Building and transforming an existing Python project into a Google Agent



Joakim Salomonsson

EC Utbildning

Projektkurs - Google Agents

2025-10

# Abstract

An exercise to familiarize oneself with working with Google Cloud Platform and, more specifically, creating AI Google Agents. This report explores methods, tools, development kits, and more to understand the workflow from an end-to-end perspective. I dive into the use of Google’s Agent Development Kit, Vertex AI and AgentSpace, which are essential parts of the Google Agent creation flow.

Innehållsförteckning

[1 Inledning 1](#_u2z4jgp2ru8o)

[2 Teori 2](#_e7f5unw1zfui)

[2.1 Google Cloud Platform 2](#_sc8bshhv0c0s)

[2.1.1 Google AgentSpace 2](#_49dacckdk0aj)

[2.1.2 Vertex AI 2](#_wd9ik7j5nd06)

[2.1.3 Agent Engine 2](#_gnuo0bggimvv)

[2.2 ADK 2](#_hjylkru2ox05)

[2.2.1 Tools 2](#_h0bhsvfcwc50)

[2.2.2 Root Agent 2](#_vf0n8iosnm2j)

[2.2.3 Agent Starter pack 3](#_ls61jkhjt8in)

[2.3 Windows tools 3](#_goe1aw4hi0cm)

[2.3.1 Make 3](#_h66x3naaps5b)

[2.3.2 Git Bash 3](#_rs32yxcwc29e)

[2.3.3 uv 3](#_xskrivyub5z)

[2.4 Regex 3](#_j6sussj5qppq)

[3 Metod 4](#_76s316a67vjv)

[3.1 Projekt kontext 4](#_aqv873y3c3jb)

[3.2 Omvandla existerande projekt till en ADK agent 4](#_d3nldtscy1c6)

[3.2.1 ADK’s byggstenar 4](#_unupf5a1wuu1)

[3.2.2 Tool 5](#_5glso94ktnh4)

[3.2.3 Root Agent 5](#_bc6wx6bvnde8)

[3.2.4 Agent Starter pack 5](#_le0ndcbmftp5)

[3.3 GCP set up 5](#_t70551ewqbh1)

[3.3.1 Autentisera, bestäm projekt och skapa bucket 5](#_f8je968cj1lg)

[3.4 Deploya till Agent Engine 5](#_cd5u5p5es0ng)

[3.5 AgentSpace registrering 6](#_9knoej3rrkmx)

[4 Resultat och Diskussion 7](#_x3di81tzggsy)

[4.1 Prestanda vid testkörningar 7](#_a3c0uiv0b9z0)

[4.2 Datavolym vid testkörningar 7](#_22svjmjrzyf)

[4.3 Utdata-kvalitet och analys 8](#_jxkefrz8fft0)

[4.4 Diskussion 8](#_pz6y53sx12cx)

[4.5 Demo (lokal körning / test dry-run) 8](#_dtv1qqa8pt7c)

[4.6 Demo i Gemini Enterprise 9](#_rh8mnnre5mxj)

[5 Slutsatser 10](#_g2zdj0igy6x0)

[5.1 End-to-End Installation & Deployment Guide 10](#_wsesicwvlbc6)

[5.1.1 Förutsättningar 10](#_raijtbttv3bl)

[5.1.2 Klona ett färdigt ADK repo 10](#_c7lm2t180ihc)

[5.1.3 Skapa .env i projektroten 10](#_2bcv9nt4mjso)

[5.1.4 Skapa och aktivera miljö 11](#_cfcpsj2616gm)

[5.1.5 Installera beroenden 12](#_7xw29014v7t8)

[5.1.6 Installera Agent Starter Pack (ASP) 12](#_7w2tc2pbnq45)

[5.1.7 Kör ASP “enhance” (lägger till playground/deploy-stöd) 12](#_8gux03pb9b2)

[5.1.8 Lägg till testdata (lokalt) 12](#_qeuekssrh4pd)

[5.1.9 Justera dependencies för Agent Engine 12](#_w8ljf6q3gkj2)

[5.1.10 Säkerställ att lokala paket följer med på deployment 13](#_j1g0d25nmc8b)

[5.1.11 Lokalt test 13](#_qx21yg2wh56x)

[5.1.12 Google Cloud konfigurering 13](#_653mhw5bqbc0)

[5.1.13 Deploy till Vertex AI Agent Engine 14](#_205eoruxkwkf)

[5.1.14 Registrera och skapa agenten i AgentSpace 14](#_5eo6mq3ddiqu)

[5.2 Nästa steg 14](#_19u5suexlsd2)

[5.3 Vidareutveckling 15](#_syvk67rz3mvr)

[6 Självutvärdering 15](#_edkeppam4z9i)

[Appendix A 16](#_uaelv61nhxnk)

[Källförteckning 17](#_7a81u94ntmyl)

# Inledning

I vårt arbetsområde data science blir det mer och mer relevant med cloud computing och för att hålla sig relevant och uppdaterad är det viktigt att lära sig dessa verktyg. I detta arbete har jag valt att gå närmare inpå Google Cloud Platform och specifikt inom Vertex AI platformen som är till för Data Science och ML applikationer. I mitt arbete kommer jag undersöka hur man går tillväga för att skapa en Google Agent med ett existerande python projekt.

Syftet med denna rapport är utreda hur man gör för att end-to-end skapa en Agent som kan användas på Google’s AgentSpace applikation och granska vilka verktyg och vilka kunskaper som behövs, för att uppfylla syftet så kommer följande frågeställning(ar) att besvaras:

1. Hur går man tillväga för att skapa en Google Agent från start till slut?

**DISCLAIMER! Efter att jag utfört arbetet och skrivit denna rapport har Google genomgått en re-branding av deras Google AgentSpace produkt. Den heter numera “Gemini Enterprise”. All referens till Google AgentSpace i rapporten syftar alltså på “Gemini Enterprise”.**

# Teori

## Google Cloud Platform

Google Cloud Platform (GCP) är en molnplattform från Google som erbjuder ett stort antal tjänster för att bygga, utveckla och skala applikationer. I Googles dokumentation beskrivs cloud computing i huvudsak som:

“Cloud computing is the on-demand availability of computing resources (such as storage and infrastructure), as services over the internet. It eliminates the need for individuals and businesses to self-manage physical resources and only pay for what they use.” (Google Cloud, n.d.-a)

### Google AgentSpace

AgentSpace är en produkt i Google Clouds Platform som är riktat till företag som hjälp till anställda för att organisera sitt arbete över alla olika interna verktyg samt till att automatisera dagliga uppgifter. Google AgentSpace kan integreras över ett företags interna verktyg ([t.ex](http://t.ex). Slack, Jira, SalesForce, Sharepoint, Outlook) och kan agera som en sökmotor (m.m.) för dessa. (Google Cloud, n.d.-b)

### Vertex AI

Vertex AI är Google Clouds ML/AI-plattform som samlar verktyg för datavetenskap, maskininlärning och AI-applikationer. Plattformen används som grund för att köra och exponera agenter via Agent Engine och för att hantera ML-relaterade arbetsflöden (Google Cloud, n.d.-c).

### Agent Engine

Agent Engine är en del av Vertex AI platform som används för att utveckla och produktionsätta AI agenter som används i [t.ex](http://t.ex). google AgentSpace. Agent Engine paketerar ens kod till en körbar tjänst och där du får en endpoint. Den hanterar även loggning och skalning. Agent Engine används även i sin tur för att exponera agenten till AgentSpace (Google Cloud, n.d.-d).

## ADK

Agent Development Kit (ADK) är ett ramverk för att bygga agenter med tydliga tools (typade in-/utdata med Python type hints, ofta visas Pydantic i ADK i exempel), sessionshantering och ett “playground” för lokala test. ADK kan användas för att kapsla ens befintliga Python-logik i en agent som sedan kan distribueras via Agent Engine (Google, n.d.-e).

### Tools

I ADK är tools namngivna, typade funktioner som agenten kan anropa. Varje tool har en tydlig indata- och utdatamodell (dict + type hints) och kan jämföras med ett API-anrop i Python. Tools används för att bygga skräddarsydda agenter där det är exakt definierat vad agenten får göra (Google, n.d.-f).

### Root Agent

Root Agenten är den centrala delen i ett ADK som beskriver hur agenten uppför sig (instruktioner) och vilka tools den har. Root Agent registreras i en ADK-app och kan köras lokalt i playground samt distribueras till Agent Engine (Google, n.d.-g).

### Agent Starter pack

Agent Starter Pack (ASP) är ett färdigt paket med mallar och verktyg som gör att man snabbt kan köra lokalt (playground) och distribuera till Vertex AI Agent Engine. ASP innehåller Make-kommandon och ett deploy-skript som paketerar kod, beroenden och extra\_packages (Google, n.d.-h).

## Windows tools

### Make

Make kör definierade mål i en Makefile (t.ex. make install, make playground, make backend). (GnuWin32 Project, n.d.) Används ofta i projekt för att automatisera lokala körningar och deployment-steg.

### Git Bash

Git Bash är ett terminalskal (Bash) som följer med Git för Windows. Det ger UNIX-liknande kommandon (t.ex. bash, rm, cp) som ofta anropas inifrån Makefile-skript. (Atlassian, n.d.)

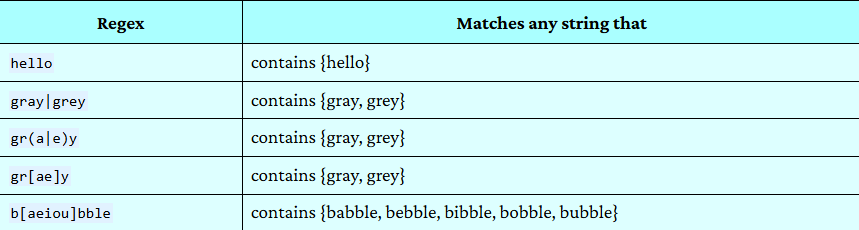
### uv

uv är ett verktyg för Python-beroenden och körning från pyproject.toml (kompletterar pip). Används ofta för att installera beroenden och uv run för att starta playground lokalt. (Astral, n.d.)

## 2.4 Regex

Regex, även benämnt som “Regular Expression” är ett koncept för att beskriva mönster i text och används ofta inom sökmotorer och text processesering. Heddings, A. (n.d.).

Se bild nedan för ett antal exempel för Regex:



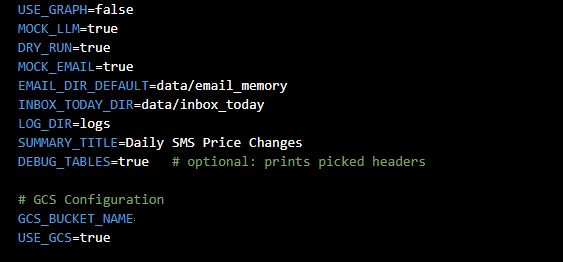
Toal, R. (n.d.)

# Metod

Målet med projektet var att återanvända det jag skapat i Avancerad python programmering del 2 och transformera det till en Google Agent.

## 3.1 Projekt kontext

Min startpunkt var projektet <https://github.com/salojoakim/supplierpriceautomation> vilket innehöll alla min scripts som jag skapat i Avancerad pythonprogrammering del 2 som är; util scripts, llm scripts och alla requirements. Utöver detta använder jag även leverantörs-email som test data vid lokala körningar som jag har hämtat hem sedan tidigare. I detta projekt har jag sedan tidigare lagt till ett antal flaggor baserat på om det är “live” eller inte. [T.ex](http://t.ex). så finns MOCK\_LLM=TRUE/FALSE. Se nedan för ett utdrag för vilka typer av olika typer flaggor som finns:

**

Om MOCK\_LLM=FALSE så använder projektet LLM för att parsa/tolka email och bilagor. Om flaggan är satt till TRUE så använder projektet regex för att parsa/tolka email och bilagor. USE\_GRAPH flaggan syftar på en anslutning till en outlook server. DRY\_RUN syftar på lokala testkörningar och i så fall använder testdatan som finns i katalogen data/email\_memory och jämför med data/inbox\_today. MOCK\_EMAIL används även vid testkörningar och istället för att skicka ut en HTML rapport via email så sparas rapporten ner lokalt i projekt katalogen.

## 3.2 Omvandla existerande projekt till en ADK agent

För att använda mitt automatiserade leverantörs-flöde i Google AgentSpace behövde jag skapa ett interface så att koden kan:

* Köras lokalt i ADK playground (Sessioner/REST)
* Koden kan packeteras och ompacketeras av Agent Engine
* Koden exponerar tydliga Tools/actions som AgentSpace kan kalla på
* Koden har metadata (instruktioner, scheman, status) som platformen förstår

### 3.2.1 ADK’s byggstenar

Det viktiga som behövs för att bygga ett ADK är framförallt ett script för “Tools” samt ett för “Root Agent”.

### 3.2.2 Tool

För att omvandla mitt existerande projekt behövde jag skapa ett Tool script. Ett Tool i ADK är en typad “wrapper” runt min befintliga Python-logik som gör den anropsbar för Agent Engine och AgentSpace. Utöver att anropa min tidigare kod exponerar Toolet ett tydligt kontrakt (dict + type hints för in- och utdata), vilket gör att plattformen kan validera indata, serialisera utdata samt visa verktyget i en lokal webbserver.

I detta projekt används Toolet för att orkestrera hela flödet, läsa email och bilagor, analysera tabeller (HTML/CSV/XLSX) och skriva JSON/HTML-rapporter samt returnerar sökvägar till genererade filer. Detta ger reproducerbara körningar lokalt och i Agent Engine, samt gör agenten enkel att konsumera från AgentSpace.

### 3.2.3 Root Agent

Jag skapade ett script [agent.py](http://agent.py) som innehåller vår Root Agent. Root Agenten är den platsen där man beskriver vad agenten är och vad den ska göra (dvs vilka tools den ska hantera och köra). ADK kan med hjälp av Root agent exponera ett HTTP API och en lokal webbserver med möjlighet att testa och köra vår daily pipeline.

### 3.2.4 Agent Starter pack

För att deploya till Google Cloud/Vertex AI och Agent Engine så finns ett Agent Starter pack som förenklar deployment. I detta starter pack så inkluderas ett script som bygger en ADKApp med min Root Agent. Den skapar även en konfig för Agent Engine ([t.ex](http://t.ex). storage/buckets, display\_name, requirements etc).

#### 3.2.4.1 ASP dependencies

För att kunna använda Agent Starter pack krävdes en del windows verktyg som jag installerade.

* make(Winget)
* Git(Bash)
* uv(Astral)

## 3.3 GCP set up

### 3.3.1 Autentisera, bestäm projekt och skapa bucket

För att gå vidare med att deploya i GCP behövde jag först autentisera med GCP och aktivera APIer som krävs för att deploya till Agent Engine.

*gcloud auth application-default login*

*gcloud auth application-default set-quota-project <PROJECT\_ID>*

*gcloud config set project <PROJECT\_ID>*

Exempel på API:er som krävdes att aktivera:

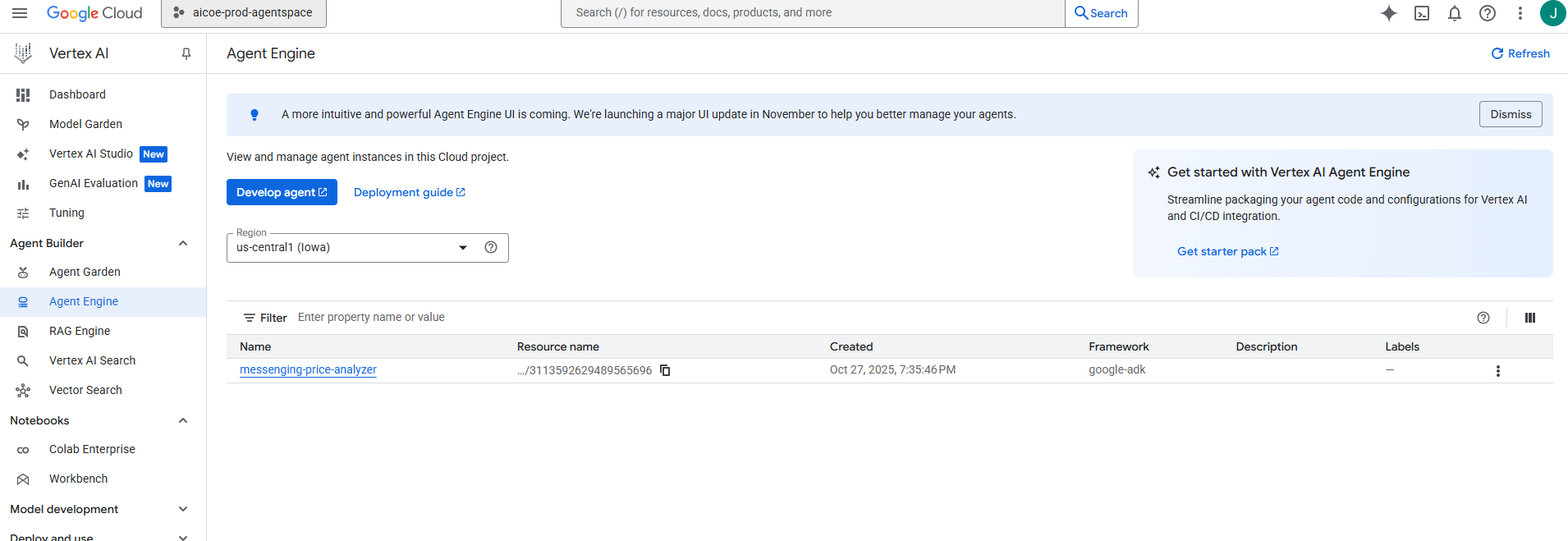
*gcloud services enable aiplatform.googleapis.com* [*storage.googleapis.com*](http://storage.googleapis.com) *iamcredentials.googleapis.com serviceusage.googleapis.com*

## 3.4 Deploya till Agent Engine

Efter körning av kommando: *make backend* paketeras min ADK-app (kod + beroenden) och det skapas/uppdaterar en Vertex AI Agent Engine.

Vad som sker efter *make backend*

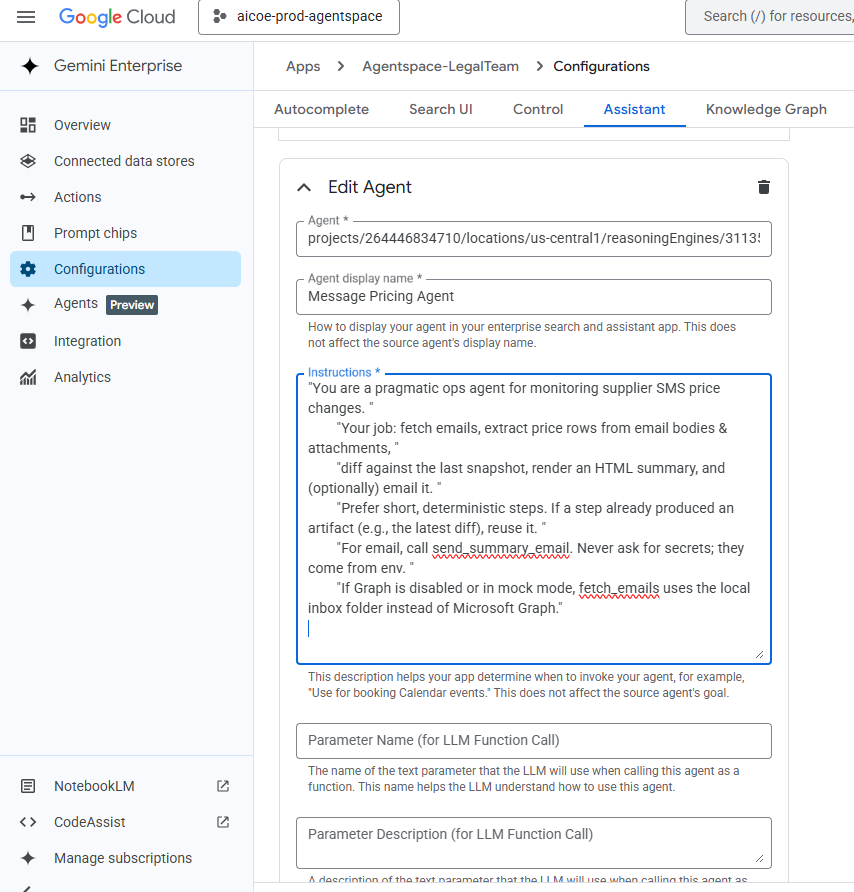
1. Export av “requirements”  
   uv export läser pyproject.toml och skapar en lätt .requirements.txt som används på byggservern.
2. Paketering av appen  
    Deploy-skriptet (agent\_engine\_app.py, genererat av ASP) samlar:  
   * Källkodspaket: ./adk\_agent, ./utils (och ev. ./llm) via extra\_packages
   * Allt som specificeras och krävs från .requirements.txt
   * Flaggor och konfiguration: env\_vars (t.ex. MOCK\_LLM=true, DRY\_RUN=true, LOG\_DIR=logs)
3. Skapande/uppdatering av Agent Engine  
   Skriptet anropar Vertex AI Agent Engine-API:et i vald region (t.ex. us-central1) och laddar upp paketet till en staging-bucket. Om en engine redan finns uppdateras den, annars skapas en ny.



*Vy i Agent Engine (Vertex AI)*

## 3.5 AgentSpace registrering

Att registrera en agent till AgentSpace via Agent Engine är relativt simpelt, man behöver ha tillgång till en Gemini Enterprise app. Gå till Configurations -> Assistant -> Agent -> Add item och fylla i ett par fält (se bild nedan). Det viktigaste är att i “Agent” fältet referera resursnamnet vilket är det som genereras efter att deployat till Agent Engine.



# Resultat och Diskussion

I ett tidigt skede testades LLM-baserad tolkning av mejl (Gemini). Vid körning på vårt testdatasätt (11 mejl) visade sig LLM-läget vara avsevärt långsammare (> 60 min och avbrutet), medan den regelbaserade lösningen (regex/tabellparser) slutförde körningen på under en minut. Resterande avsnitt redovisar mätresultat, datamönster i latest.json, diskussion och en video demo av regex testkörning.

## 4.1 Prestanda vid testkörningar

| Konfiguration | Antal mail | Total tid | Resultat |
| --- | --- | --- | --- |
| LLM = True (Gemini) | 11 | > 60 min | Avbruten körning |
| LLM = False (regex) | 11 | < 1 min | Slutfördes utan fel |

För extrahering av strukturerade prislistor är regelbaserad parsning snabbare och effektivare. LLM blir eventuellt relevant först när datan som hämtas är ostrukturerad (fri text, okända mallar).

## 4.2 Datavolym vid testkörningar

* Totalt antal extraherade rader: 8 683
* Antal unika leverantörer: 10

Rader per leverantör

| Leverantör / ämnesrad | Antal rader |
| --- | --- |
| Fw\_ [Nexmo] SMS Price change notification (1) | 2 421 |
| Fw\_ [Nexmo] SMS Price change notification | 2317 |
| Fw\_ CLX Direct-MO – Price Update | 1211 |
| Fw\_ GlobaHub\_ Price Change Notification (Wholesale) | 1164 |
| Fw\_ Cequens Coverage Update Notification | 700 |
| Fw\_ Rate Notification for Sinch Direct EUR EUR, 07 Sep 2025 | 151 |
| Fw\_ A2P – Ameex – CLX Weekly Report (2025-09-01 to 2025-09-07) | 64 |
| Fw\_ JT Global – Daily statistics for Sunday, 07\_09\_2025 | 15 |
| Fw\_ Silverstreet – Price Change Notification | 6 |
| Fw\_ Javna – Price Change Notification Account\_SinchDom1…SinchDom8 | 4 |

## 4.3 Utdata-kvalitet och analys

* new\_price: ifyllt i stort sett överallt.
* currency: saknas ofta. Valuta anges ibland i sidhuvud/rubrik i stället för i tabellkolumn.
* variation (increase/decrease): förekommer i vissa källor men saknas i många.
* Dubbletter: förekommer när samma data finns i både HTML och bifogad fil (CSV/XLSX).

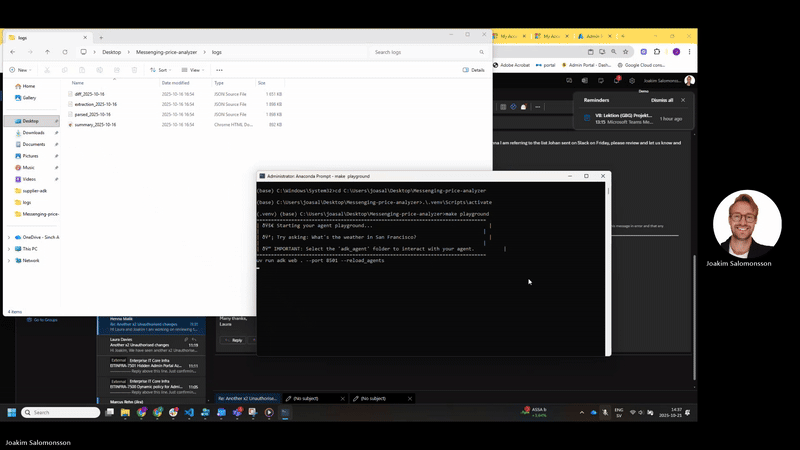
## 4.4 Diskussion

Regex är bättre eftersom källformaten är stabila (inte stor variation på test datan). LLM kan dock vara användbart i framtiden eftersom företaget skulle kunna få in nya leverantörer med fri text utan tabeller, andra filformat etc. Därför har jag valt att ha kvar LLM scripten i projektet för att i framtiden kunna köra på en hybridstrategi där regex är standard men kan falla tillbaka till LLM när datan inte känns igen.

## 4.5 Demo (lokal körning / test dry-run)

En kort video visar flödet (regex-läget), från läsning av mejl till genererad HTML-sammanställning och latest.json.

Länk till hela demot: [Demo-20251021\_143628-Meeting Recording.mp4](https://academediase-my.sharepoint.com/:v:/g/personal/joakim_salomonsson1_learnet_se/ERFmSzFoYI9Bpq3tKwI-4F4BGqPFwCNKT-vSooElYJGpZA?nav=eyJyZWZlcnJhbEluZm8iOnsicmVmZXJyYWxBcHAiOiJPbmVEcml2ZUZvckJ1c2luZXNzIiwicmVmZXJyYWxBcHBQbGF0Zm9ybSI6IldlYiIsInJlZmVycmFsTW9kZSI6InZpZXciLCJyZWZlcnJhbFZpZXciOiJNeUZpbGVzTGlua0NvcHkifX0&email=peter.lantz%40ecutbildning.se&e=wrL5jh)

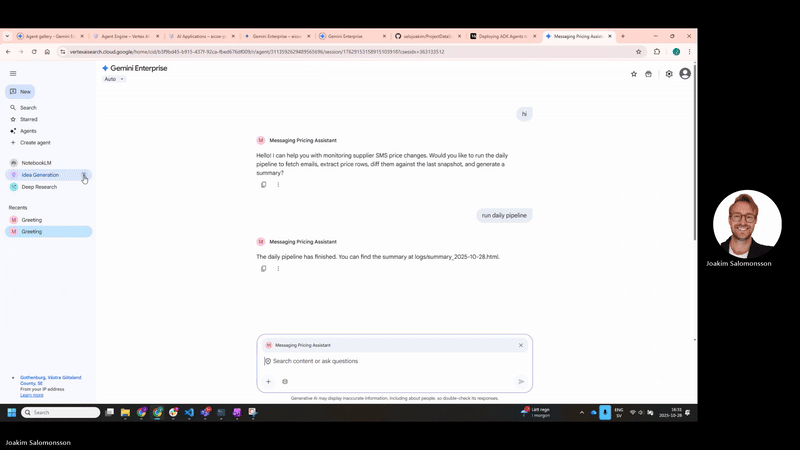


*(Kort version av demot i GIF format)*

## 4.6 Demo i Gemini Enterprise

En kort video som visar Gemini Enterprise och den publicerade agenten (demo mode dock!).

[Demo Gemini Enterprise-20251028\_163042-Meeting Recording.mp4](https://academediase-my.sharepoint.com/:v:/g/personal/joakim_salomonsson1_learnet_se/ER46CdOHqSNNoBseIWf3-JMBGq-pn92ekG6QLyyHNN6rPg?nav=eyJyZWZlcnJhbEluZm8iOnsicmVmZXJyYWxBcHAiOiJPbmVEcml2ZUZvckJ1c2luZXNzIiwicmVmZXJyYWxBcHBQbGF0Zm9ybSI6IldlYiIsInJlZmVycmFsTW9kZSI6InZpZXciLCJyZWZlcnJhbFZpZXciOiJNeUZpbGVzTGlua0NvcHkifX0&email=antonio.prgomet%40ecutbildning.se&e=vRDKX8)



*(Kort version av demot i GIF format)*

# Slutsatser

För att skapa en Google Agent krävs en hel del steg från start till slut. För att summera stegen har jag skapat en checklista från start till slut, se nedan:

## 5.1 End-to-End Installation & Deployment Guide

Den här guiden tar dig från ett nytt repo till lokal körning i ADK-playground och vidare till deployment på Vertex AI Agent Engine med Agent Starter Pack (ASP). Jag använde mig av Anaconda prompt och därför är cmd kommandon i guiden baserade på Anaconda prompt.

### 5.1.1 Förutsättningar

* Windows med Anaconda Prompt
* Python 3.10–3.12
* Git
* Make
* uv
* Google Cloud SDK (gcloud) + gsutil
* Ett Google Cloud konto med uppsatt projekt och fakturering aktiverad
* Några .eml-filer för lokala test

### 5.1.2 Klona ett färdigt ADK repo

*cd \sökväg\till\din\arbetskatalog*

*git clone https://example.com/your-repo.git*

*cd your-repo*

#### 5.1.2.1 Alternativ 2, omvandla ett existerande python projekt till ADK

Omvandla repot till en ADK agent

* Skapa ett agent paket
  + Skapa en Root agent
  + Skapa en “Tools”

### 5.1.3 Skapa .env i projektroten

USE\_GRAPH=false

MOCK\_LLM=true

DRY\_RUN=true

MOCK\_EMAIL=true

EMAIL\_DIR\_DEFAULT=data/email\_memory

INBOX\_TODAY\_DIR=data/inbox\_today

LOG\_DIR=logs

SUMMARY\_TITLE=Daily SMS Price Changes

DEBUG\_TABLES=true

# (valfritt) Microsoft Graph appbehörigheter

MS\_TENANT\_ID=

MS\_CLIENT\_ID=

MS\_CLIENT\_SECRET=

MS\_SHARED\_MAILBOX=

MS\_MAIL\_FOLDER=Inbox

MS\_DAYS\_BACK=1

MS\_TOP=100

# (valfritt) Gemini API-nyckel om LLM ska användas

GOOGLE\_API\_KEY=

# (valfritt) SMTP för utskick

SMTP\_HOST=

SMTP\_PORT=

SMTP\_USER=

SMTP\_PASSWORD=

SMTP\_FROM=

SMTP\_TO=

SMTP\_STARTTLS=true

### 5.1.4 Skapa och aktivera miljö

**venv**

*python -m venv .venv*

*.\.venv\Scripts\activate*

*python -m pip install --upgrade pip*

### 5.1.5 Installera beroenden

*pip install -r requirements.txt*

### 5.1.6 Installera Agent Starter Pack (ASP)

*pip install --upgrade agent-starter-pack*

### 5.1.7 Kör ASP “enhance” (lägger till playground/deploy-stöd)

Från projektroten:

*agent-starter-pack enhance*

**Rekommenderade svar i dialogen:**

1. Proceed? → Y
2. Template → Base template
3. Project directory to enhance → peka på din adk\_agent-app (eller motsv.)
4. Deployment target → Vertex AI (Agent Engine)
5. CI/CD runner → välj t.ex. GCP
6. Region → us-central1
7. Account/Project → bekräfta rätt GCP-konto och projekt

### 5.1.8 Lägg till testdata (lokalt)

Lägg några .eml i data\email\_memory eller data\inbox\_today för lokala körningar.

### 5.1.9 Justera dependencies för Agent Engine

I **pyproject.toml**:

google-adk>=1.15.0,<2.0.0

google-cloud-aiplatform[agent-engines]>=1.118.0,<2.0.0

protobuf>=5.26.1,<6.0.0

pandas>=2.2

beautifulsoup4>=4.12

lxml>=4.9

openpyxl>=3.1

xlrd>=2.0

python-dotenv>=1.0

opentelemetry-exporter-gcp-trace>=1.9.0,<2.0.0

google-cloud-logging>=3.12.0,<4.0.0

### 5.1.10 Säkerställ att lokala paket följer med på deployment

Om din Engine-app importerar lokala moduler (./utils, ./llm), lägg in dessa i extra\_packages i ditt deploy-skript (t.ex. agent\_engine\_app.py):

extra\_packages = ["./adk\_agent", "./utils"] # lägg till "./llm" om det används

Se också till att varje sådan mapp har en \_\_init\_\_.py så att de är giltiga Python-paket.

### 5.1.11 Lokalt test

Efter enhance finns make-targets:

*make install*

*make playground*

* Öppna browsern (vanligen<http://127.0.0.1:8501>)
* Välj adk\_agent-mapp när UI frågar
* Kör t.ex. verktyget run\_daily\_pipeline
* Kolla resultat i logs\summary\_YYYY-MM-DD.html

### 5.1.12 Google Cloud konfigurering

1. Autentisera och välj projekt:

*gcloud auth application-default login*

*gcloud config set project YOUR\_DEV\_PROJECT\_ID*

1. Skapa staging-bucket (samma region som Engine):

*set REGION=us-central1*

*gsutil mb -l %REGION% gs://YOUR-STAGING-BUCKET*

1. (Vid behov) Aktivera API:er:

*gcloud services enable aiplatform.googleapis.com storage.googleapis.com iamcredentials.googleapis.com serviceusage.googleapis.com*

1. Verifiera aktivt projekt:

*gcloud config get-value project*

### 5.1.13 Deploy till Vertex AI Agent Engine

Efter enhance finns oftast:

*make backend*

Detta innebär:

* Exporterar “requirements”(från pyproject)
* Paketerar appen + extra\_packages
* Skapar/uppdaterar en Agent Engine i rätt projekt/region
* Skickar med flaggor

### 5.1.14 Registrera och skapa agenten i AgentSpace

* I Google Cloud Console → Gemini Enterprise → Configuration → Assistant → Agent → Add new item
* Lägg till projektspecifika variabler (Resursnamn etc)

## 5.2 Nästa steg

Det finns en hel del kvar att göra på det här projektet men jag stötte på många hinder på vägen som blockerade mig att nå mitt initiala mål vilket var en färdig Agent registrerad på Google AgentSpace med företagets outlook server och en specifik mailgrupp som datakälla. Många organisationella blockers, som [t.ex](http://t.ex). få tillgång till GCP, förseningar med Outlook integrationen samt funktionalitet som ej var aktiverad. Väl när jag var igång med utvecklingen och jobbade på att bygga mitt ADK insåg jag att det var väldigt få i företaget som faktiskt hade kunskap om detta. Jag fick lista ut själv hur själva deployment gick till, ingen hade tidigare utfört detta. Det nästa steg efter Agent Engine deployment är i själva verket registreringen i Google AgentSpace. Till en början hade jag inte tillgång till företagets AgentSpace applikation vilket gjorde det svårt att publicera agenten. Utöver detta hade ingen tidigare gjort detta steg heller och jag hade workshops tillsammans med min LIA handledare och en Data Science kollega och försökte lösa detta tillsammans. Tillslut kom vi hela vägen i mål tack vare hjälp av en kollega som hade tillgång till att publicera ADK agenter på företagets Gemini Enterprise app.

### 5.3 Vidareutveckling

Tanken är även att vidareutveckla agenten och lägga till fler funktioner (tools) för att göra Agenten ännu bättre. Jag har idéer som [t.ex](http://t.ex):

* Analysera och ta fram de lägsta priserna till ett specifikt operatörsnät.
* Jämföra leverantörers erbjudanden i olika valutor.
* Dela upp HTML rapporterna i mer läs- och användarvänliga summeringar.
* Ta bort eventueller dubletter
* Currency-inferens per källa (t.ex. Nexmo EUR/USD beroende på rubrik)
* Variation-beräkning mot historik (CSV/SQLite) om källan saknar fältet.
* Och mycket mer!

# Självutvärdering

1. Utmaningar du haft under arbetet samt hur du hanterat dem.

Organisatoriska, få tillgång till licenser och verktyg. Förseningar med planerade integrationer (outlook/AgentSpace). Innan jag fick tillgång försökte jag planera allt så gott jag kunde så jag hade en tydlig plan vad jag skulle göra när jag fick tillgång. Angående den försenade integrationen fick jag istället göra en demo version och lära mig verktygen och plattformen i djupet så att när integrationen är klar så kan jag snabbt byta funktionalitet från demo/lokal till outlook.

1. Vilket betyg du anser att du skall ha och varför.

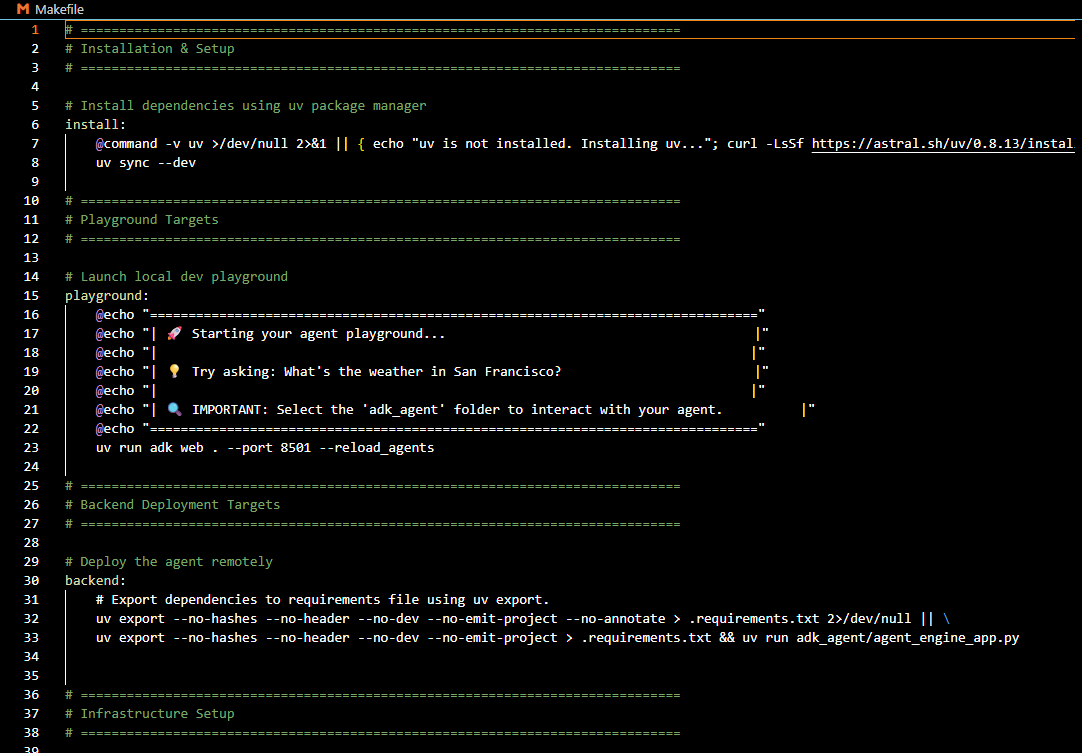
Jag tycker att jag förtjänar G eftersom jag har klarat alla kursen delmoment.

1. Något du vill lyfta fram till Antonio?

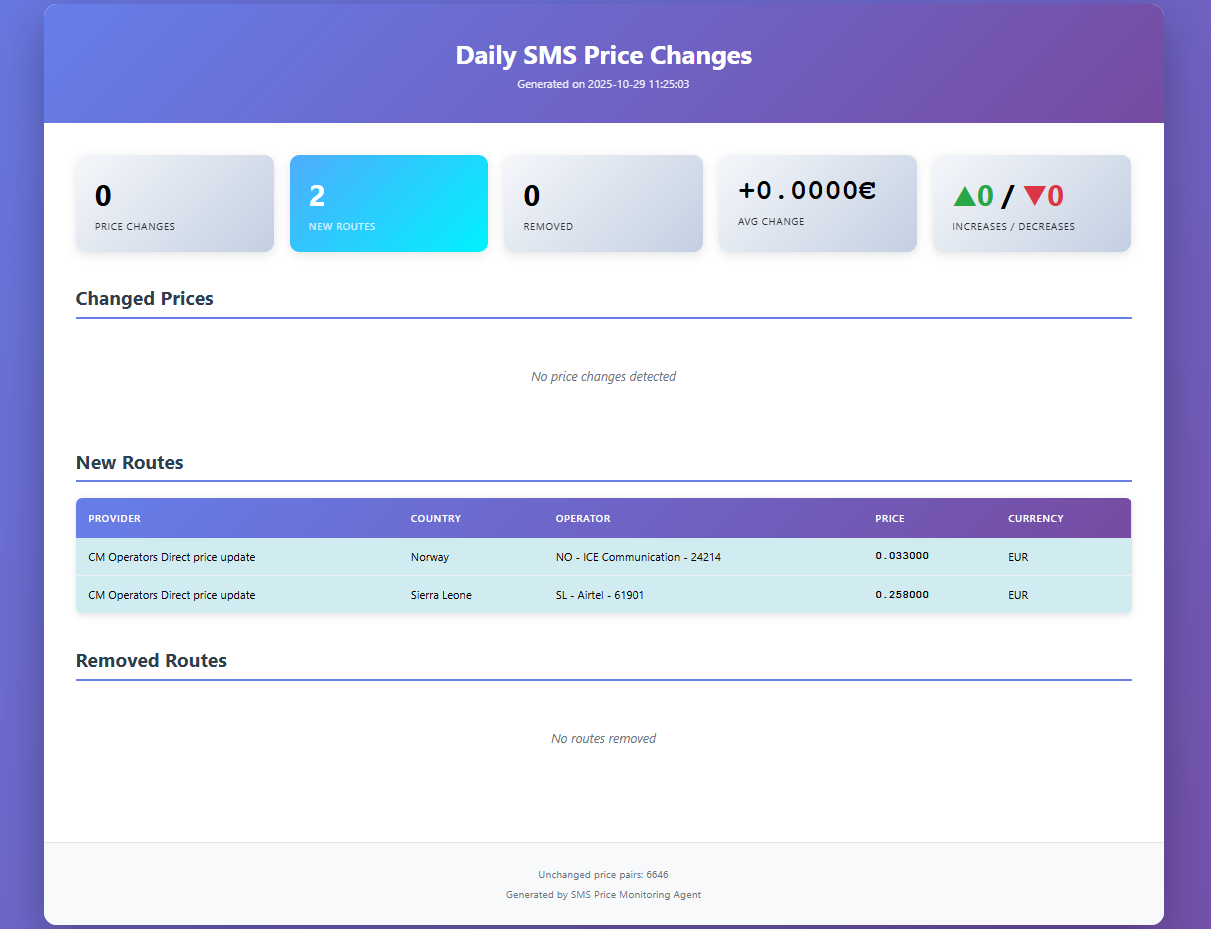
Inget speciellt, varit mycket självständigt arbete denna kurs och det har varit väldigt kopplat till mitt jobb vilket har varit kul projekt att få testa på.

# Appendix A

Make targets snippet:



Ny version av HTML summary:



# Källförteckning

Google Cloud. (n.d.-a). *What is cloud computing?* Google Cloud. Hämtad oktober 2025, från<https://cloud.google.com/learn/what-is-cloud-computing>

Google Cloud. (n.d.-b). *Google AgentSpace enables the agent-driven enterprise.* Google Cloud Blog. Hämtad oktober 2025, från<https://cloud.google.com/blog/products/ai-machine-learning/google-agentspace-enables-the-agent-driven-enterprise>

Google Cloud. (n.d.-c). *Vertex AI—Unified platform for AI & ML*. Google Cloud. Hämtad oktober 2025, från<https://cloud.google.com/vertex-ai/docs/start/introduction-unified-platform>

Google Cloud. (n.d.-d). *Agent Engine overview*. Google Cloud. Hämtad oktober 2025, från [https://cloud.google.com/vertex-ai/generative-ai/docs/agent-engine/overview](https://cloud.google.com/vertex-ai/generative-ai/docs/agent-engine/overview?utm_source=chatgpt.com)

Google. (n.d.-e). *Agent Development Kit (ADK) docs*. Hämtad oktober 2025, från [https://google.github.io/adk-docs/](https://google.github.io/adk-docs/?utm_source=chatgpt.com)

Google. (n.d.-f). *Tools* (ADK docs). Hämtad oktober 2025, från<https://google.github.io/adk-docs/tools/>

Pydantic. (n.d.). *Models*. Hämtad oktober 2025, från [https://docs.pydantic.dev/latest/concepts/models/](https://docs.pydantic.dev/latest/concepts/models/?utm_source=chatgpt.com)

Google. (n.d.-g). *Quickstart: Exposing a root agent* (ADK docs). Hämtad oktober 2025, från https://google.github.io/adk-docs/a2a/quickstart-exposing/

Google. (n.d.-h). *Agent Starter Pack (ASP) overview / docs*. Hämtad oktober 2025, från [https://google.github.io/adk-docs/](https://google.github.io/adk-docs/?utm_source=chatgpt.com) (eller din projektspecifika ASP-README om du vill vara mer exakt)

GnuWin32 Project. (n.d.). *Make (Windows port)*. Hämtad oktober 2025, från<https://gnuwin32.sourceforge.net/packages/make.htm>

Atlassian. (n.d.). *Git Bash tutorial*. Hämtad oktober 2025, från<https://www.atlassian.com/git/tutorials/git-bash>

Astral. (n.d.). *uv documentation*. Hämtad oktober 2025, från<https://docs.astral.sh/uv/>

Heddings, A. (n.d.). *How Do You Actually Use Regex?* How-To Geek. Hämtad oktober 2025, från [https://www.howtogeek.com/devops/how-do-you-actually-use-regex/](https://www.howtogeek.com/devops/how-do-you-actually-use-regex/?utm_source=chatgpt.com)

Toal, R. (n.d.). *Regular expressions*. Loyola Marymount University. Hämtad oktober 2025, från [https://cs.lmu.edu/~ray/notes/regex/](https://cs.lmu.edu/~ray/notes/regex/?utm_source=chatgpt.com)