i **WordPress** använder vi både Wordfence och WPScan. Wordfence skyddar mot inloggningsattacker, skadlig kod och sårbarheter i teman och tillägg, medan WPScan används för att skanna efter och identifiera kända sårbarheter i **WordPress**, plugins och teman.

Infrastrukturen kan vidareutvecklas för att uppnå högre tillgänglighet genom att införa redundans i form av klustring av flera servrar. Detta skulle möjliggöra mindre belastning på servrarna och förbättra svarstiderna. Det skulle även bidra till att balansera trafiken mellan servrarna, vilket minskar den totala belastningen på systemet och gör det mer stabilt.

När det gäller kapacitet är detta något vi enkelt kan övervaka och justera i Proxmox. Ett tydligt exempel på detta är när jag arbetade med Graylog och märkte att den nästan använde allt tilldelat minne. Vid ett senare tillfälle upptäckte jag även att Graylog behövde mer lagringsutrymme. Detta resulterade i att webbgränssnittet (GUI) inte startade och att vi manuellt fick gå in i containern och frigöra utrymme för att kunna starta den igen. (*se bild för illustration)*.



Dessa resurser kunde vi enkelt justera i Proxmox eftersom plattformen ger en tydlig överblick över hur mycket CPU, RAM och lagring varje container (CT) eller virtuell maskin (VM) använder. På så sätt kan vi snabbt anpassa resurserna efter behov och säkerställa att systemet har rätt kapacitet.

För att upprätthålla en hög nivå av säkerhet har vi valt att använda **OPNsense**, en open source-brandvägg. Med hjälp av **OPNsense** kan vi övervaka aktiviteter i nätverket och konfigurera så att all trafik från våra containrar och virtuella maskiner passerar genom brandväggen. Genom att använda Graylog för logghantering kan vi dessutom samla in, analysera och visualisera loggar från andra system. På så sätt får vi en tydlig överblick över nätverkets aktivitet och kan upptäcka avvikelser eller misstänkt beteenden.

Effektutvinning ur ett IT-system

Effektutvinningen i ett IT-system är kritisk för att maximera hur mycket man kan göra med de resurser som finns tillgängliga. Detta är applicerbart oavsett om man arbetar i en cloud-miljö eller har en self-hostad lösning. I molnet, där man generellt betalar för de resurser man behöver, innebär mer effektiv användning av resurser att man kan hålla kostnaden nere. I en self-hostad miljö innebär mer effektiv användning av resurser självklart att man klarar sig längre utan att behöva investera i mer hårdvara när man skalar upp sin plattform.

I och med att vi utgick från en befintlig hemmaserver som kör Proxmox hade vi tillgång till en fast mängd resurser under projektets gång. Lyckligtvis var dessa resurser mer än tillräckliga för våra behov, men det är ändå intressant att reflektera över hur man kan använda de resurser man har tillgång till så effektivt som möjligt, eftersom det skulle vara högst relevant om vi skulle bygga vidare på vår plattform och fortsätta att skala upp den.

En fördel med Proxmox är att det är lika enkelt att virtualisera LXCs som det är att virtualisera hela VMs. Som tidigare nämnt virtualiseras det ingen kernel i en LXC, och den blir därmed mycket mindre resurskrävande än en VM. En grundläggande princip för effektutvinning i vårt projekt blev därför att använda LXCs i alla fall där en hel VM inte var nödvändig, vilket sparade mycket på resurser. Det finns självklart användningsområden där en dedikerad kernel är fördelaktigt eller till och med nödvändigt att ha, så som till exempel brandväggar, men de allra flesta servrar vi satte upp som en del av vår plattform fungerade utmärkt som LXCs.

Om man monitorerar resursförbrukning med till exempel Zabbix, som vi har konfigurerat för detta projekt, skulle det vara relativt enkelt att undersöka hur mycket av allokerade resurser våra olika servrar använder och anpassa hur mycket resurser de tilldelas baserat på det. Självklart skulle en del av detta vara att ha koll på hur behovet fluktuerar, till exempel beroende på antalet besökare på hemsidan, samt att räkna med viss marginal, men ur ett längre tidsperspektiv skulle det kunna vara en effektiv metod för att spara på resurser.

En till metod som är mycket användbar i vissa specifika fall är att manuellt anpassa resursallokering temporärt. Detta användes till exempel under vårt projekt när en resursintensiv kompilering skulle köras vid konfigurationen av Vaultwarden. En stor del av de resurser som fanns tillgängliga allokerades då till den containern, och kompileringen blev då färdig mycket snabbare än den annars hade blivit. Resurserna kunde sedan återbördas för att än en gång finnas tillgängliga till andra servrar.

Det är värt att notera att Proxmox har viss inbyggd funktionalitet för att dynamiskt allokera tillgängliga resurser. Det är till exempel möjligt att allokera fler CPU-cores än man fysiskt egentligen har, och låta Proxmox avgöra när behovet finns att faktiskt bruka dessa, och i andra fall lämna de tillgängliga för andra VMs och containrar. Om detta projekt hade pågått längre hade det varit en god idé att undersöka exakt hur detta fungerar och utifrån det optimera hur vi allokerar de resurser vi har för att få maximal effektutvinning.

Backup

Backup-servern i projektet är byggd med **Proxmox Backup Server (PBS)** som installerades i en egen container i Proxmox-miljön. Syftet med PBS är att hantera säkerhetskopiering av alla andra containers och servrar i projektet, till exempel Graylog, monitoreringsservern och brandväggservern.

Backup-servern installerades på en Debian 12-container där programmet *proxmox-backup-server* installerades via APT. En mapp skapades som datalagring på sökvägen **/mnt/backups**, där alla backupfiler sparas. PBS fick en fast IP-adress (**192.168.10.234**) och är ansluten till samma nätverksbridge (**vmbr0**) som övriga containrar, vilket gör att all kommunikation sker internt i det lokala nätverket.

För att koppla ihop PBS med Proxmox-hosten verifierades serverns **fingerprint** i Proxmox-gränssnittet. Därefter kunde Proxmox skicka backuper direkt till PBS. I Proxmox valdes vilka containers som skulle säkerhetskopieras, och som mål sattes det nya datastoret i PBS.

En fullständig backup av containern **Graylog (ID 105)** genomfördes, och sedan testades återställning (restore) till en ny container (**Graylog 200**). Testet lyckades och visade att både kommunikationen och backup-funktionen fungerade som planerat.

Eftersom hela miljön ligger på samma nätverk (**vmbr0**) kunde backuptrafiken ske säkert och utan att exponera PBS mot internet. Detta minskade risken för obehörig åtkomst.

Resultatet blev att backup-servern fungerar stabilt och tillförlitligt. Proxmox kan nu skicka backuper automatiskt till PBS, och återställning av en server kan genomföras på cirka fem minuter. Automatiska backuper görs var 14:e dag, och gamla backuper tas även bort vid samma intervall för att spara lagringsutrymme.