



id: SI_003

model: openai/gpt-5.4-mini

seeds:

04_risiko: mulighet

03_relasjon: konkurranse

05_handling: vent

01_perspektiv: strategisk

02_tidhorisont: langsiktig

Instans SI_003

Seeds: strategisk | langsiktig | konkurranse | mulighet | vent

For en **200-personers helseorganisasjon med veldig streng kontroll** er den strategiske hovedlinjen: **bygg swarm som en kontrollert Azure-arkitektur, ikke som et “plugin-økosystem” i Copilot Studio**. Vent litt med full skalering; start der muligheten er størst og risikoen lavest.

1) Arkitekturvalg

Best fit: Azure OpenAI + egne Python-orchestratorer (evt. Functions/Container Apps) med Entra ID, Key Vault, Private Link, Purview.

- **Azure OpenAI + Python:** mest fleksibelt for 100-agents parallellisme, logging, retries, policy, batching, evals.
- **Copilot Studio + Power Automate:** bra for arbeidsflyt, men svakt for ekte swarm—begrenset kontroll på agentrolle, parallelisme, observability og kost.
- **CrewAI/Anthropic Swarm:** rask prototyping, men svakere Microsoft-governance og større leverandør-/datastrategisk friksjon.

Konklusjon: behold Microsoft-investeringen ved å bruke **Azure OpenAI som inference-lag + egen orkestrering**.

2) Case 1: 100-agents perspektivanalyse

Ikke dupliser prompt. Gi hver agent:

- **rolle**
- **hypotese**



- **begrenset evidenspakke**
- **beslutningsmål**
- **output-schema**

Eksempel:

```
agents = [  
    {"role": "regulatory_reviewer", "goal": "find compliance blockers"},  
    {"role": "cost_analyzer", "goal": "estimate operational impact"},  
    {"role": "competitive_intel", "goal": "identify market implications"},  
]
```

Bruk en **hierarkisk aggregator**:

1. producer agents
2. kritiker-agent
3. synthesizer-agent
4. final decision memo med scorecards

Dette hindrer “100 meninger” og gir **konkurranse mellom perspektiver**.

3) Case 2: distributed micro-case-processing

Bruk **queue/inbox-model**:

- Dataverse/SQL som case-kilde
- Service Bus/Storage Queue som dispatch
- worker-agenter skaleres etter kølengde
- hver case får **samme base-template**, men dynamisk “context tiering”:
 - enkel case → liten prompt
 - kompleks case → ekstra evidens + second-pass agent

4) Data og kontekst

Ikke kopier sensitiv data til eksterne API-er. Bruk:

- **RAG i Azure AI Search**
- **Managed Identity**
- **Private networking**
- **Purview + DLP + audit logs**
- **redaction før prompting**
- **least-privilege per agent**



5) Observerbarhet og kost

Track per run:

- token/case
- latency per agent
- retry-rate
- disagreement score
- cost per accepted output

ROI-optimalisering:

- caching av embeddings og mellomresultater
- billig modell til triage, dyr modell kun for final synthesis
- batch-kjøring om natten
- hard stop ved lav confidence

6) Transition fra Copilot

0-6 uker: pilot på Case 2 med 1-2 prosesser

6-12 uker: Case 1 for én beslutningstype

3-6 mnd: produksjon med governance, evals, kostkontroll

Når ikke swarm? Ved lav kompleksitet, høy reguleringsrisiko uten robust datatilgang, eller når én god Copilot-workflow løser 80%.

Neste steg:

1. Velg én høy-verdi case per use case og definer output-schema.
2. Sett opp Azure OpenAI + queue + logging i en isolert pilot-subscription.
3. Lag eval-metrikker for kvalitet, kost og compliance før du skalerer.