n8n · Gemini · MCP Server 기반 GCP 로그 분석 자동화

n8n·Gemini·MCP Server 기반 로그분석 아키텍처

Choi Seong Gi linux1547@hanmail.net

목차

개요

목적

본 문서의 목적은 Google Cloud Platform(GCP) 프로젝트에서 발생하는 로그를 매일 지정된 시각에 자동으로 수집·분석하고, 그 결과를 보고서 형태로 전송하는 시스템을 구축하는 것입니다.
운영 현장에서 반복적이고 수작업으로 진행되던 로그 모니터링 업무를 자동화하여 운영 효율성을 높이고, 신속한 인시던트 탐지 및 실시간 의사결정 지원을 도모하고자 합니다.

배경

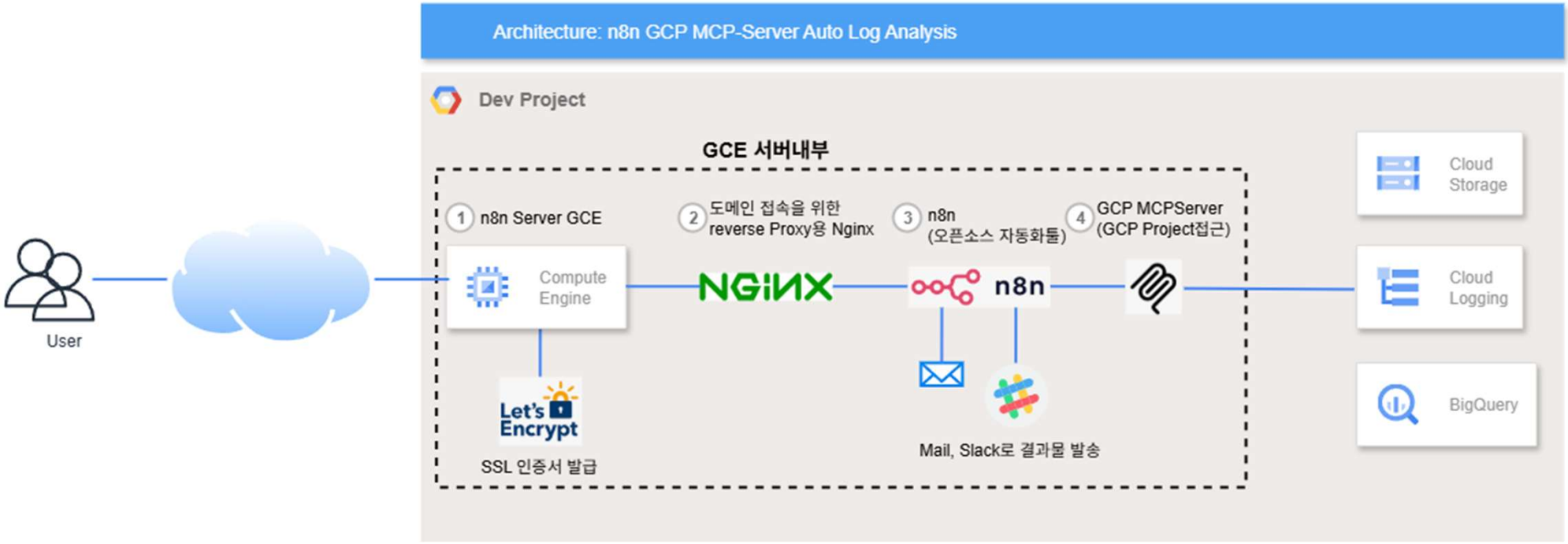
클라우드 환경에서는 서버, 네트워크, 애플리케이션 등 다양한 영역에서 방대한 로그가 발생합니다.
수작업으로 로그를 분석하고 관리하기에는 한계가 있습니다.
DevOps 및 Site Reliability Engineering(SRE) 문화가 확산되면서, 운영의 일관성과 모니터링 자동화의 필요성이 커지고 있습니다.
또한 AI와 오픈소스 도구 생태계의 발전으로, n8n, Gemini, MCP Server 같은 자동화·AI 도구를 활용해 중소규모 팀이나 개인도 손쉽게 자동화 솔루션을 설계할 수 있게 되었습니다.

고려사항

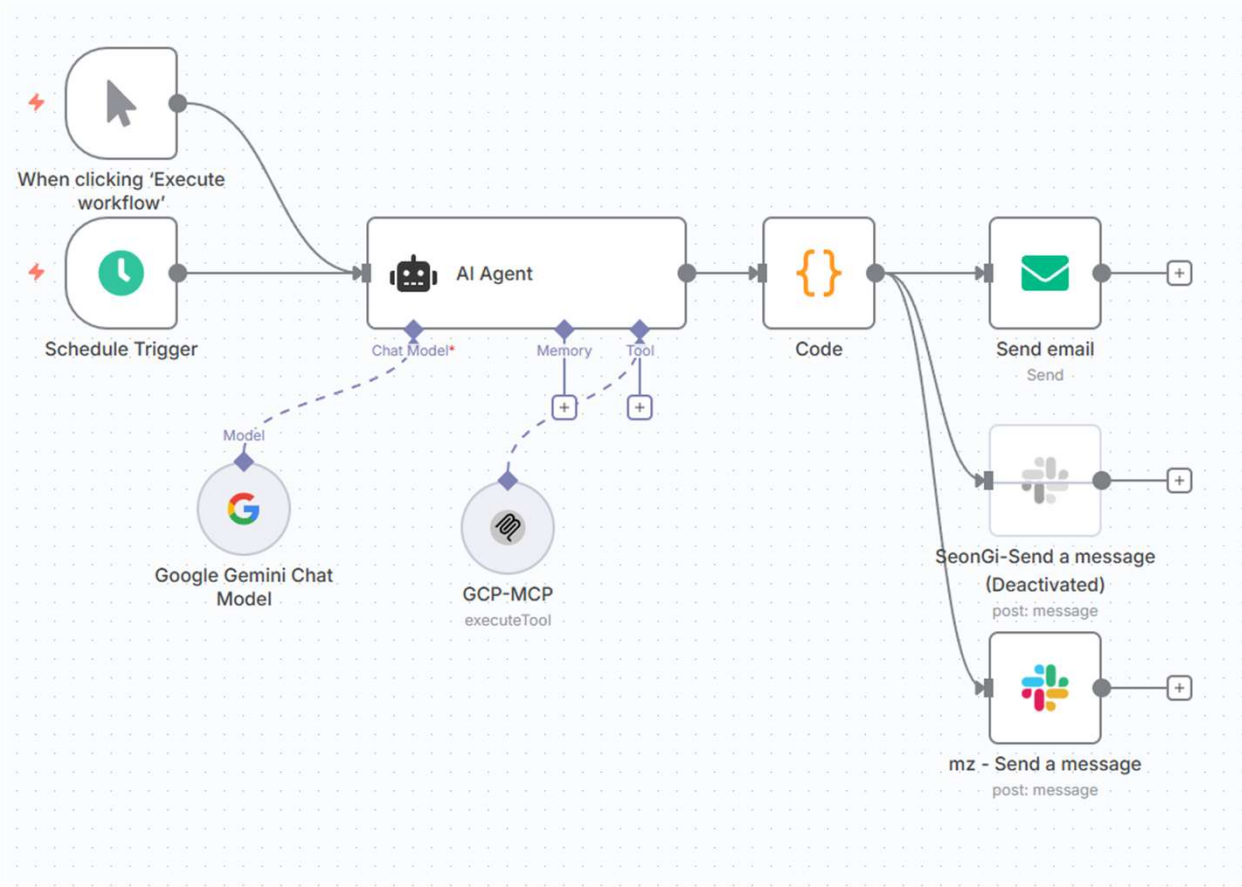
- 신뢰성: 매일 지정된 시각에 전일 로그를 누락 없이 수집하고 분석해야 합니다.
- 비용 및 운영 편리성: 오픈소스 도구와 GCP Cloud Logging을 결합해 비용 효율적으로 운영합니다.

구성도

n8n·Gemini·MCP Server 기반 자동화 아키텍처



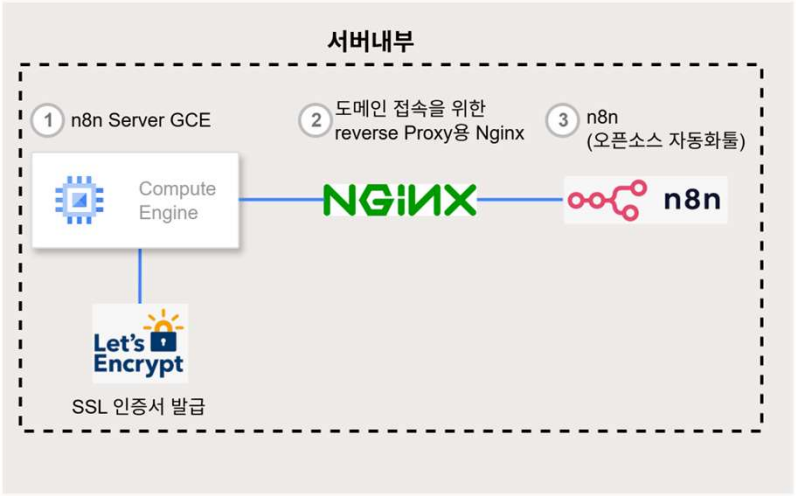
n8n 구성 흐름도


When clicking 'Execute workflow'
Schedule Trigger
AI Agent
Chat Model*
Memory
Tool
Model
Google Gemini Chat Model
GCP-MCP
executeTool
Code
Send email
Send
SeonGi-Send a message (Deactivated)
post: message
mz - Send a message
post: message

n8n · Gemini · MCP Server 기반 자동화 설치, 설정

배경과 목표

자동화 솔루션인 n8n을 통해 자동화를 쉽게 할 수 있습니다.
n8n의 설치는 npm으로 직접설치, Docker 사용, 관리형n8n사용 등이 있이 있습니다.
본문서에서는 빠르게 적용할 수 있는 Docker를 통해 n8n을 설치하고,
https도메인 접속을 위한 reverse Proxy용 Nginx 그리고 SSL인증서 발급을 위한
Let's Encrypt 에 대한 설치, 설정을 진행합니다.



전체 구성 개요

- n8n 서버는 Google Compute Engine(GCE)에 Docker 컨테이너로 배포되어 5678 포트에서 동작합니다.
- Nginx는 80, 443 포트에서 사용자 요청을 받아 SSL 처리 후 n8n으로 리버스 프록시합니다.
- Let's Encrypt(certbot): 무료 SSL 인증서 자동 발급 및 갱신
- MCP Server는 GCP 프로젝트 내 로그 수집과 명령 실행을 지원하는 도구로, n8n과 연동됩니다.

준비사항

- GCE상에서의 Ubuntu 서버
- GCE에서 Cloud API access scopes에 대해 Allow full access to all Cloud APIs 로 설정
- 도메인/DNS가 서버 공인IP에 연결
- 도메인 예시: gcp-ai.seonggi.kr
- 80, 443 포트 방화벽 오픈

nginx 설치, 설정

도메인 접속을 위한 reverse Proxy용 Nginx 설치와 설정

Nginx 설치

```
sudo apt update
sudo apt install nginx
```

설정 파일 위치

/etc/nginx/sites-available/n8n (신규 파일로 생성 후 수정)

```
server {
    listen 80;
    server_name n8n.도메인.kr;

    location / {
        proxy_pass http://localhost:5678;
        proxy_http_version 1.1; # Required for WebSocket
        proxy_set_header Upgrade $http_upgrade; # Required for WebSocket
        proxy_set_header Connection "upgrade"; # Required for WebSocket
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
}
```

설정 적용

설정파일을 심볼릭 링크로 연결하고 nginx 설정 파일 체크 후 구동합니다.

```
sudo ln -s /etc/nginx/sites-available/n8n /etc/nginx/sites-enabled/
sudo nginx -t   # 설정 테스트
sudo systemctl reload nginx
```

설정 확인

systemctl status, netstat 명령으로 nginx 정상동작과 80, 443 Port가 LISTEN되어있는지 확인합니다.

```
linux1547@n8n-test-gce:~$ systemctl status nginx
● nginx.service - A high performance web server and a reverse proxy server
     Loaded: loaded (/usr/lib/systemd/system/nginx.service; enabled; preset: enabled)
     Active: active (running) since Sun 2025-07-27 08:30:35 KST; 8h ago
       Docs: man:nginx(8)
    Process: 648 ExecStartPre=/usr/sbin/nginx -t -q -g daemon on; master_process on; (code=exited, status=0/SU>
    Process: 696 ExecStart=/usr/sbin/nginx -g daemon on; master_process on; (code=exited, status=0/SUCCESS)
   Main PID: 705 (nginx)
      Tasks: 3 (limit: 4618)
     Memory: 4.9M (peak: 10.6M)
        CPU: 461ms
     CGroup: /system.slice/nginx.service
             ├─705 "nginx: master process /usr/sbin/nginx -g daemon on; master_process on;"
             ├─706 "nginx: worker process"
             └─707 "nginx: worker process"

Jul 27 08:30:35 n8n-test-gce systemd[1]: Starting nginx.service - A high performance web server and a reverse >
Jul 27 08:30:35 n8n-test-gce systemd[1]: Started nginx.service - A high performance web server and a reverse p>

linux1547@n8n-test-gce:~$ sudo netstat -tnlp | grep 443
tcp        0      0 0.0.0.0:443             0.0.0.0:*               LISTEN      705/nginx: master p
linux1547@n8n-test-gce:~$ sudo netstat -tnlp | grep 80
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN      705/nginx: master p
tcp6       0      0 :::80                   :::*                    LISTEN      705/nginx: master p
linux1547@n8n-test-gce:~$
```

Let's Encrypt(SSL 인증서) 발급 및 자동 등록

Load balancer를 통해 Https(SSL) 통신을 할 수도 있지만, 본 문서에서는 최소한의 구성으로 진행하기 위해 GCE(VM)에 직접 SSL인증을 하는 방향으로 진행합니다.

인증서 발급시 80, 443 Port가 모두 방화벽에서 Open되어 있어야합니다.
80 Port가 Open필요한 사유는 Let’s Encrypt가 도메인 소유권 검증시 HTTP-01 챌린지방식을 사용하는데, 외부 인증서 발급 기관이 80번 포트(HTTP)를 통해 서버의 특정 경로에 임시 파일을 확인하기 때문입니다.

Let’s Encrypt 설치, 설정

```
sudo apt install certbot python3-certbot-nginx
sudo certbot --nginx -d n8n.도메인.kr
```

이 후 모든 http 트래픽을 https로 리다이렉트할지 묻는 질의에 Y로 진행합니다.

SSL 자동갱신

SSL이 자동으로 갱신되는지 아래의 명령어로 테스트

```
sudo certbot renew --dry-run
```

Let’s Encrypt 설치시 nginx의 설정파일을 자동으로 아래와 같이 수정합니다.
443 포트(HTTPS) - 인증서 적용
80 포트(HTTP) - 리다이렉션 및 404 처리
설정된 도메인은 https로 리다이렉션, 그외 요청은 404(거부)로 처리

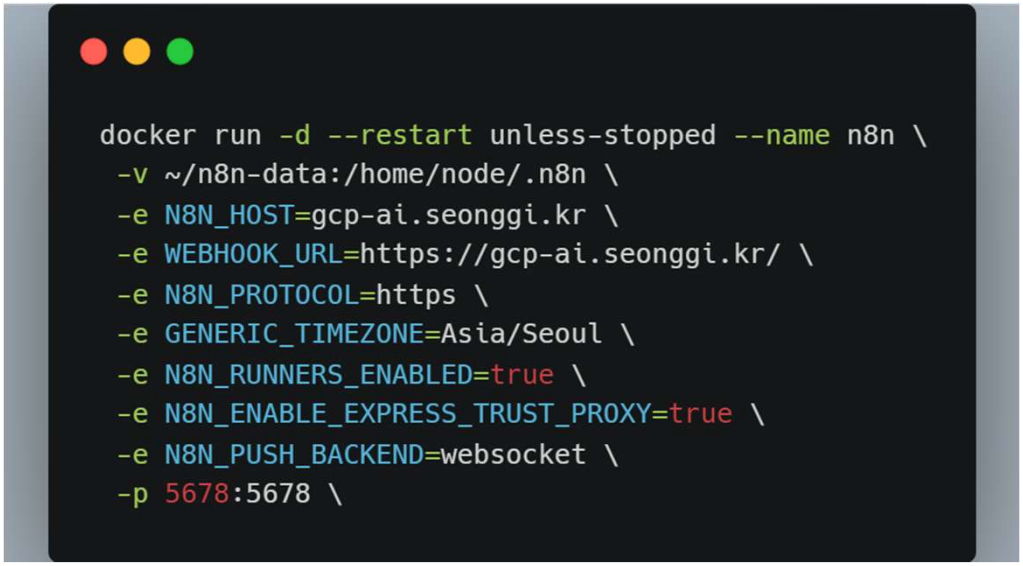
```
    listen 443 ssl; # managed by Certbot
    ssl_certificate /etc/letsencrypt/live/gcp-ai.seonggi.kr/fullchain.pem; # managed by Certbot
    ssl_certificate_key /etc/letsencrypt/live/gcp-ai.seonggi.kr/privkey.pem; # managed by Certbot
    include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot

}

server {
    if ($host = gcp-ai.seonggi.kr) {
        return 301 https://$host$request_uri;
    } # managed by Certbot


    listen 80;
    server_name gcp-ai.seonggi.kr;
    return 404; # managed by Certbot


}
```

n8n 설치, 설정

n8n솔루션을 도메인과 ssl로 접속하기 위한 준비는 모두 완료되었습니다.
이제 빠르게 적용할 수 있는 Docker를 통해 n8n을 설치하겠습니다.

설치 전 정보가 저장될 n8n-data 디렉토리를 하나 생성합니다.
저는 “/home/사용자/n8n-data” 로 생성하였습니다.
향 후 해당 디렉토리만 백업하면, 타 서버에도 동일하게 설정을 복원할 수 있습니다.

n8n docker 명령어



```
docker run -d --restart unless-stopped --name n8n \
 -v ~/n8n-data:/home/node/.n8n \
 -e N8N_HOST=gcp-ai.seonggi.kr \
 -e WEBHOOK_URL=https://gcp-ai.seonggi.kr/ \
 -e N8N_PROTOCOL=https \
 -e GENERIC_TIMEZONE=Asia/Seoul \
 -e N8N_RUNNERS_ENABLED=true \
 -e N8N_ENABLE_EXPRESS_TRUST_PROXY=true \
 -e N8N_PUSH_BACKEND=websocket \
 -p 5678:5678 \
```

Docker 옵션

옵션 명령어	설명
-d	컨테이너를 백그라운드(daemon)로 실행
--restart unless-stopped	컨테이너가 예기치 않게 중지되면 자동 재시작, 사용자가 직접 중지하면 재시작하지 않음
--name n8n	컨테이너 이름을 n8n으로 설정
-v ~/n8n-data:/home/node/.n8n	호스트의 ~/n8n-data 폴더를 컨테이너 /home/node/.n8n 경로에 마운트하여 데이터(설정, 워크플로우 등) 영속화
-e N8N_HOST=...	n8n 인스턴스의 접근 도메인/호스트네임 지정
-e WEBHOOK_URL=...	워크플로우 Webhook 접속용 URL 지정
-e N8N_PROTOCOL=https	n8n 인스턴스 접속시 사용하는 프로토콜(https)
-e GENERIC_TIMEZONE=Asia/Seoul	n8n 서버 타임존 설정(Asia/Seoul)
-e N8N_RUNNERS_ENABLED=true	n8n 실행환경 “runners” 기능 활성화
-e N8N_ENABLE_EXPRESS_TRUST_PROXY=true	리버스 프록시 환경일 때 Express 프록시 신뢰 활성화
-e N8N_PUSH_BACKEND=websocket	n8n 내부 실시간 푸시방식(웹소켓) 설정
-p 5678:5678	호스트 5678포트를 컨테이너 5678포트로 포워딩, 외부에서 n8n 웹에 접근 가능하게 함

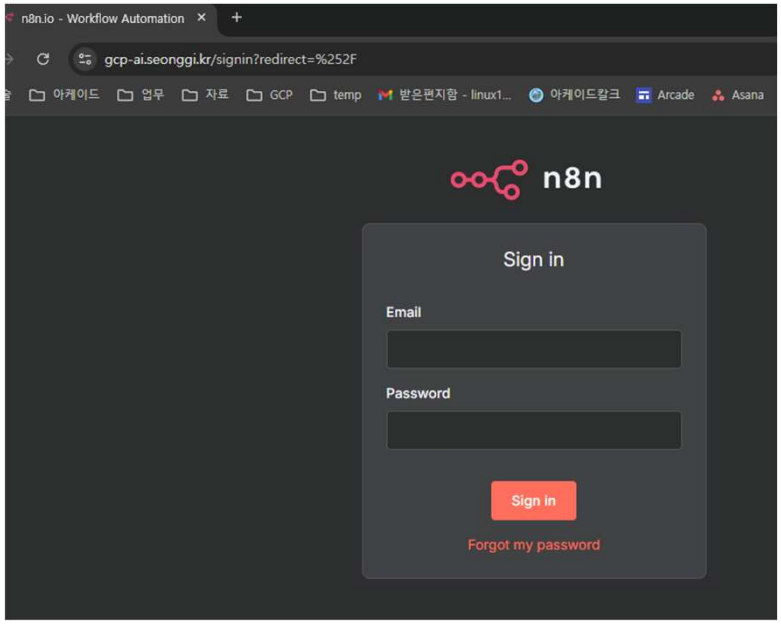
n8n 동작확인

동작확인

“docker ps”를 통해 n8n docker가 5678 Port에서 정상동작임을 확인합니다.

```
linux1547@n8n-test-gce:~$ docker ps
CONTAINER ID   IMAGE                      COMMAND                  CREATED      STATUS       PORTS                                       NAMES
2bdae4bd48aa   docker.n8n.io/n8nio/n8n    "tini -- /docker-ent…"   7 days ago   Up 2 hours   0.0.0.0:5678->5678/tcp, [::]:5678->5678/tcp   n8n
```

그리고 설정해둔 도메인 주소로 접속합니다.
맨처음에는 사용자의 email과 패스워드를 설정하는 화면이며, 이후로는 로그인 화면으로 접속됩니다.



mcp server 설치

MCP Server (Google Cloud MCP Server) 설치, 설정
Google Cloud MCP Server 는 Google Cloud의 공식 제품은 없습니다.

설치 Github docs : https://github.com/eniayomi/gcp-mcp

각 MCP Server들은 오픈소스 상태여서 정해진 설치 방안이 없습니다.
본 문서에서는 제일 간단한 “https://github.com/eniayomi/gcp-mcp“의 MCP Server를 사용합니다.

설치는 간단하게 Git에서 Clone 후 npm 설치를 통해 진행합니다.
- Git Clone
git clone https://github.com/eniayomi/gcp-mcp
cd gcp-mcp

- Install
npm install

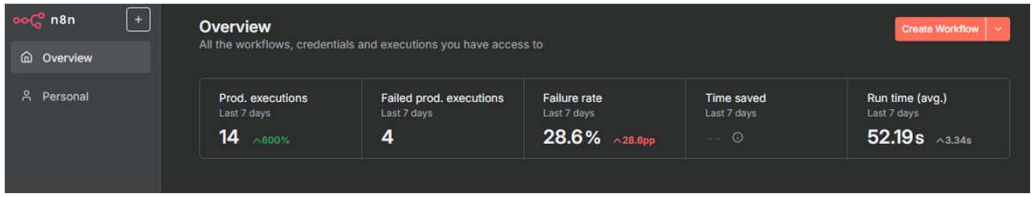
해당 디렉토리에서 설치했기에 “~/gcp-mcp” 디렉토리 내부에서
“npm list --depth=0” 실행하여 패키지가 잘 설치되었는지 확인합니다.

```
linux1547@n8n-test-gce:~/gcp-mcp$ npm list --depth=0
gcp-mcp@1.0.2 /home/linux1547/gcp-mcp
├── @google-cloud/bigquery@7.9.2
├── @google-cloud/billing-budgets@6.1.0
├── @google-cloud/billing@4.6.0
├── @google-cloud/compute@4.12.0
├── @google-cloud/container@5.19.0
├── @google-cloud/functions@3.6.1
├── @google-cloud/logging@11.2.0
├── @google-cloud/resource-manager@5.3.1
├── @google-cloud/run@1.5.1
├── @google-cloud/sql@0.19.1
├── @google-cloud/storage@7.15.2
├── @modelcontextprotocol/sdk@1.6.1
├── @types/node@22.13.9
├── google-auth-library@9.15.1
├── googleapis@146.0.0
├── ts-morph@24.0.0
├── tslib@2.8.1
├── tsx@4.19.3
├── typescript@5.8.2
└── zod@3.24.2
```

n8n 설정

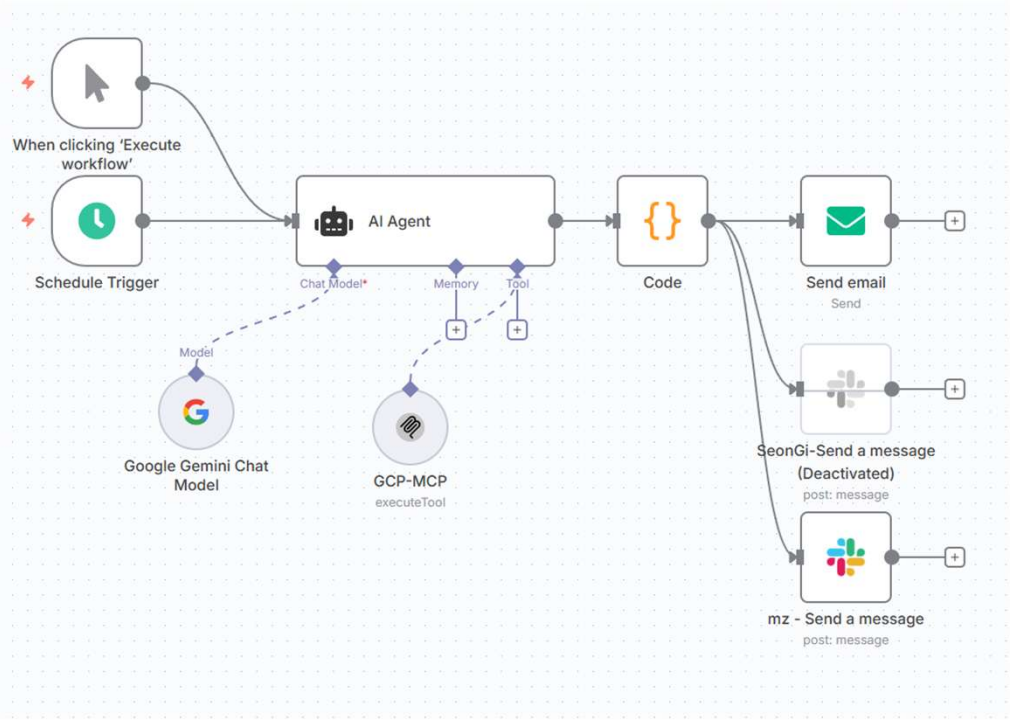
드디어, n8n과 mcp서버를 사용할 수 있도록 서버에서의 사전작업이 모두 끝났습니다.
이제 n8n 을 통해 gemini가 mcp server 를 통해 GCP 프로젝트의 로그를 분석하고, 리포팅 하는 방안을 시작하겠습니다.

n8n main화면에서 “Create Workflow”를 선택합니다.



흐름도는 다음과 같습니다.
“Schedule Trigger” -> AI Agent -> Chat Model -> tool(MCP Server) -> Chat Model -> Code -> Email, Slack

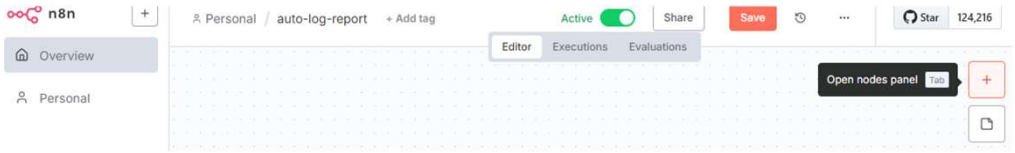
n8n을 통한 Gemini-MCP Server - code – Email, Slack 흐름도



Schedule Trigger

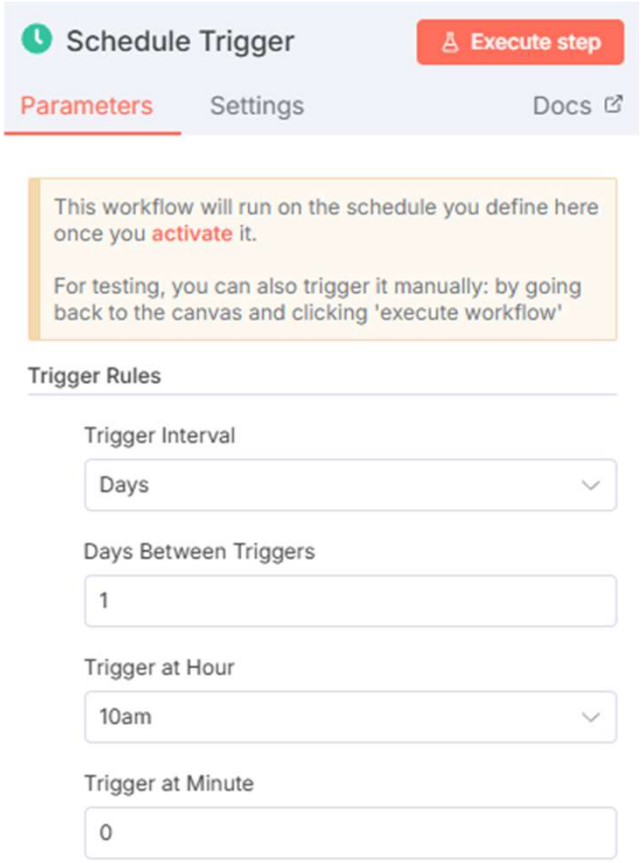
지정된 시각에 동작하기 위해서는 “Schedule Trigger”을 통해 설정합니다.
Windows의 작업스케줄러나 Unix, Linux Cron과 같습니다.

각 모듈의 추가는 오른쪽에 +버튼을 눌러 진행합니다.



Day, 시간, 분, 트리거 기간등을 설정합니다.
본 문서에서는 매일 10시 정각에 시작하도록 설정하였습니다.

Schedule Trigger 설정화면



AI Agent

중간에서 제일 중요한 역할을 하는 Ai Agent 설정 입니다.

sourece for Prompt : Define below / 아래의 정의내용으로 질의하기 위해 선택합니다.

Prompt : User가 AI Chat Model에 질의할 부분입니다.
상세한 내용은 다음 장에 설명하겠습니다.

Require Specific Output Format : 값 출력시 특정 값으로 변경하여 출력합니다. 저의 경우 줄바꿈이나 사람이 읽기 쉽도록 해당 옵션대신 Code를 선택하였습니다.

Enable Fallback Model : 기본 모델이 실패할 경우 대체 모델로 사용할 추가 언어 모델을 연결합니다.

System Message : 대화가 시작되기 전에 Chat Model에 전송되는 메시지

AI Agent 설정화면

AI Agent -> Chat Model

Chat Model은 Prompt와 system Message에 따라서 결과 값이 많이 달라집니다.

저의 경우 테스트를 통해 오른쪽 화면과 같이 했을 때 원하는 결과를 받았습니다.

Prompt를 구체적으로 상세하게 줄 수록 결과물이 달라집니다.

“Prompt”
1. GCP-MCP를 실행하여 받은 로그를 바탕으로 요약해서 보고해줘
2. 로그 분석 결과 보고서를 경고, 에러, 일반 순으로 번호를 매기고 사람이 읽기 쉽도록 가독성 좋게 정리해서 작성해
3. 해결방안이 있다면 아래에 별도로 작성해줘

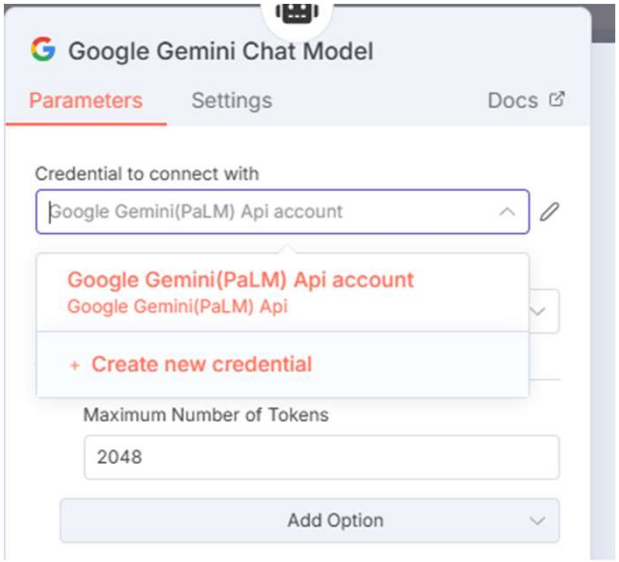
“system Message”
"너는 GCP 프로젝트를 운영하는 전문가로 매일 로그를 분석하는 업무를 하고있어"
"로그를 분석하여 장애를 예방하고, 엔지니어에게 선제적 조치를 취할 수 있도록 분석하는 것이 너의 업무야"
"보고서는 끝까지 작성해 중간에 끊기는일 없이"

"토큰이 2,000개 이상 쓰여질 것이 예상되면 경고, 에러에 대해서만 분석하고 토큰 때문에 일반 로그는 분석을 하지 못했다라고 작성해줘 다만 왠만하면 모든 로그에 대해 분석해"

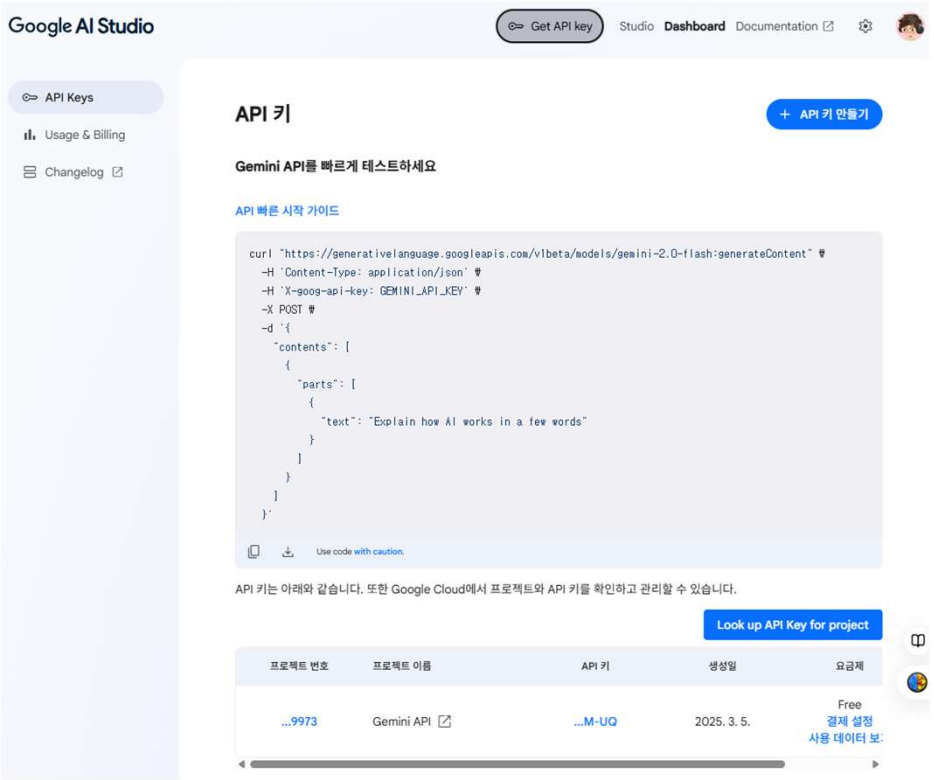
AI Agent -> Chat Model

Chat Model을 선택합니다.
본 문서에서는 Google Gemini 를 사용하도록 하겠습니다.

Gemini를 사용하기 위해 “Create new credential” 선택하여 신규 인증을 설정합니다.



https://aistudio.google.com
이동하여, Get API Key를 통해 API KEY를 발급받습니다.
* 기업환경의 경우 Vertex AI를 사용하시는 것을 추천합니다.

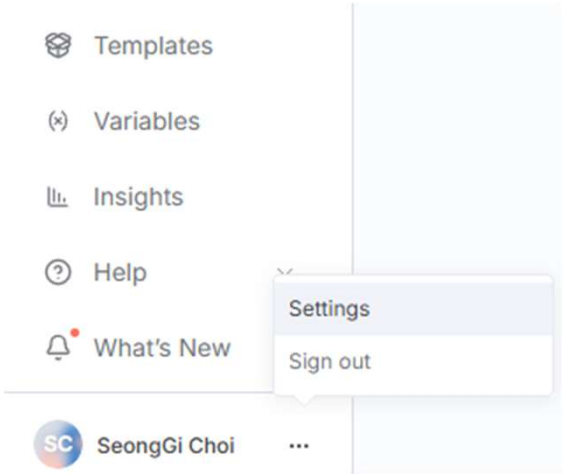


MCP Server

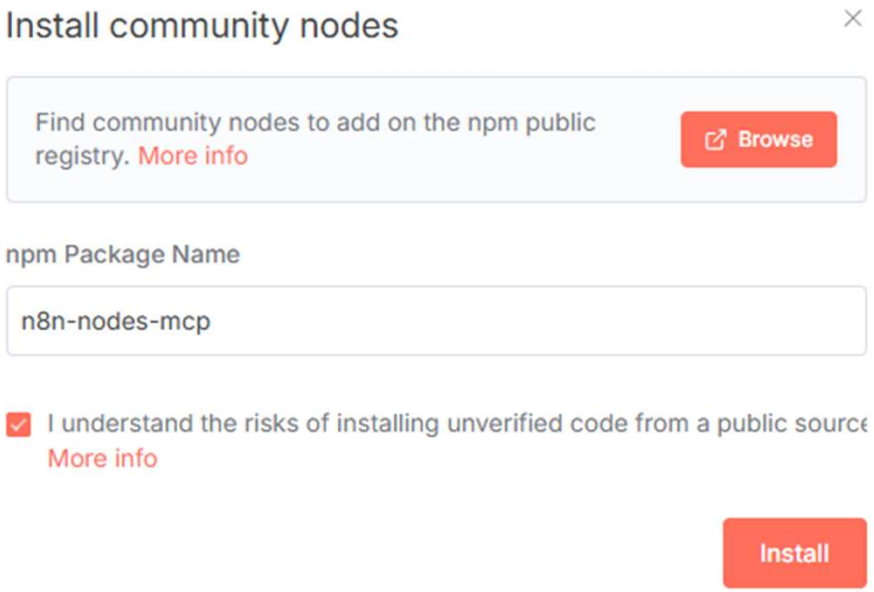
본 문서에 사용되는 MCP Server는 명령줄 기반 전송방식인 STDIO입니다.
n8n에서 바로 지원되지 않고, “n8n nodes mcp” 를 설치해야 됩니다.

아래는 공식 docs페이지 입니다.
https://www.npmjs.com/package/n8n-nodes-mcp

기본환경에서는 STDIO설정이 불가하기에, 하단 메뉴에서 settings -> Community nodes로 이동합니다.



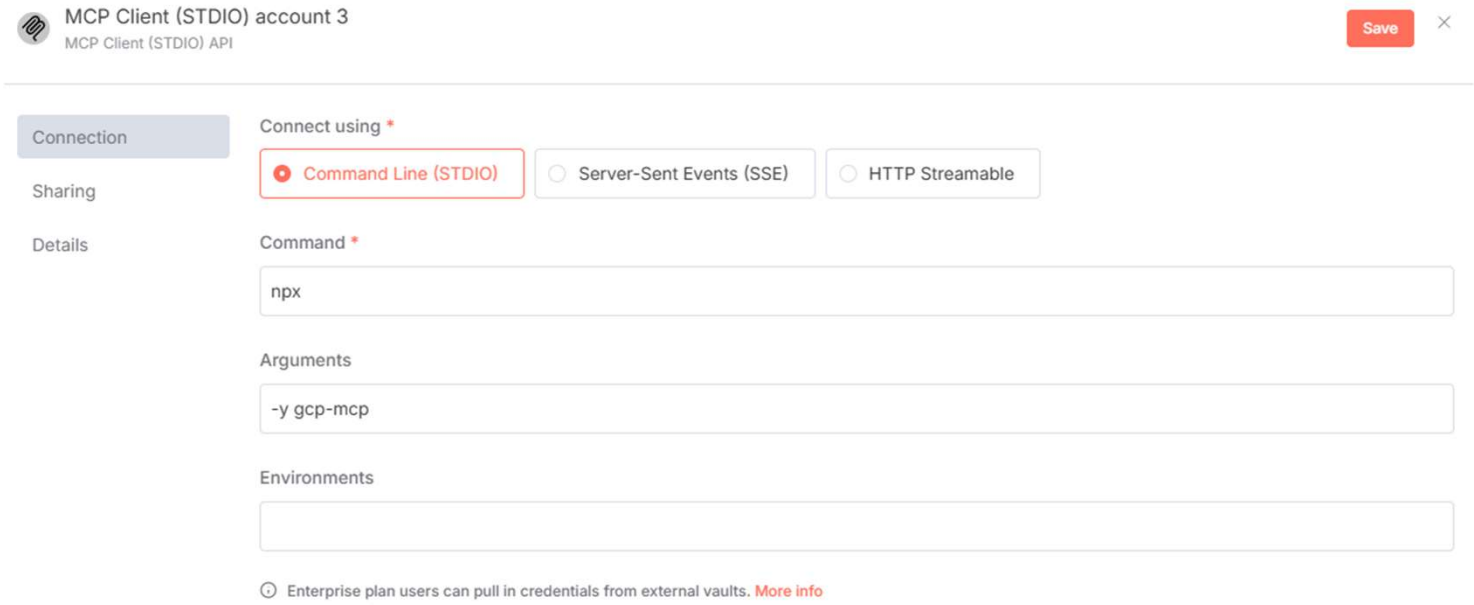
Community nodes에서 Install을 선택 후 아래 화면처럼 “n8n-nodes-mcp”를 설치 진행합니다.
설치 후에는 리스트에 표기됩니다.



MCP Server

AI Agent에서 Tool을 추가합니다.

n8n nodes mcp 모듈을 설치했기에 Command Line(STDIO)가 지원됩니다. Command : npx, Arguments : -y gcp-mcp 값을 입력합니다.



MCP Server

Tool Parameters는 아래와 같이 설정하였습니다.
내용이 길기 때문에 옆의 첨부파일을 참조하시길 바랍니다.

mcp-text.txt

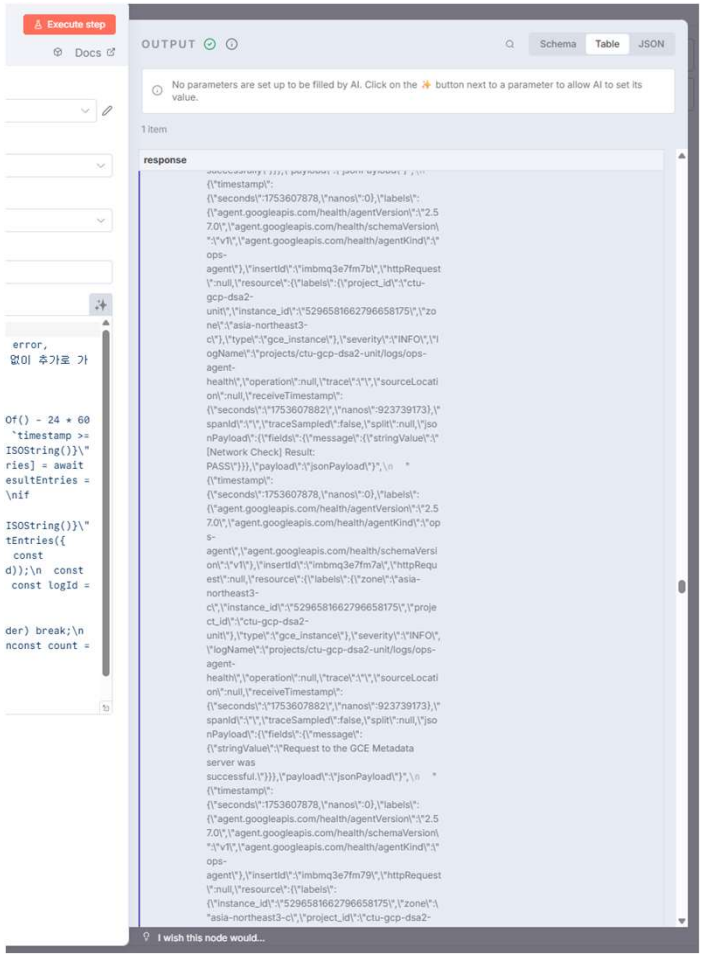
코드의 핵심은 아래와 같습니다.
"어제 날짜(UTC 기준)의 GCP 프로젝트 전체 로그(raw) 중 error, warning 로그를 우선 1000개까지 수집하고, 부족할 경우 일반 로그도 중복 없이 추가로 가져옵니다.

이 후 “Execute step”을 선택하여 MCP Server가 지정된 GCP Project에서 로그를 가져오는지 확인합니다. 제대로 되었다면, 오른쪽 화면과 같이 OUTPUT값을 확인할 수 있습니다.

만약 제대로 결과 값이 나오지 않는다면 API 호출이 불가할 가능성이 높습니다.

GCE에서 Cloud API access scopes에 대해 Allow full access to all Cloud APIs 로 설정합니다.

*Cloud Logging API 한계로 한번에 가져올 수 있는 로그의 수는1,000개입니다.



Code

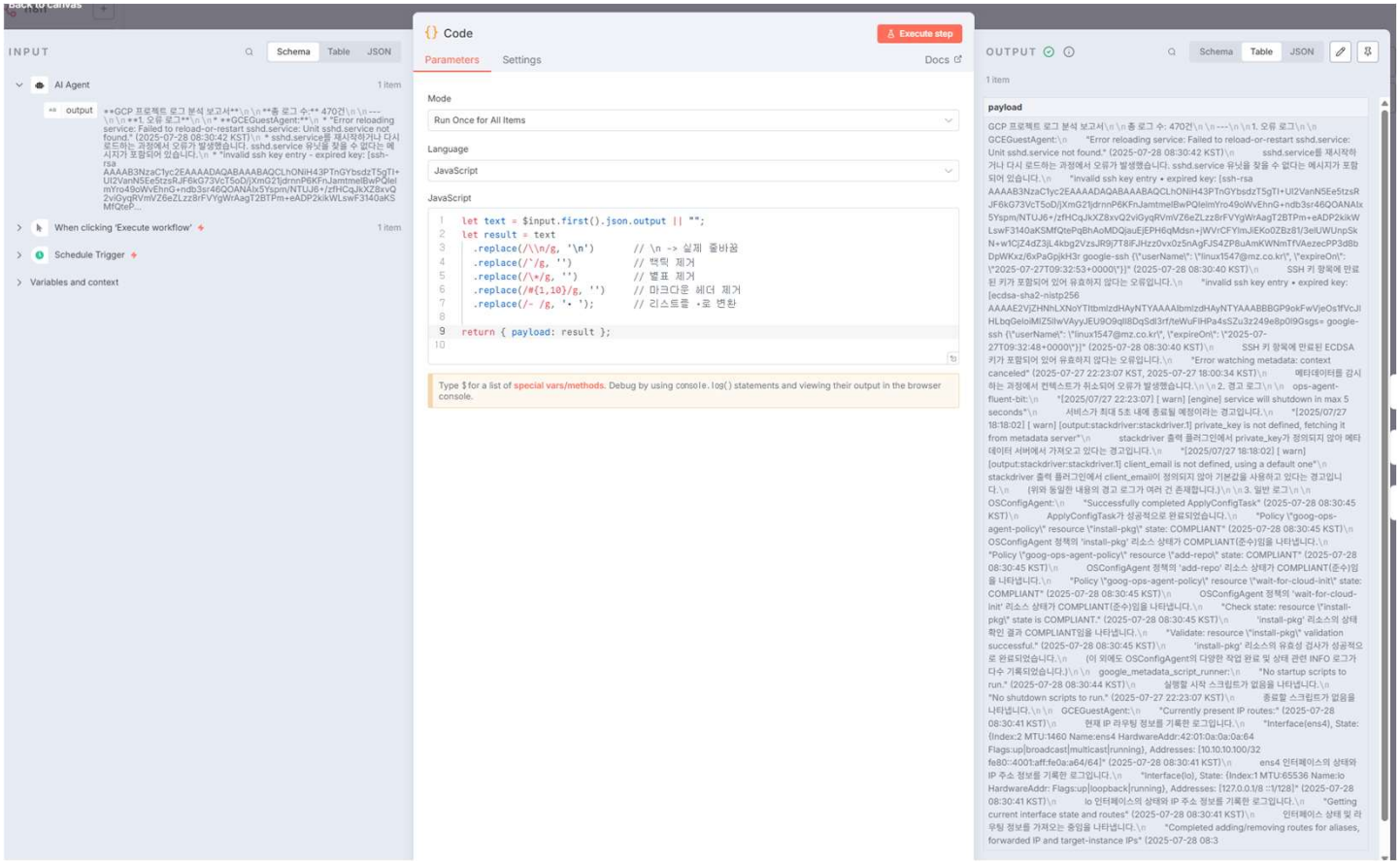
이제 MCP Server -> AI Agent(Chat Model)을 통해 나온 답변을 사용자에게 전달해주는 부분입니다.
그대로 받으면 줄바꿈과 별표등 사람이 보기 어렵기 때문에 아래와 같은 javaScript를 통해 사람이 보기 편하게 변경해줍니다.

```
let text = $input.first().json.output || "";
let result = text
  .replace(/\\n/g, '\n')        // \n -> 실제 줄바꿈
  .replace(/`/g, '')            // 백틱 제거
  .replace(/\*/g, '')           // 별표 제거
  .replace(/#{1,10}/g, '')      // 마크다운 헤더 제거
  .replace(/- /g, '• ');        // 리스트를 •로 변환

return { payload: result };
```

Mode는 모든 환경에 대해 한번만 실행되도록 “Run Once For All Items”를 선택합니다.

Code

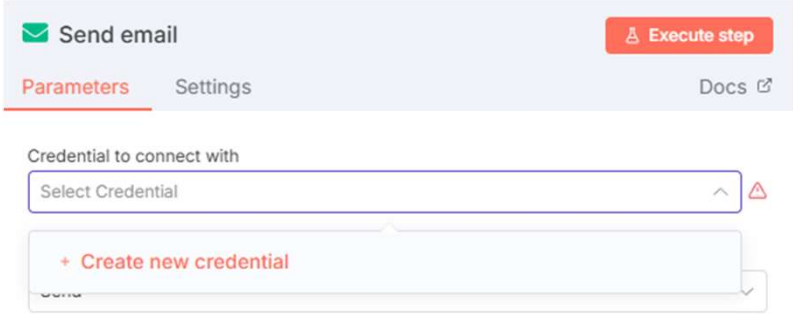
“Execute step”로 실행하여, Output이 잘 되는지 확인합니다.



Mail

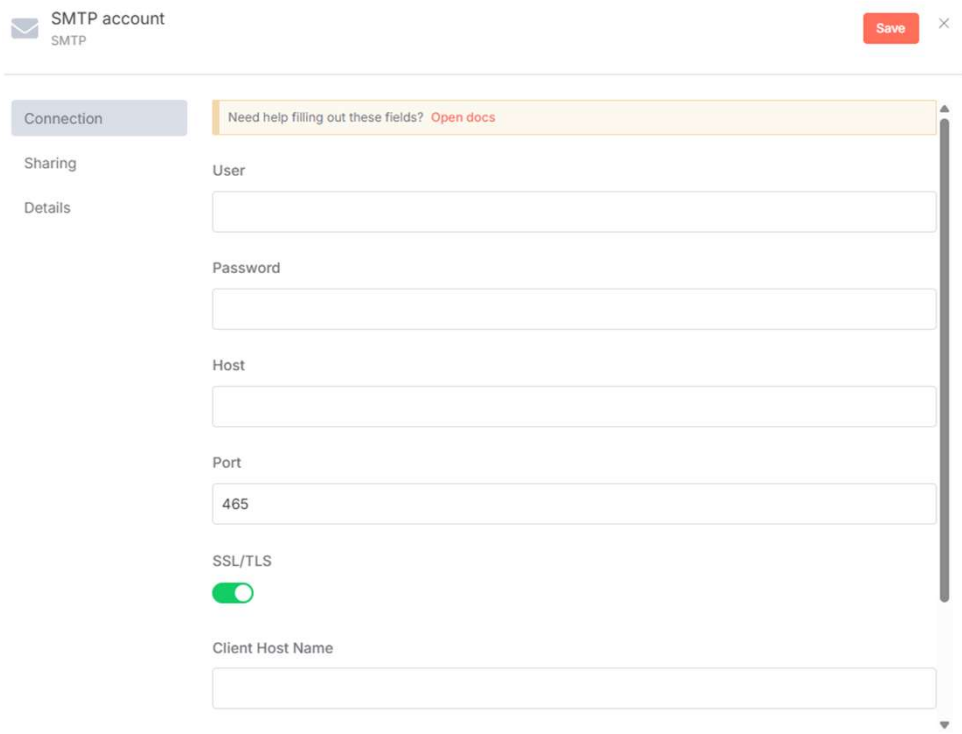
이제 결과물을 메일로 보낼 차례입니다.
본문서에서는 Gmail을 이용하겠습니다.

Send email 을 생성하고, Create New credential을 통해 새로 인증을 받습니다.



인증 설정은 오른쪽 화면과 같이 진행합니다.

User : gmail주소
Password : 앱비밀번호에서 설정한 비밀번호 (다음장에서 설명)
Host : smtp.gmail.com
Port : 465 (기본 값)
SSL/TLS : Enable (기본 값)

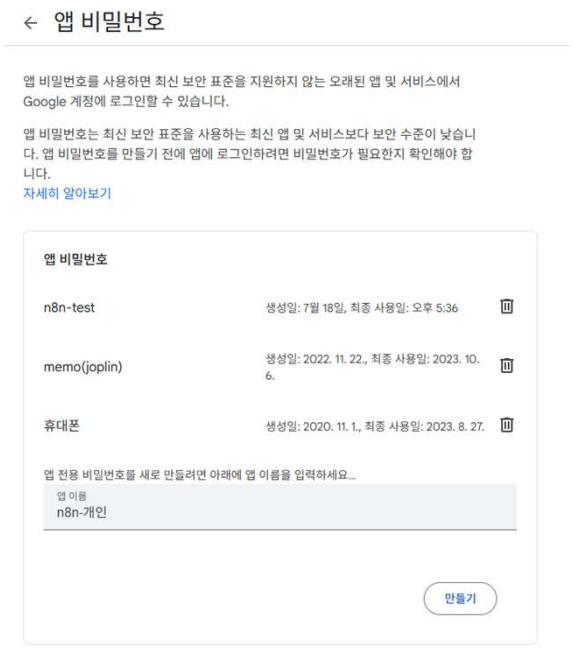


Mail

Gmail에서 외부에서 smtp사용시 앱비밀번호에서 설정한 비밀번호로 진행되어야합니다.

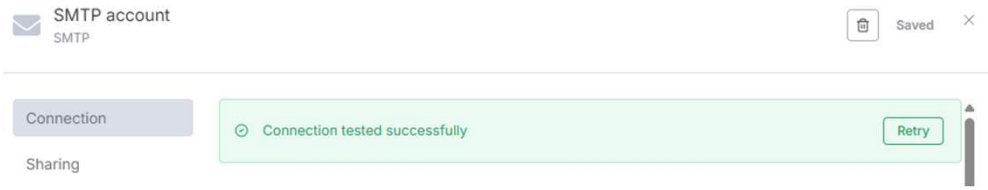
https://myaccount.google.com/apppasswords

알아보기 쉽도록 설정한 후에 만들기를 클릭합니다.



생성된 비밀번호를 n8n send email 패스워드 부분에 입력합니다.

Smtp에서 연결테스트를 확인한 후 마무리합니다.



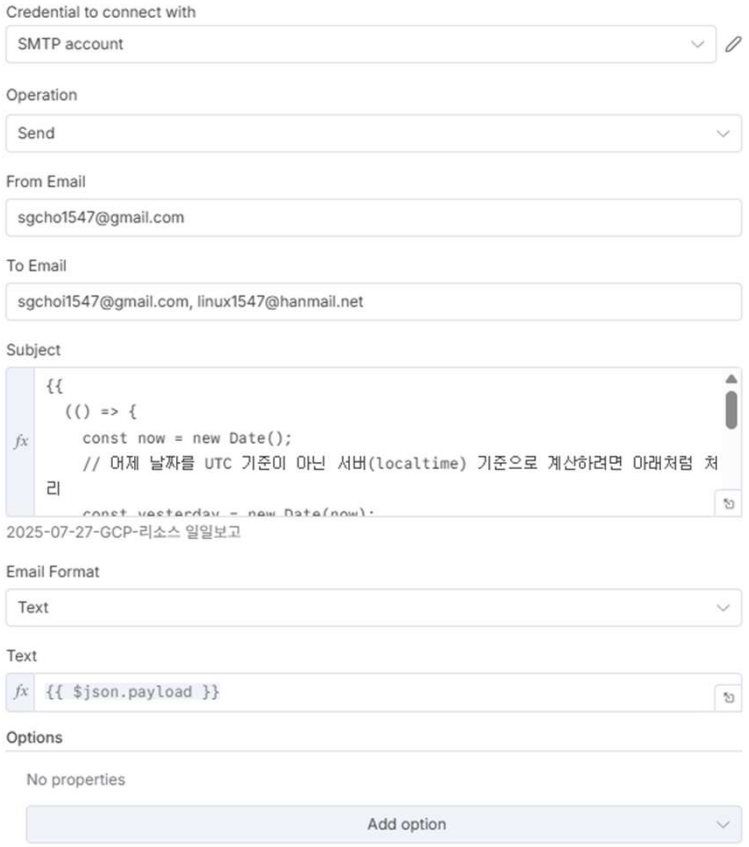
Mail

매일 새로운 날짜로 Mail을 발송하기 위해 아래와 같이 JavaScript 형태로 입력을 합니다.

```
{{
  (() => {
    const now = new Date();
    // 어제 날짜를 UTC 기준이 아닌 서버(localtime) 기준으로 계산하려면 아래처럼 처리
    const yesterday = new Date(now);
    yesterday.setDate(now.getDate() - 1);
    // YYYY-MM-DD 형식 생성
    const yyyy = yesterday.getFullYear();
    const mm = String(yesterday.getMonth() + 1).padStart(2, '0');
    const dd = String(yesterday.getDate()).padStart(2, '0');
    return `${yyyy}-${mm}-${dd}-GCP-리소스 일일보고`;
  })()
}}
```

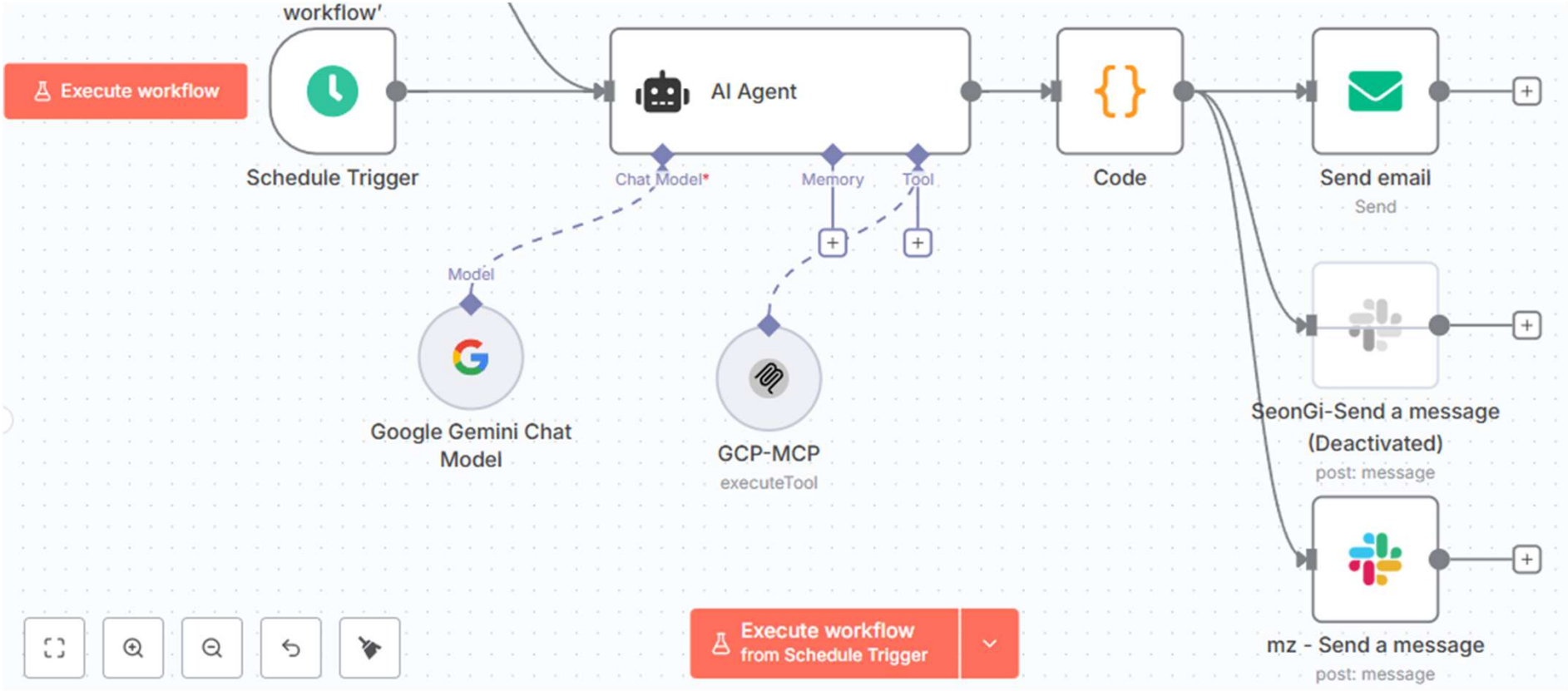
아래와 같은 양식으로 메일 제목이 설정되어 발송됩니다.

최종 발송을 하기 위한 설정



테스트

이제 정상동작이 되는지 테스트를 진행합니다.
“Execute workflow”클릭하여 n8n 워크플로우를 실행합니다.



최종결과

2025-07-27-GCP-일일보고

1 개의 메일

sgchoi1547@gmail.com <sgchoi1547@gmail.com> 2025년

받는사람: sgchoi1547@gmail.com, linux1547@hanmail.net

로그 분석 결과 보고서

1. 전체 로그 요약

총 470개의 로그가 수집되었으며, 주요 내용은 GCE Guest Agent 및 OSConfigAgent 관련 로그입니다.
특히, SSH 관련 오류 및 OSConfigAgent의 정책 적용 관련 정보가 확인되었습니다.

2. 로그 분석 결과

1) 오류 (ERROR) 로그

GCEGuestAgent • Error reloading service: Failed to reload-or-restart sshd.service: Unit sshd.service not found..
SSHD 서비스 재시작 또는 리로드 과정에서 sshd.service를 찾을 수 없다는 오류가 발생했습니다. 이는 SSH 서비스 관련 설정 문제 또는 서비스 파일의 누락을 나타낼 수 있습니다.
GCEGuestAgent • invalid ssh key entry • expired key: [username]:[key type]... google-ssh {username, expireOn: [date]}
SSH 접속에 사용되는 키의 유효기간이 만료되었다는 로그입니다. expireOn 필드에 기록된 날짜를 기준으로 만료된 키를 사용하려 할 때 발생하는 문제입니다.
GCEGuestAgent • Error watching metadata: context canceled
GCE 메타데이터를 감시하는 과정에서 컨텍스트가 취소되어 오류가 발생했습니다. 이는 일시적인 문제일 수도 있으나, 지속될 경우 GCE 인스턴스와 메타데이터 간의 통신 문제를 나타낼 수 있습니다.

2) 경고 (WARNING) 로그

ops-agent-fluent-bit • [warn] [engine] service will shutdown in max 5 seconds
Fluent Bit 엔진이 5초 내에 종료될 예정이라는 경고 메시지입니다. 이는 정상적인 서비스 종료 절차의 일부일 수 있습니다.
ops-agent-fluent-bit • [warn] [output:stackdriver:stackdriver.1] private_key is not defined, fetching it from metadata server
Stackdriver 출력 플러그인에서 private key가 정의되지 않아 메타데이터 서버에서 가져오려 시도하는 경고입니다. 설정 파일에 private key가 누락되었을 가능성이 있습니다.
ops-agent-fluent-bit • [warn] [output:stackdriver:stackdriver.1] client_email is not defined, using a default one
Stackdriver 출력 플러그인에서 client email이 정의되지 않아 기본값을 사용한다는 경고입니다. 인증 관련 설정이 누락되었을 가능성이 있습니다.
(위와 동일한 내용의 로그가 여러 개 확인되었습니다.)

3) 정보 (INFO) 로그

OSConfigAgent • Successfully completed ApplyConfigTask
OS 구성 에이전트가 구성을 성공적으로 적용했음을 나타냅니다.
OSConfigAgent • Policy \[policy name] resource \[resource name] state: COMPLIANT
특정 OS 정책의 리소스가 규정을 준수(COMPLIANT) 상태임을 나타냅니다.
GCEGuestAgent • Interface(ens4), State: {Index:2 MTU:1460 Name:ens4 HardwareAddr:42:01:0a:0a:0a:64 Flags:up|broadcast|multicast|running}, Addresses: \[10.10.10.100/32 fe80::4001:aff:fe0a:a64/64]
GCE 인스턴스의 네트워크 인터페이스(ens4) 상태 및 IP 주소 정보를 보여줍니다.
GCEGuestAgent • Currently present IP routes: ...
현재 인스턴스에 설정된 IP 라우팅 정보를 보여줍니다.
GCEGuestAgent • Getting current interface state and routes
네트워크 인터페이스 및 라우팅 상태를 가져오는 과정입니다.
GCEGuestAgent • Scheduler • added: \[...]
예약된 작업(job)이 스케줄러에 성공적으로 추가되었음을 나타냅니다.

결론 및 향후 방향

Cloud 환경에서는 네트워크, VM, 서버리스, SaaS 등 매우 다양한 서비스가 사용되고, 이에 따라 방대한 양의 로그가 생성됩니다.
2025년 기준으로 Google Cloud Platform(GCP)에서 공식적으로 제공하는 서비스는 100개를 넘으며, 세부 API와 리전별 리소스까지 포함하면 200개 이상의 항목에 달합니다.
이 모든 서비스에서 발생하는 로그를 수동으로 분석하고 일일이 대응하는 것은 현실적으로 쉽지 않습니다.

이처럼 방대한 서비스를 사용하는 과정에서 무수한 로그가 자연스럽게 발생하며, 운영진은 모니터링 솔루션과 알림 시스템을 통해 장애를 신속하게 탐지하고 대응해왔습니다.

하지만 매일 쏟아지는 방대한 로그 데이터를 모두 사람이 분석하고 개별 대응하는 것은 현실적으로 매우 어렵고, 반복적인 작업에 시간이 소모됩니다.

본 보고서에서 소개한 바와 같이, AI 기술과 오픈소스 자동화 도구인 n8n, Gemini, MCP Server를 활용하면, GCP 프로젝트 내 다양한 로그를 수집·분석하고, 이를 기반으로 자동 보고서 생성과 알림 발송까지 자동화할 수 있습니다.
이러한 자동화가 가능해짐에 따라 운영 효율성이 크게 향상되며, 잠재적인 문제를 조기에 발견하여 선제적으로 대응할 수 있다는 이점이 있습니다.

앞으로의 발전 방향으로 기대되는 것은 다음과 같습니다.

자동화 범위 확대: 장애 탐지뿐만 아니라 장애 예측, 보안 위협 식별, 리소스 최적화 등 보다 다양한 분야로 자동화를 확대
AI 활용 고도화: Gemini 등 고성능 AI 모델을 활용해 로그의 심층 분석, 이상징후 탐색, 상황별 맞춤형 대응 제안 기능을 강화합니다.
운영 정책 및 시스템 지속 개선: 자동화된 분석 결과를 바탕으로 운영 정책을 정기적으로 리뷰·개선하고, 시스템 구성과 보안 체계를 고도화

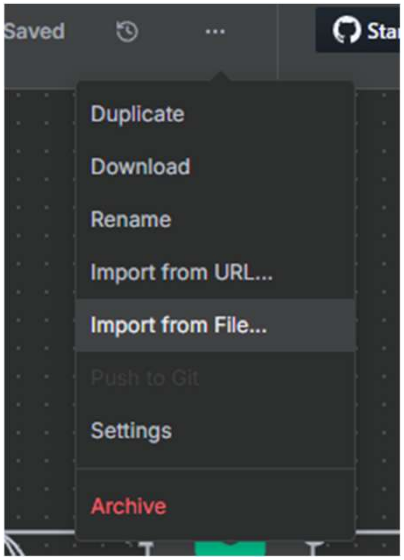
결론 및 향후 방향

n8n 템플릿 설정시 mcp server, chat model의 prompt등 많은 설정 내용이 있어서, 본 문서에서 가독성을 위해 제외하였습니다.

아래의 템플릿 파일을 다운로드 한 후 “Import from File” 선택하여 업로드 하면 설정내용을 모두 한번에 볼 수 있습니다.

본 문서의 내용을 바탕으로 GCP가 아닌 AWS, Azure MCP Server를 이용하여 해당 Cloud 로그도 분석이 가능합니다.

Templet 파일 다운로드



Thank you