

Google API Test

사전 설정

- Cloud Console에 새 프로젝트를 만듭니다.
- vision API를 검색하여 사용 설정합니다.
- 서비스 계정을 만듭니다.

Google Cloud Platform VisionTest

API 및 서비스 Cloud Vision API

사용자 인증 정보 + 사용자 인증 정보 만들기 삭제

이 API와 호환되는 사용자 인증 정보

모든 사용자 인증 정보를 보거나 새 사용자 인증 정보를 만들려면 [API 및 서비스의 사용자 인증 정보](#)로 이동하세요.

애플리케이션에 대한 정보를 포함하여 OAuth 동의 화면을 구성해야 합니다.

동의 화면 구성

OAuth 2.0 클라이언트 ID

<input type="checkbox"/>	이름	생성일 ↓	유형	클라이언트 ID
<input type="checkbox"/>	▲		서비스 계정 클라이언트	10742359161627596...

서비스 계정 [서비스 계정 관리](#)

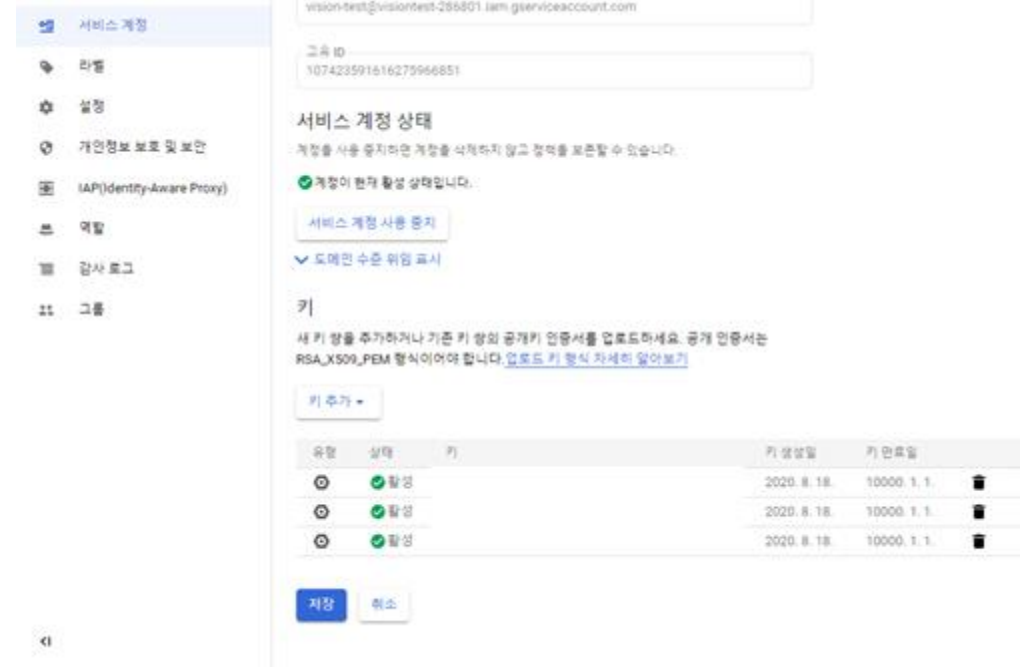
<input type="checkbox"/>	이메일	이름	이 서비스의 사용 현황(지난 30일) ↓	모든 서비스 사용량(지난 30일) ↓
<input type="checkbox"/>	vision-test@visiontest-286801.iam.gserviceaccount.com	vision_test	1	1

사전 설정

- 새 키를 만듭니다. (json파일을 다운 받습니다.)
- 해당 키가 저장된 경로를 복사해 둡니다.
- power shell에 들어갑니다
- `.$env:GOOGLE_APPLICATION_CREDENTIALS="[PATH]"`
에서 [path]대신 아까 복사해둔 경로/키 파일.json과 같이 입력합니다.

```
C:\Users\memen> $env:GOOGLE_APPLICATION_CREDENTIALS="C:\Users\memen\VisionTest-b7032df5ad1f.json"
```

- 그 후 <https://cloud.google.com/sdk/docs> 로 들어가 cloud sdk를 운영체제에 맞게 설치합니다.
- 이제 API를 쓰기 위한 기본 작업은 완료되었습니다.



이미지 내 텍스트 감지

- 먼저 요청 본문을 JSON 파일 형식으로 작성합니다.
- 예제는 우측과 같습니다.

```
{
  "requests": [
    {
      "image":
        {
          "content": "base64-encoded-image"
        },
      "features": [
        {
          "type": "TEXT_DETECTION"
        }
      ]
    }
  ]
}
```

- 입력 이미지는 base64 인코딩 형식으로 웹 사이트 (<https://www.base64-image.de/>) 혹은 코드를 동작시켜 추출하여야 합니다.
- 이 내용을 request.json으로 저장합니다
- 그 후 power shell에서 아래와 같이 입력합니다.(우측 텍스트를 복사해주세요)

```
PS C:\Users\memen> $env:GOOGLE_APPLICATION_CREDENTIALS="C:\Users\memen\VisionTest-b7032df5ad1f.json"
PS C:\Users\memen> $cred = gcloud auth application-default print-access-token
PS C:\Users\memen> $headers = @{ "Authorization" = "Bearer $cred" }
PS C:\Users\memen> Invoke-WebRequest `
>> -Method POST `
>> -Headers $headers `
>> -ContentType: "application/json; charset=utf-8" `
>> -InFile request.json `
>> -Uri "https://vision.googleapis.com/v1/images:annotate" | Select-Object -Expand Content
```

```
$cred = gcloud auth application-default print-access-token
$headers = @{ "Authorization" = "Bearer $cred" }
```

```
Invoke-WebRequest `
-Method POST `
-Headers $headers `
-ContentType: "application/json; charset=utf-8" `
-InFile request.json `
-Uri "https://vision.googleapis.com/v1/images:annotate" | Select-Object -Expand Content
```

참고

- VISION AI API 들은

```
PS C:\Users\memen> $env:GOOGLE_APPLICATION_CREDENTIALS="C:\Users\memen\VisionTest-b7032df5ad1f.json"
PS C:\Users\memen> $cred = gcloud auth application-default print-access-token
PS C:\Users\memen> $headers = @{ "Authorization" = "Bearer $cred" }
PS C:\Users\memen> Invoke-WebRequest
>> -Method POST
>> -Headers $headers
>> -ContentType: "application/json; charset=utf-8"
>> -InFile request.json
>> -Uri "https://vision.googleapis.com/v1/images:annotate" | Select-Object -Expand Content
```

powershell 명령어들은 바뀌지 않고
request.json 내용중 빨간 상자 부분들만 계속 바꾸어 주시면됩니다.

request.json 파일

```
{
  "requests": [
    {
      "image": {
        "content": "base64-encoded-image"
      },
      "features": [
        {
          "type": "TEXT_DETECTION"
        }
      ]
    }
  ]
}
```

테스트 이미지

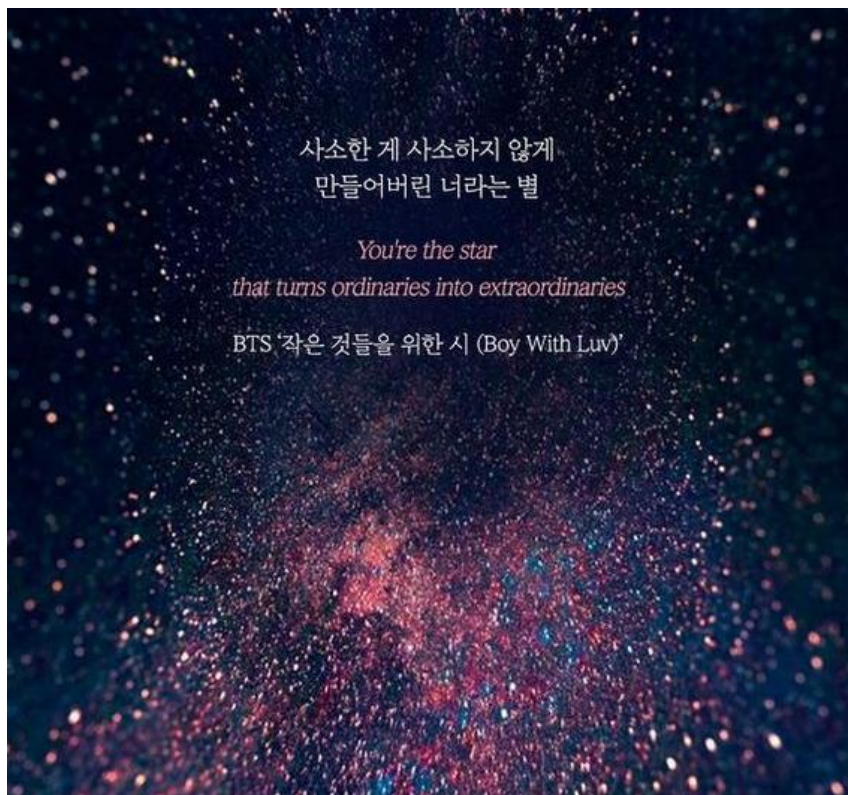


실행 결과

```
{
  "responses": [
    {
      "textAnnotations": [
        {
          "locale": "en",
          "description": "Wake up human!\n",
          "boundingPoly": {
            "vertices": [
              {
                "x": 30,
                "y": 396
              },
              {
                "x": 567,
                "y": 396
              },
              {
                "x": 567,
                "y": 465
              },
              {
                "x": 30,
                "y": 465
              }
            ]
          }
        },
        {
          "description": "Wake",
          "boundingPoly": {
            "vertices": [
              {
                "x": 30,
                "y": 396
              },
              {
                "x": 209,
                "y": 396
              },
              {
                "x": 209,
                "y": 457
              },
              {
                "x": 30,
                "y": 457
              }
            ]
          }
        },
        {
          "description": "up",
          "boundingPoly": {
            "vertices": [
              {
                "x": 230,
                "y": 406
              },
              {
                "x": 305,
                "y": 406
              },
              {
                "x": 305,
                "y": 463
              },
              {
                "x": 230,
                "y": 463
              }
            ]
          }
        },
        {
          "description": "human!",
          "boundingPoly": {
            "vertices": [
              {
                "x": 326,
                "y": 396
              },
              {
                "x": 567,
                "y": 396
              },
              {
                "x": 567,
                "y": 465
              },
              {
                "x": 326,
                "y": 465
              }
            ]
          }
        }
      ]
    }
  ],
  .....
}
```

테스트 이미지2

이번엔 이중 자막과 같이 2개의 언어가 한 이미지에 있을 경우 어떻게 처리하는지 확인해 봅시다.



테스트한 이미지는 좌측이고, 이를 실행한 결과 중 일부가 하단입니다.

한국어, 영어, 특수문자 `와 () 모두 이상 없이 인식하였습니다.

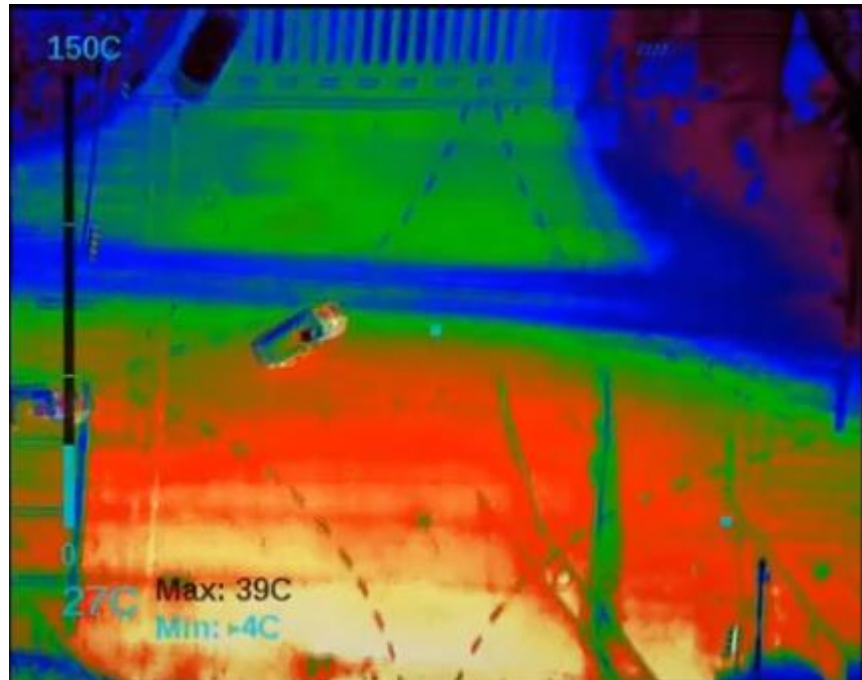
```

    "text": "사소한 게 사소하지 않게.ㄴ만들어버린 너라는 별ㄴYou're the starㄴthat turns ordinaries into extraordinariesㄴBTS '작은 것들을 위한 시 (Boy With Luv)'ㄴㄴ"

```


테스트 이미지3

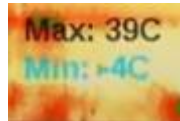
열화상 이미지에서 온도 추출이 가능할까?



```
"text": "150C\n1111\n27 Max: 39C\nMin: -4C"
```

이 이미지는 예상대로
Max : 39
Min : -4
가 그대로 나왔다.

하지만 문자가 아닌 중앙 상단 횡단보도 이미지를 숫자 1111로 인식하는 문제가 있다.

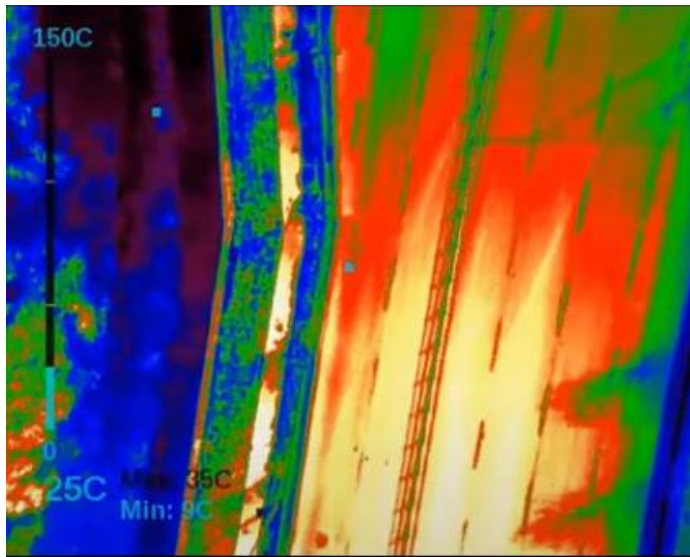


```
"text": "Max: 39C\nMax: WnMin: -4C"
```

온도 부분만 잘라서 인식 해도 Max 글자 인식 가능

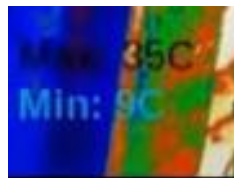
테스트 이미지4

열화상 이미지에서 온도 추출이 가능할까?



```
"text": "150C\n25C\n35C\nMin: 9C"
```

당연히 최고 온도 값도 Max : 35 이렇게 나올 줄 알았지만
그 주변의 온도가 낮아 어둡게 나와 인식되지 않은 듯 하다.



```
"text": "35C\nMin: 9C"
```

온도 부분만 잘라서 인식 해도 Max 글자 인식 안됨