



id: SI\_009

model: anthropic/claude-sonnet-4.6

seeds:

04\_risiko: mulighet

03\_relasjon: makt

05\_handling: implementer

01\_perspektiv: strategisk

02\_tidhorisont: langsiktig

## Instans SI\_009

**Seeds:** strategisk | langsiktig | makt | mulighet | implementer

### Strategisk hovedanbefaling: Azure OpenAI Service + Semantic Kernel, ikke Copilot Studio

For en 200-ansatt helseorganisasjon med batch-toleranse er **Copilot Studio et feilvalg for swarm**. Det er designet for enkelt-bruker-interaksjon, ikke parallell agent-orkestrering. Makten ligger i Azure OpenAI + Semantic Kernel (Microsoft-nativt, Python-basert, Entra ID-integrert).

### Arkitektur-kjernen

```
Azure Service Bus (queue)
  → Orchestrator-funksjon (Azure Functions)
    → N parallelle Semantic Kernel-agenter
      → Azure OpenAI (gpt-4o-mini for enkle cases, gpt-4o for komplekse)
      → Aggregator-agent (syntese)
      → Dataverse/SharePoint (output)
```

Data forlater **aldri** tenant. Azure OpenAI i Norway East med Private Endpoint eliminerer ekstern eksponering.



## Agent-spesialisering (Case 1): Ikke 100 like — 100 ulike linser

Unngå prompt-duplisering. Implementer **rolle-matrisen**:

```
AGENT_ROLES = {
    "regulatory_reviewer": {
        "system": "Du er GDPR/Pasientrettighetsloven-ekspert. Analyser KUN compliance-risiko.",
        "temperature": 0.1, # deterministisk
        "model": "gpt-4o"
    },
    "operational_cost": {
        "system": "Du er prosessøkonom. Kvantifiser ressursforbruk i NOK og timer.",
        "temperature": 0.3,
        "model": "gpt-4o-mini" # kostnadsoptimert
    }
    # ... 98 til
}
```

Aggregator-agenten får **kun strukturerte JSON-outputs**, ikke rå tekst — dette forhindrer kaos.

## Case 2: Inbox-modellen

Azure Service Bus med **competing consumers**-pattern. Hver melding inneholder case-ID + kompleksitetsscore (0-1). Score > 0.7 → gpt-4o, ellers gpt-4o-mini. Konsistens sikres via **shared Pydantic output-schema** — agenter deler template, ikke system-prompt.

## Observerbarhet

Azure Application Insights med custom metrics:

- `tokens_per_agent`, `cost_per_case`, `p95_latency`
- Alert ved token-forbruk > 150% av baseline
- Semantic Kernel har innebygd telemetri-hooks



---

## Reell failure mode

**Swarm er feil** når: problemet er lineært, kontekst-avhengig mellom steg, eller krever menneskelig eskalering underveis. Ikke bruk swarm på enkeltpasient-journalføring.

---

---

## Timeline

- **Måned 1-2:** PoC med 5 agenter, én case-type, Azure OpenAI + Semantic Kernel
  - **Måned 3-4:** Service Bus-integrasjon, 20 agenter, observerbarhet live
  - **Måned 5-6:** Skalering til 100 agenter, kostnadsmodell validert
- 

---

## Neste steg

1. **Etabler Azure OpenAI-instans i Norway East** med Private Endpoint og Entra ID-rolleoppsett denne uken
2. **Bygg rolle-matrisen** for de 10 mest verdifulle perspektivene i din primære case-type — ikke 100 fra dag én
3. **Implementer én Azure Function** med Semantic Kernel som prosesserer én case ende-til-ende med logging til Application Insights — dette er makten du bygger videre på