Uppgiften i korthet

Detta var en gruppuppgift i kursen “IT-infrastruktur”. Syftet med uppgiften var att skapa någon typ av plattform med hjälp av någon sorts virtualiseringsverktyg, antingen self-hosted, i en cloudmiljö eller i ett kluster. Utöver plattformen i sig skulle vi även implementera ett antal ytterligare funktioner, utifrån ett antal förslag. Arbetet skulle även dokumenteras löpande och en daglig “check-in” skulle göras i en allmän kanal på discord där vi uppdaterade vad vi gjort under dagen och vad vi stod inför därnäst. Grupparbetet avslutades med en presentation för resten av klassen och en slutgiltig rapport som lämnades in. Denna rapport, med mer teoretisk beskrivning av verktygen och koncepten som projektet berörde, kan också hittas i denna mapp.

Projektöversikt: Skalbar webbplattform i virtualiserad miljö

Vi avgjorde under projektets början att skapa en webbserver med apache och php och en tillhörande databas på en separat virtuell maskin. Att separera webbserver och databas har flera fördelar, huvudsakligen skalbarhet, prestanda och säkerhet. Det är lättare att expandera lösningen när behovet tillkommer av ökad användning, allokerade resurser kan anpassas mer specifikt och separerade servrar gör att en attack på en server inte nödvändigtvis skadar hela infrastrukturen.

Till detta valde vi även att konfigurera ett antal tillhörande funktioner, nämligen en loggserver, en monitoreringsserver, en backupserver, en brandvägg och en server för lösenordshantering. Då en i vår grupp redan hade en enkel hemmaserver konfigurerad redan, valde vi att skapa en dedikerad miljö på den och använda för hosting. Servern körde virtualiseringsverktyget “Proxmox”, vilket även passade bra för detta projekt. Vi använde oss av Twingate för att ge alla gruppmedlemmar åtkomst till servern från sin egen hårdvara så att alla kunde arbeta på sin del närsomhelst och vart som helst. Twingate har även fördelen att det är säkrare än en traditionell VPN, i och med deras “Zero Trust Network Access”. Hemsidan som vi hostade på webbservern fanns under projektets gång tillgänglig på domänen “keepersoflogs.site”.

Vår grupp bestod av sex medlemmar. Tre personer fick ansvar för att sätta upp miljön i Proxmox, konfigurera Twingate samt att sätta upp webb- och databasservern. En person fick ansvara för loggservern, en person för backupservern och jag fick ansvar för monitoreringsservern och servern för lösenordshantering. Jag blev även utsedd till gruppledare, vilket innebar att jag hade allmän översikt över arbetets fortgång, såg till att samarbetet mellan de olika funktionerna fungerade, förberedde, sammankallade och ledde gruppmöten samt hade huvudsakligt ansvar för daglig check-in.

Mina bidrag till projektet

Innan projektets början visste jag väldigt lite om monitorering. Därmed började med att researcha vilka olika verktyg som fanns tillgängliga, hur dessa kunde konfigureras och samspela med varandra samt vad en rimlig struktur var för infrastrukturen kring monitoreringen. Jag bestämde mig för att använda Zabbix och Uptime Kuma, och att integrera dessa i Grafana. I och med att Uptime Kuma inte har stöd för att direkt integreras i Grafana använde jag mig också av Prometheus för att skrapa datan som Uptime Kuma samlade in och, anslöt sedan Prometheus som datakälla i Grafana.

Zabbix är ett monitoreringsverktyg vars primära syfte är att övervaka hårdvarustatus, t.ex. RAM-användning, CPU-användning och diskutrymme. Zabbix fungerar med två komponenter, Zabbix server och Zabbix agents. Zabbix server installeras och körs på den server där man samlar sin data och där man har tillgång till användargränssnittet. Agenter installeras på de servrar som man vill övervaka. Dessa samlar in data på servern de är installerade på och skickar den till Zabbix server.

Uptime Kuma är ett verktyg som används för att monitorera tillgänglighet. Jag konfigurerade Uptime Kuma på en egen server där jag även körde Prometheus. Uptime Kuma fungerar genom att skicka olika typer av datapaket med jämna mellanrum och utvärdera svaren. Till exempel lät jag Uptime Kuma skicka pings till alla andra servrar i vår miljö, och baserat på om ett svars ges eller inte kan man monitorera om servern är igång och fungerar som den ska. Jag konfigurerade även Uptime Kuma att skicka HTTP-requests till vår domän för att monitorera om hemsidan var uppe som den skulle. För att snabbt bli informerad om när vår hemsida, som var hela syftet med vår plattform, låg nere konfigurerade jag så att ett varningsmeddelande automatiskt skickades till gruppens discord-kanal när hemsidan blev otillgänglig, som kan ses i en av bilderna nedan.

Grafana används för att samla in data från andra monitoreringsverktyg och visa datan på ett och samma ställe med hjälp av ett enkelt och tydligt användargränssnitt. Grafana använder sig av dashboards och dessa går att anpassa till att visa precis det man vill baserat på datan som Grafana får in. Det finns också ett stort utbud av förkonfigurerade dashboards tillgängligt, vilket gör det mycket enkelt att komma igång med Grafana.

För en enkel lösning för lösenordshantering valde jag att konfigurera VaultWarden på en egen server. Det är simpelt att sätta upp och fungerar som ett valv med alla projektets lösenord sparade på en central punkt, bakom ett enda masterlösenord. I och med att lösningen är helt self-hosted är denna lösning mycket säker så länge masterlösenordet är starkt och inte blir läckt. Även om denna specifika instans konfigurerades för en mindre grupp, gav det en god förståelse för hur centraliserad lösenordshantering kan höja säkerhetsnivån i en organisation.

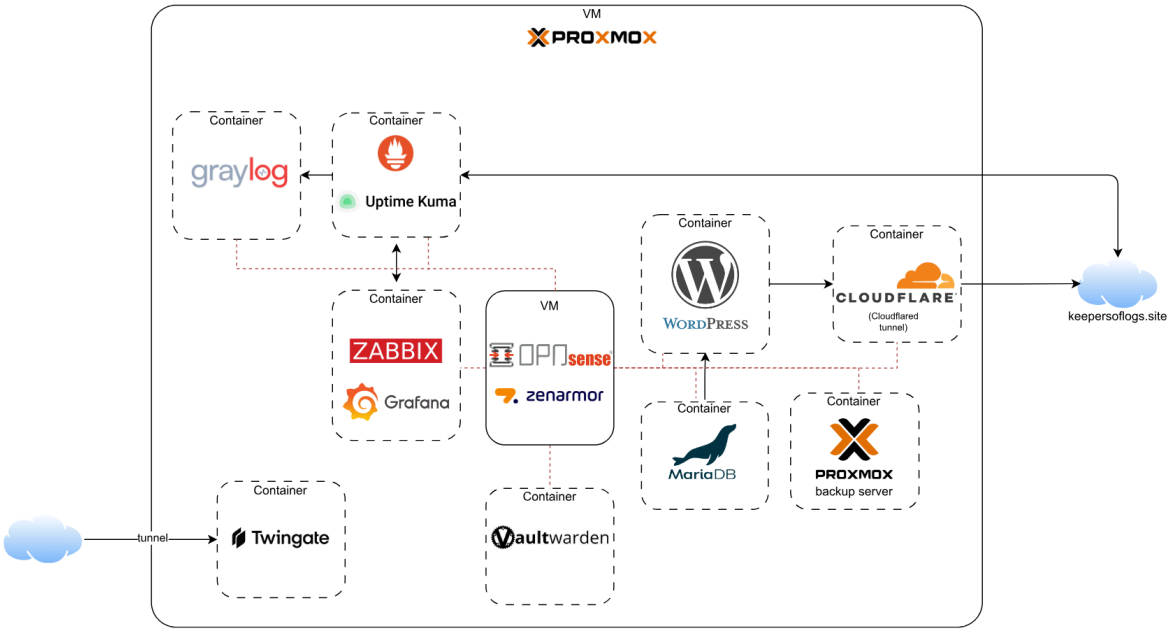
Slutsats och lärdomar

Projektet resulterade i en robust och redundant infrastruktur som framgångsrikt hostade en fungerande webbplattform. Genom att använda Proxmox som hypervisor fick vi en djupare förståelse för hur resursallokering och virtualisering fungerar i praktiken, samt hur kritiskt det är med säker fjärråtkomst i en distribuerad arbetsgrupp. Integrationen av Twingate var avgörande för att vi effektivt skulle kunna samarbeta utan att exponera interna tjänster direkt mot det publika internetet.

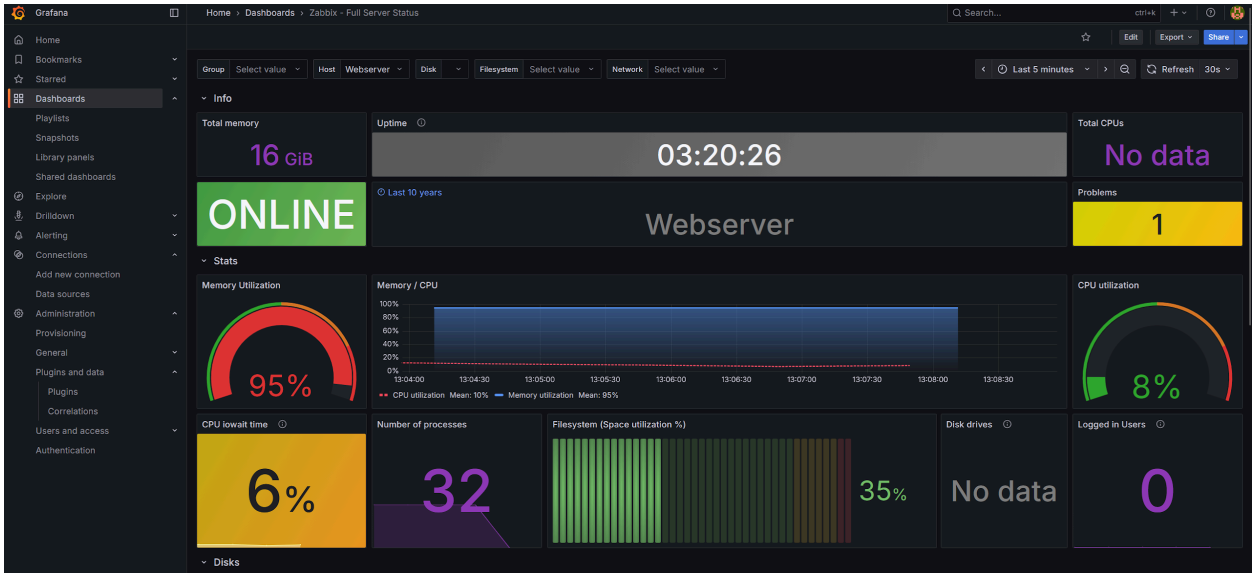
För min egen del gav projektet två stora insikter. Tekniskt sett var implementationen av monitoreringsstacken – Zabbix, Prometheus, Uptime Kuma och Grafana – en mycket utmanande, men väldigt givande erfarenhet. Att konfigurera automatiska larm till Discord gav en praktisk förståelse för hur incidentrespons kan automatiseras i en modern IT-miljö.

Ur ett ledarskapsperspektiv var rollen som gruppledare mycket lärorik. Att koordinera sex personer med olika ansvarsområden krävde tydlig kommunikation och struktur. Genom att upprätthålla dagliga check-ins och dokumentera framsteg löpande kunde vi identifiera beroenden mellan de olika servrarna tidigt (exempelvis hur monitoreringen behövde konfigureras för varje individuell agent). Detta projekt har stärkt min förmåga att driva projekt framåt och tydliggjort vikten av att kombinera teknisk expertis med ledarskap för att nå ett framgångsrikt slutresultat.

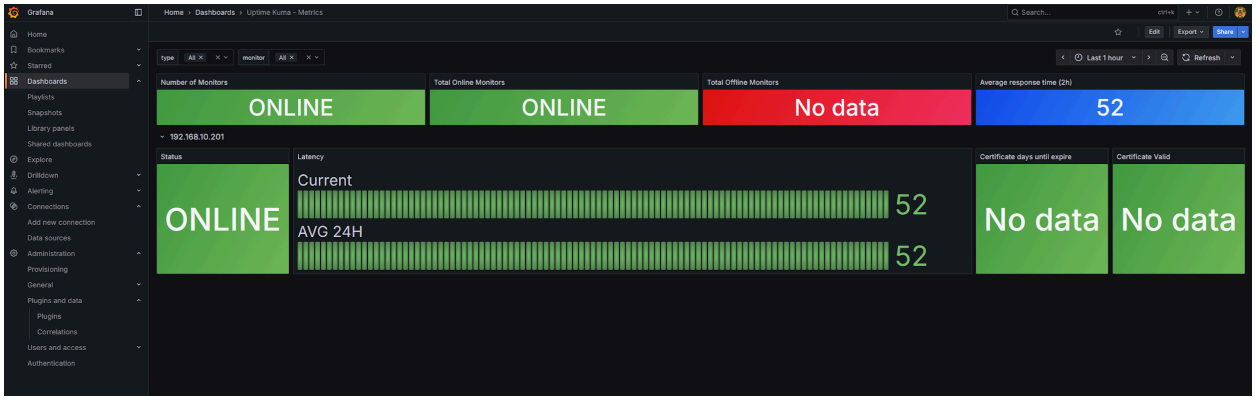
Bilder



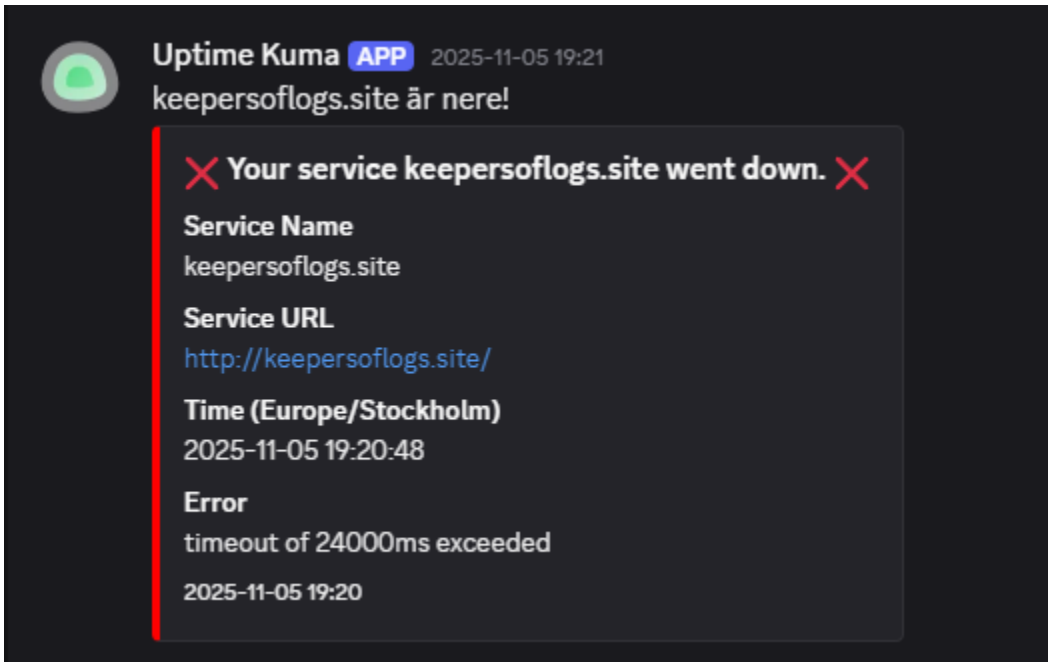
En överblick av infrastrukturen i projektet



Ett exempel på en dashboard i Grafana med data hämtad från en av våra servrar med en Zabbix-agent



Ett exempel på en dashboard i Grafana med data hämtad från Uptime Kuma och skickad via Prometheus



En automatiserad varning i vår discordkanal från när vår hemsida var nere