



id: SI_001

model: openai/gpt-5.4-mini

seeds:

04_risiko: sårbarhet

03_relasjon: makt

05_handling: vent

01_perspektiv: strategisk

02_tidhorisont: kortsiktig

Instans SI_001

Seeds: strategisk | kortsiktig | makt | sårbarhet | vent

For en **Microsoft-sentrisk, helse-streng, cloud-only** organisasjon ville jeg **ikke** startet med CrewAI/Anthropic som primærvei. Det gir rask prototype, men svekker **makt over data, audit og residency**. For din kontekst er beste valg:

Azure OpenAI + Python-orchestrering på Azure som hovedspor, med **Copilot Studio/Power Automate kun for lette, menneske-inn-the-loop workflows**. Dette bevarer Microsoft-investment og gir sverm-skala.

1) Arkitekturvalg

- **Azure OpenAI + Python:** best for 100-agents parallellisme, custom routing, observability, retry/backoff, policy enforcement.
- **Copilot Studio + Power Automate:** raskt, men for svak kontroll til 100 agents rolle-separasjon og batch-orkestrering.
- **Proprietære swarm-rammeverk:** gode for demo, men høy vendor-friksjon og svakere governance.

Anbefaling: bygg en **thin orchestration layer** i Python på Azure Functions/Container Apps, kall Azure OpenAI, lagre state i Cosmos DB/Dataverse, kø via Service Bus.

2) Agent-spesialisering

Ikke dupliser samme prompt. Bruk **rolle + delkontekst + eval-mål**:

- 100 agenter = 8-12 distinkte roller, hver med 8-12 varianter av fokus
- Eksempel: regulatory, cost, clinical risk, ops, security, procurement



- Hver agent får:
 - felles problem statement
 - egen “lens”
 - begrenset evidenspakke
 - eksplisitt output-skjema: `claim | evidence | confidence | risk | recommendation`

Aggregator: bruk en **critic/meta-agent** som rangerer, dedupliserer og produserer beslutningsnotat. Ikke la alle stemmer bli like.

3) Micro-case processing

Bruk **queue-based inbox**:

- Inngang: Service Bus queue
- Worker pool: ledige agenter henter case
- Routing: case metadata bestemmer rolle/mode
- Variabel kompleksitet: pre-classify case til “short/medium/deep”

Del gjerne template, men ha **rolle-specific system prompts**. Konsistens sikres med **fixed output schema + validation**.

4) Data/kontekst-injection

Bruk **RAG internt i Microsoft-grensen**:

- SharePoint/OneDrive/Exchange via Graph
- Dataverse som strukturell kilde
- Embeddings og retrieval i Azure AI Search
- **Ikke kopier rådata til eksterne API-er**

Kritisk: Entra ID, managed identity, private endpoints, DLP, audit logs, RBAC per agentklasse, dataminimering.

5) Observability og kost

Mål per agent: tokens, latency, fail-rate, retrieval hit-rate, confidence. Verktøy: Azure Monitor + App Insights + Log Analytics. ROI-optimalisering:

- cache av retrieval og mellomresultater
- batch kall der mulig
- små modeller for triage, store kun ved lav confidence
- stopp tidlig når beslutningsterskel er nådd



6) Transition

Realistisk: **6-12 uker pilot, 3-6 måneder produksjonsklar delvis swarm.** Quick wins:

1. 10-agent perspektivanalyse på et regulerings-/risikovalg
2. queue-basert small-batch case triage
3. governance dashboard med kost/latency/feil

Når ikke swarm? Når oppgaven er enkel, deterministisk, eller data er for sensitiv uten moden governance. Da er Copilot nok.

Neste steg:

1. Velg én konkret use case og bygg en 10-agent pilot i Azure.
2. Definer output-schema og eval-metrikk før første prompt.
3. Etabler sikkerhetsmodell med Entra ID, private endpoints og audit fra dag 1.