



id: SI_008

model: google/gemma-4-26b-a4b-it

seeds:

04_risiko: sårbarhet

03_relasjon: samarbeid

05_handling: vent

01_perspektiv: strategisk

02_tidhorisont: kortsiktig

Instans SI_008

Seeds: strategisk | kortsiktig | samarbeid | sårbarhet | vent

Analyse: Operasjonalisering av Swarm i Helse-kontekst

1. Arkitektur: Azure OpenAI + Python (LangGraph/Autogen) er din eneste vei.

Strategisk: Copilot Studio er for rigid for 100 agenter; det vil kveles av latency og manglende kontroll over state-management. For å bevare Microsoft-investering må du bruke **Azure OpenAI Service** integrert med **Azure Functions** eller **AKS**. Dette gir deg kontroll over Entra ID-autentisering og sikrer at data aldri forlater din tenant. Sårbarhet: Ved å unngå SaaS-sverm-rammeverk (CrewAI) minimerer du “black box”-risikoen i et strengt regulert miljø.

2. Case 1: Perspektivanalyse (Syntese vs. Kaos) Ikke dupliser prompts. Bruk en Hierarkisk Orkerstrering.

- Nivå 1: 100 spesialiserte agenter (via dynamisk prompt-injeksjon basert på metadata).
- Nivå 2 (Syntetiker): En “Moderator-agent” som mottar alle 100 outputs og utfører en Reductio ad Unum – identifiserer konsensus, avvik og kritiske risikofaktorer. Implementering: Bruk en `System-Persona-Map` i en SQL-tabell som injiserer spesifikke constraints (f.eks. “Helse-regulatoriske krav”) per agent.

3. Case 2: Inbox-modell (Distribuert Processing) Bruk en Queue-based pattern

(Azure Service Bus). Arkitektur: Case lander i kø → En “Dispatcher-agent” analyserer kompleksitet → Allokere til en spesialisert agent-pool. Consistency: Bruk en “Golden Template” for output-struktur (JSON schema), men la “Reasoning-trinnene” være agenter-spesifikke.



4. Sikkerhet og Data (Zero Trust) Sårbarhet: Unngå RAG som kopierer data til vektordatabaser utenfor kontroll. Bruk **Azure AI Search** med Security Filters basert på Entra ID-brukerrettigheter. Agenten må “arve” brukerens tilgangsnivå.

5. Observerbarhet og ROI Kortsiktig: Implementer **Azure Monitor + Application Insights** for å spore token-bruk per agent-ID. Cost-optimering: Bruk GPT-4o for “Dispatcher” og “Syntetiker”, men bruk GPT-4o-mini for de 100 “arbeider-agentene”. Dette gir massiv ROI.

6. Transition

- Fase 1 (0-3 mnd): Pilot Case 2 (distribuert batch) for å teste stabilitet.
- Fase 2 (3-9 mnd): Skalering til Case 1 (kompleks analyse).

Neste steg:

1. Sett opp en Azure OpenAI-instans med strict data residency.
2. Utvikle en “Dispatcher” prototype i Python som kan lese fra en Azure Service Bus.
3. Definer JSON-skjemaet for alle agent-outputs for å sikre maskinell aggregering.