




Runpod과 Llama-Factory를 활용하여 Fine-Tuning 빠르게 수행하기

파이프-튜닝



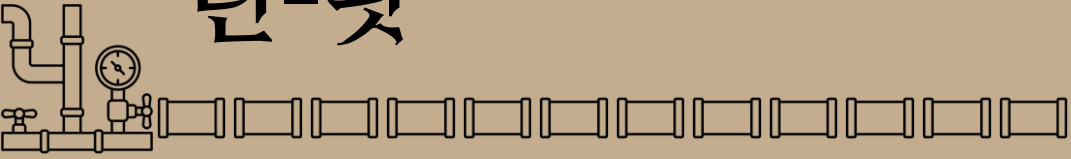
- LLM(Large Language Model) : 대규모 언어 모델
- 방대한 양의 텍스트 데이터를 학습하여, 인간과 유사한 방식으로 자연어를 이해하고 생성하는 인공지능 모델
- 우리는 LLM을
 - 1. 파인 튜닝(Fine-Tuning) 하거나
 - 2. RAG(Retrieval Augmented Generation) 하여 목적에 맞게 사용할 수 있다.
- 장점 : 특정한 도메인에 대한 성능 향상, 특정 작업에 대해 정확하게, 일관된 결과를 도출 가능
- 단점 : 새 도메인에 대한 재 학습 필요, 원본 모델의 일반화 성능 저하



라마팩토리

- LLM(Large Language Model)을 파인 튜닝하기 위한 오픈소스 프레임워크
- 약 100개의 사전 학습된 모델을 제공함
- 특징
 - 1. CLI, Web UI를 모두 제공하는 사용자 친화적 인터페이스
 - 2. LoRA, QLoRA 등 파인 튜닝 효율화 기법 내장
 - 3. 다양한 데이터셋 포맷을 지원(사용자 정의 데이터셋 통합 가능)
 - 4. 모델 내보내기 및 서비스화 지원

런-팻



- 클라우드 GPU 서비스로, 시간당 비용 지불 방식의 합리적인 활용 가능



런-팻 디플로이

CPU와 GPU 사용 방식에 대한 세팅 및 글로벌 네트워크 설정

GPU

CPU

Secure Cloud

☒ Global Networking

Network Volume

Any

Additional Filters

Select an Instance

Filter GPUs by VRAM

Any16244880140192288376432560658720900987112812601536

NVIDIA

Latest Gen

H100 SXM

\$2.99/hr
2.49/hr

80 GB VRAM
125 GB RAM • 16 vCPU

8 max
High

RTX 4090

\$0.69/hr
0.59/hr

24 GB VRAM
41 GB RAM • 6 vCPU

8 max
High

RTX 4000 Ada

\$0.38/hr
0.29/hr

20 GB VRAM
47 GB RAM • 9 vCPU

8 max
High

L40S

\$0.86/hr
0.73/hr

48 GB VRAM
62 GB RAM • 16 vCPU

7 max
Medium

RTX 2000 Ada

\$0.28/hr
0.21/hr

16 GB VRAM
31 GB RAM • 6 vCPU

6 max
Medium

B200

\$7.99/hr
6.99/hr

180 GB VRAM
283 GB RAM • 36 vCPU

5 max
Low

H200 SXM

\$3.99/hr

141 GB VRAM
251 GB RAM • 20 vCPU

8 max
Low

H100 NVL

\$2.79/hr
2.49/hr

94 GB VRAM
94 GB RAM • 16 vCPU

6 max
Low

L40

\$0.99/hr
0.84/hr

48 GB VRAM
94 GB RAM • 8 vCPU

7 max
Low

RTX 6000 Ada

\$0.77/hr
0.65/hr

48 GB VRAM
62 GB RAM • 16 vCPU

7 max
Low

RTX 5090

\$0.89/hr

32 GB VRAM
93 GB RAM • 16 vCPU

4 max
Low

L4

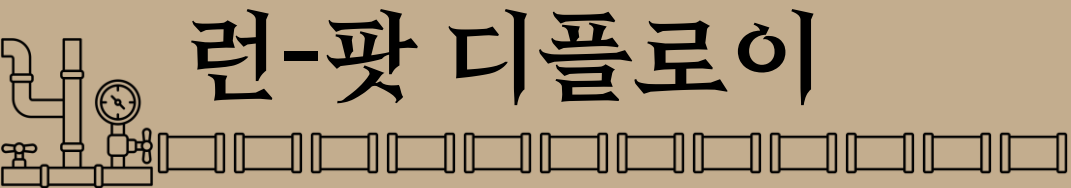
\$0.43/hr
0.37/hr

GPU는 원하는 종류 선택 가능

- 시간별 금액

- 가상메모리 용량

- 사용 가능 수준(High-Low)



런-팟 디플로이

Configure Deployment

Pod Name *

pod-1

Pod Template

정해진 규격의 Template 중에서 변경할 수 있습니다.



RunPod Pytorch 2.1

runpod/pytorch:2.1.0-py3.10-cuda11.8.0-devel-ubuntu22.04



Pod의 템플릿을 유지하면서 일부 설정을 변경할 수 있습니다.
드라이브(저장 장소) 용량, 특히 데이터가 크면 Volume Disk를 크게 잡는 것이 유리합니다.

Edit Template

Change Template

Pod Template Overrides

Container Image

runpod/pytorch:2.1.0-py3.10-cuda11.8.0-devel-ubuntu22.04

Container Start Command

This overrides the CMD in the Docker container

Container Disk

Temporary disk space for the container

- 20 GB +

Volume Disk

Persistent disk space mounted to the container

- 20 GB +

Volume Mount Path

/workspace

Expose HTTP Ports (Max 10)

8888

Expose TCP Ports

22



RunPod Pytorch 2.1.1

runpod/pytorch:2.1.1-py3.10-cuda12.1.1-devel-ubuntu22.04



RunPod Pytorch 2.8.0

runpod/pytorch:2.8.0-py3.11-cuda12.8.1-cudnn-devel-ubuntu22.04



RunPod Pytorch 1.13.1

runpod/pytorch:1.13.0-py3.10-cuda11.7.1-devel-ubuntu22.04



RunPod Desktop

runpod/kasm-docker:cuda11



RunPod VS Code Server

runpod/vscode-server:0.0.0



RunPod Disco Diffusion

runpod/discoart:web



RunPod SD InvokeAI v3.3.0

runpod/stable-diffusion:invoke-3.3.0



RunPod Kobold AI United

koboldai/koboldai:united



RunPod Stable Diffusion

runpod/stable-diffusion:web-ui-10.2.1



RunPod Tensorflow

runpod/tensorflow:1.0.3

How To?

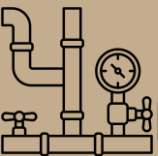


- 훈련(train)
 - LoRA adaptor의 가중치만 훈련됨
 - 원본 모델은 그대로 두고, LoRA의 low-rank parameter만 학습함
- 병합(merge)
 - 학습된 LoRA 가중치와 원래 모델을 하나로 병합
 - 이 단계를 거쳐야 하나의 완성된 모델이 됨
- 추론(predict)



Train 세팅

- 훈련(train)
 - 데이터 전처리
 - 툴 콜 포맷 구성
 - 템플릿 세팅
 - 학습 CLI 명령 적용



Train 세팅 1. 환경 구성

```
git clone --depth 1 https://github.com/hiyouga/LLaMA-Factory.git
```

#LLaMA-Factory 폴더 위치로 이동

```
%cd LLaMA-Factory
```

```
pip install -e “[torch,metrics]”
```

#허깅페이스 로그인

```
from huggingface_hub import notebook_login
```

```
notebook_login()
```



Train 세팅 2. 데이터셋 준비

```
from datasets import load_dataset  
ds = load_dataset("b-mc2/sql-create-context")
```

```
DatasetDict(  
  train: Dataset(  
    features: ['answer', 'question', 'context'],  
    num_rows: 78577  
  )  
)
```

```
print(ds['train']['question'][0])  
print(ds['train']['answer'][0])  
print(ds['train']['context'][0])
```

```
How many heads of the departments are older than 56 ?  
SELECT COUNT(*) FROM head WHERE age > 56  
CREATE TABLE head (age INTEGER)
```



```
formatted_data = []
```

```
for context, sql, question in W
```

```
formatted_data.append({'instruction': '사용자의 질문: ' + question +
```

```
'input': "
```

```
'output' : '결과 : ' + sql })
```

Train 세팅 2. 데이터셋 준비

LLaMA-Factory / data / alpaca_en_demo.json

hiyouga use pre-commit

Code Blame 5002 lines (5002 loc) · 841 KB Code 55% faster with GitHub Copilot

```
1  [
2    {
3      "instruction": "Describe a process of making crepes.",
4      "input": "",
5      "output": "Making crepes is an easy and delicious process! Here are step-by-step instructions: 1. Wash the crepe flour, salt, and eggs in a bowl. 2. Add milk and water to the bowl. 3. Mix the ingredients with a whisk. 4. Pour the batter into a crepe pan. 5. Cook the crepe for 1-2 minutes. 6. Roll the crepe up and serve it with your favorite filling."
6    },
7    {
8      "instruction": "Transform the following sentence using a synonym: The car sped quickly.",
9      "input": "",
10     "output": "The car accelerated rapidly."
11  }
```

- Instruction
- Input
- Output

LLaMA-Factory / data / glaive_toolcall_en_demo.json

hiyouga use pre-commit

Code Blame 9158 lines (9158 loc) · 722 KB Code 55% faster with GitHub Copilot

```
1  [
2    {
3      "conversations": [
4        {
5          "from": "human",
6          "value": "Hi, I have some ingredients and I want to cook something. Can you help me find a recipe?"
7        },
8        {
9          "from": "gpt",
10         "value": "Of course! I can help you with that. Please tell me what ingredients you have."
11       }
12     ]
13   }
```

- Conversations
 - From (human / gpt)
 - Value : 실제 내용



Train 세팅 2. 데이터셋 준비

```
converted = []
for item in data:
    instruction = item.get("instruction", "").strip()
    input_text = item.get("input", "").strip()
    output = item.get("output", "").strip()

    # output에서 "쿼리 결과:" 제거
    if output.startswith("쿼리 결과:"):
        output = output.replace("쿼리 결과:", "").strip()

    # human 발화 구성
    human_text = instruction
    if input_text:
        human_text += "\n\n" + input_text


    chat = { "conversations": [ {"from": "human", "value": human_text}, {"from": "gpt", "value": output}],
            "tools": TOOLS_INFO }
    converted.append(chat)
```

Train 세팅 3. 툴 콜

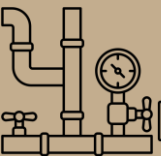
툴 정보를 여기에 미리 고정 (string 형태로 넣을지 실제 JSON으로 넣을지는 선택 가능)

```
TOOLS_INFO = json.dumps([
    {
        "name": "generate_sql",
        "description": "사용자의 질문과 테이블 정의(DDL)를 바탕으로 SQL 쿼리를 생성합니다.",
        "parameters": {
            "type": "object",
            "properties": {
                "instruction": {
                    "type": "string",
                    "description": "사용자의 자연어 질문 (예: '부서장이 56세 이상인 사람이 몇 명인가요?' 그리고, ddl_statements는 SQL의 테이블 구조가  
어떻게 형성되었는지에 대한 힌트입니다.)"
                },
                "ddl": {
                    "type": "string",
                    "description": "해당 테이블의 DDL 구문 (예: 'CREATE TABLE head (age INTEGER)')"
                }
            },
            "required": ["instruction", "ddl"]
        }
    }
], ensure_ascii=False)
```

Train 세팅 4. 템플릿 세팅



```
args = dict(
    stage="sft", # 지도 학습 방식의 파인튜닝(Supervised Fine-Tuning)을 수행합니다.
    do_train=True, # 학습을 실행할지 여부를 설정합니다. True일 경우 학습이 시작됩니다.
    model_name_or_path="allganize/Llama-3-Alpha-Ko-8B-Instruct", # bnb-4bit 양자화된 LLaMA-3-8B-Instruct 모델
    # dataset_dir="/workspace/LLaMA-Factory/data/output_llama3.json",
    # formatting="sharegpt", # (선택사항) 데이터 포맷이 ShareGPT 스타일일 경우 지정
    dataset="llama3_custom_sql", # 사용할 커스텀 데이터셋의 이름입니다 (dataset_info.json에 등록된 이름)
    template="llama3", # LLaMA-3 스타일의 채팅 프롬프트 템플릿을 사용합니다.
    finetuning_type="lora", # LoRA 방식으로 파인튜닝하여 메모리를 절약합니다.
    lora_target="all", # 모든 선형 계층에 LoRA 어댑터를 적용합니다.
    output_dir="checkpoint", # 학습 결과(LoRA 어댑터 등)가 저장될 디렉토리입니다.
    per_device_train_batch_size=2, # GPU 하나당 학습 배치 크기를 2로 설정합니다.
    gradient_accumulation_steps=4, # 그래디언트 누적 횟수를 4로 설정하여 유효 배치 크기를 늘립니다.
    lr_scheduler_type="cosine", # 코사인 스케줄러를 사용해 학습률을 조절합니다.
    logging_steps=10, # 학습 상태를 10 스텝마다 로그로 출력합니다.
    warmup_ratio=0.1, # 전체 학습의 10% 동안 학습률을 점진적으로 증가시킵니다 (워밍업 단계).
    save_steps=1000, # 1000 스텝마다 체크포인트를 저장합니다.
    learning_rate=5e-5, # 기본 학습률을 5e-5로 설정합니다.
    num_train_epochs=5.0, # 전체 학습을 5 에폭 동안 수행합니다.
    max_samples=500, # 각 데이터셋에서 최대 500개의 샘플만 사용하여 빠른 실험이 가능하게 합니다.
    max_grad_norm=1.0, # 그래디언트 클리핑 최대값을 1.0으로 설정하여 안정적인 학습을 도모합니다.
    quantization_bit=4, # QLoRA 방식의 4비트 양자화를 적용하여 메모리 사용량을 줄입니다.
    loraplus_lr_ratio=16.0, # LoRA+ 알고리즘 사용 시 learning rate scaling factor를 16으로 설정합니다.
    fp16=True, # float16(반정밀도) 혼합 정밀도 학습을 사용하여 속도와 메모리 효율을 높입니다.
)
json.dump(args, open("train_llama3.json", "w", encoding="utf-8"), indent=2)
```

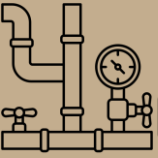


Train 세팅 5. 학습 CLI 명령 적용

```
!llamafactory-cli train ./train_llama3.json
```

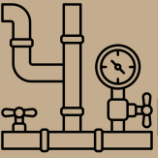
```
[INFO|tokenization_utils_base.py:2510] 2025-04-23 03:05:52,446 >> tokenizer config file saved in checkpoint/tokenizer_config.json
[INFO|tokenization_utils_base.py:2519] 2025-04-23 03:05:52,450 >> Special tokens file saved in checkpoint/special_tokens_map.json
***** train metrics *****
epoch                =      4.928
total_flos           = 11326379GF
train_loss           =      0.0661
train_runtime        = 0:07:31.45
train_samples_per_second =      5.538
train_steps_per_second  =      0.687
[INFO|modelcard.py:450] 2025-04-23 03:05:52,670 >> Dropping the following result as it does not have all the necessary fields:
{'task': {'name': 'Causal Language Modeling', 'type': 'text-generation'}}
```


병합 세팅



```
!llamafactory-cli export  
--model_name_or_path allganize/Llama-3-Alpha-Ko-8B-Instruct  
--adapter_name_or_path /workspace/LLaMA-Factory/checkpoint  
--export_dir merged_model  
--finetuning_type lora
```

추론 세팅



```
from transformers import AutoModelForCausalLM, AutoTokenizer
```

```
model = AutoModelForCausalLM.from_pretrained("/workspace/merged_model",  
torch_dtype="auto")
```

```
tokenizer = AutoTokenizer.from_pretrained("/workspace/merged_model")
```

```
prompt = "부서장이 56세 이상인 사람이 몇 명인가요? DDL: CREATE TABLE head (age INTEGER)"
```

```
inputs = tokenizer(prompt, return_tensors="pt").to(model.device)
```

```
outputs = model.generate(**inputs, max_new_tokens=100)
```

```
print(tokenizer.decode(outputs[0], skip_special_tokens=True))
```

추론 세팅



```
prompt = "출생연도가 2010년대 이후인 아기들의 이름을 알려줘 DDL : CREATE TABLE person (birth DATE, name STR)"
inputs = tokenizer(prompt, return_tensors="pt").to(model.device)
outputs = model.generate(**inputs, max_new_tokens=100)
print(tokenizer.decode(outputs[0], skip_special_tokens=True))
```

Setting `pad_token_id` to `eos_token_id`:128001 for open-end generation.

```
출생연도가 2010년대 이후인 아기들의 이름을 알려줘 DDL : CREATE TABLE person (birth DATE, name STR) SELECT name FROM person WHERE birth >
"2010-01-01"
```

Heroku와 Git으로 FastAPI 백엔드 배포하기





0. 파일 준비

Procfile (확장자 없음) / 헤로쿠 서버가 실행되었을 때 어떤 명령어가 자동으로 실행될지에 대한 내용을 담음

```
1 web: uvicorn fastapp:app --host=0.0.0.0 --port=$PORT --workers=1
2 |
```

Aptfile (확장자 없음) / 헤로쿠 서버의 초기 설치에 필요한 디펜던시 목록을 담음

```
mysql-common
mariadb-common
...
ffmpeg
```

APT 파일 작성



1. 헤로쿠 배포 준비

- 헤로쿠
 - 헤로쿠 회원가입
 - 헤로쿠 CLI 설치(로그를 보기 편함)
 - 헤로쿠 웹페이지에서 앱을 설치



1. 헤로쿠 배포 준비

<https://devcenter.heroku.com/articles/getting-started-with-python#set-up>

Download and run the installer for your platform:



Install Homebrew and run:

```
$ brew tap heroku/brew && brew install heroku
```



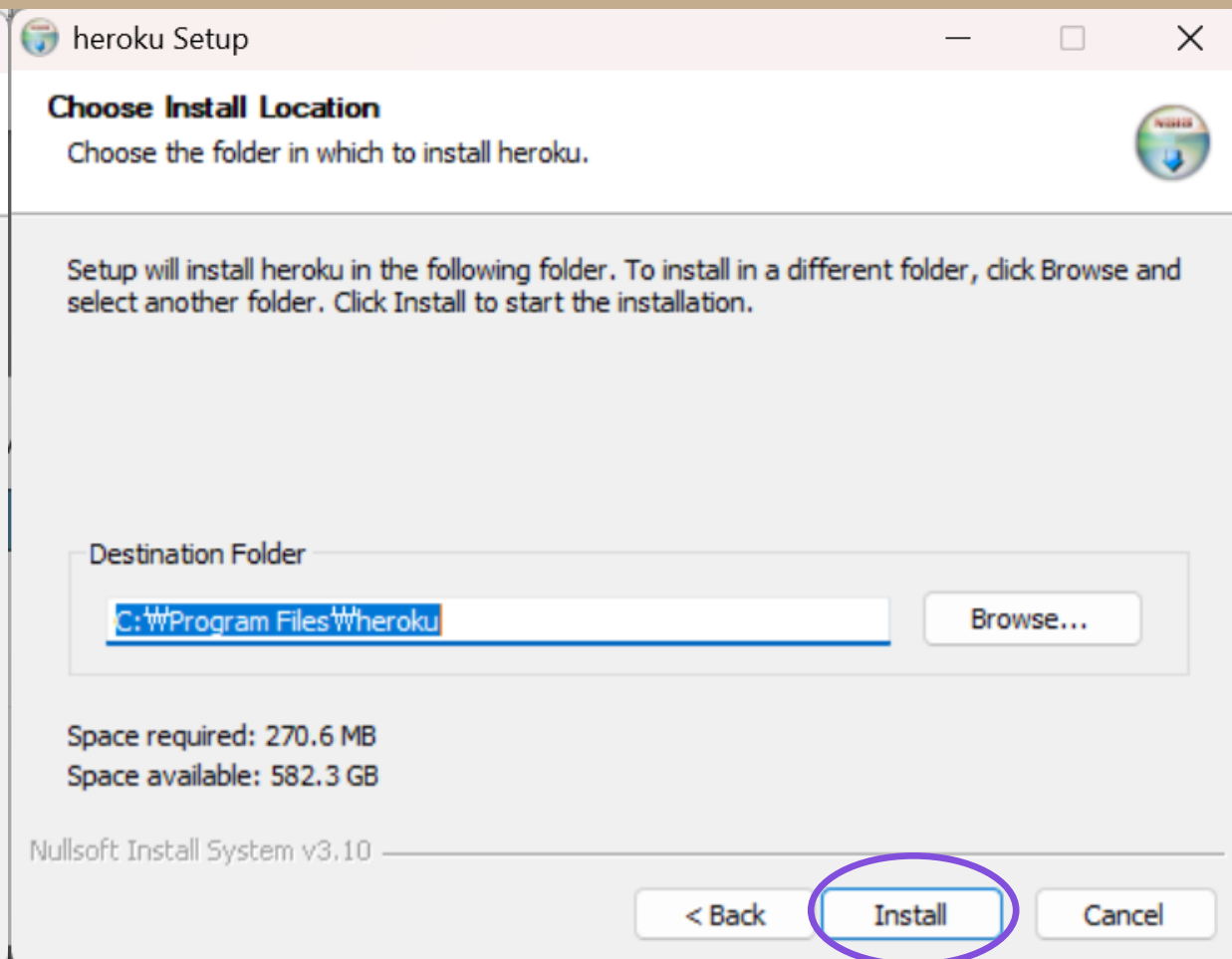
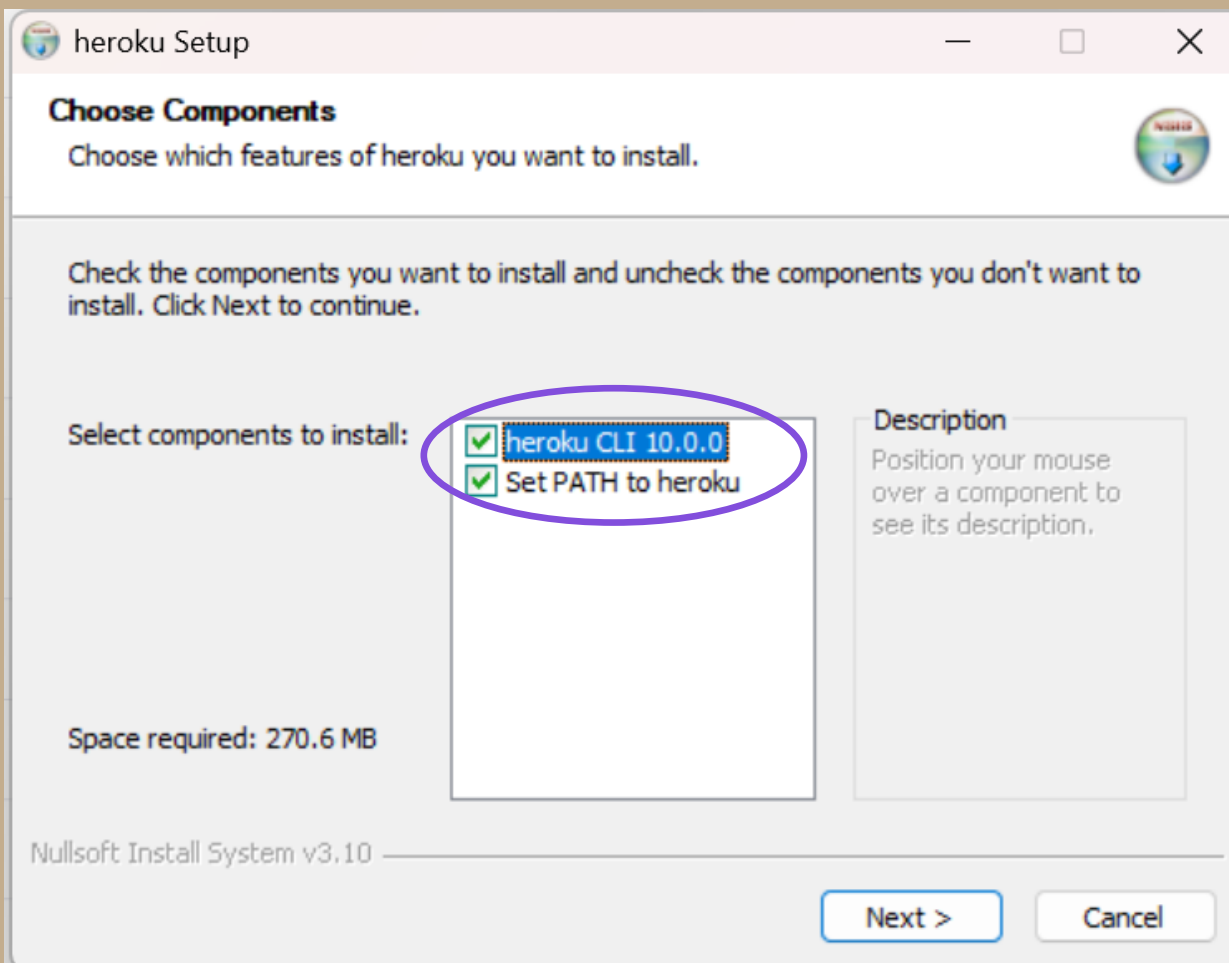
Download the appropriate installer
for your Windows installation:

64-bit installer

32-bit installer



1. 헤로쿠 배포 준비




2. 깃 레포 준비

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk (*).

Owner *

 agent47ag925 ▾

Repository name *

/ ai_app

✔ ai_app is available.

Great repository names are short and memorable. Need inspiration? How about [bookish-parakeet](#) ?

Description (optional)



Public

Anyone on the internet can see this repository. You choose who can commit.



Private

You choose who can see and commit to this repository.

Initialize this repository with:



Add a README file

This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

.gitignore template: None ▾

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

License: None ▾

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)



You are creating a public repository in your personal account.

Create repository

3. 헤로쿠 앱 만들기



Jump to Favorites, Apps, Pipelines, Spaces...



The Next Generation Platform is Coming - [Get Ready Now!](#)

Personal ▾

New ▾

Create your first app!

Apps and pipelines you create or collaborate on appear here.

Create New App

Looking for help getting started with your language?

Get started by reading one of our language guides in the Dev Center



Ruby



PHP



Node.js



Python



Java



Go



Clojure



Scala

3. 헤로쿠 앱 만들기

App name

Give this app a globally unique name. For example, acme-production-app.

Location

Choose a Common Runtime region for this app. [Learn more.](#)

Common Runtime
CEDAR



United States

Common Runtime
CEDAR



Europe



Add this app to a pipeline

Create a new application as a personal app.

Cancel

Create app

3. 헤로쿠 앱 만들기

Deployment method



Heroku Git
Use Heroku CLI



GitHub
Connect to GitHub



Container Registry
Use Heroku CLI

Connect to GitHub

Connect this app to GitHub to enable code diffs and deploys.

Search for a repository to connect to



agent47ag925



ai_app

Search

Missing a GitHub organization? [Ensure Heroku Dashboard has team access.](#)



agent47ag925/ai_app

Connect

2일째 Github에서 만든 Repo 검색

3. 헤로쿠 앱 만들기

Automatic deploys

Enables a chosen branch to be automatically deployed to this app.

You can now change your main deploy branch from "master" to "main" for both manual and automatic deploys, please follow the instructions [here](#).

Enable automatic deploys from GitHub

Every push to the branch you specify here will deploy a new version of this app. **Deploys happen automatically**; be sure that this branch is always in a deployable state and any tests have passed before you push. [Learn more](#).

Choose a branch to deploy

master

☐ Wait for GitHub checks to pass before deploy

Only enable this option if you have a Continuous Integration service configured on your repo.

Enable Automatic Deploys

Manual deploy

Deploy the current state of a branch to this app.

Deploy a GitHub branch

This will deploy the current state of the branch you specify below. [Learn more](#).

Choose a branch to deploy

master

Personal > lang-aiapp

GitHub agent47ag925/ai_app master

Overview

Resources

Deploy

Metrics

Activity

Access

Settings

Push할 때 마다 자동으로 배포되게 함

Automatic deploys

Enables a chosen branch to be automatically deployed to this app.

You can now change your main deploy branch from "master" to "main" for both manual and automatic deploys, please follow the instructions [here](#).

Automatic deploys from master are enabled

Every push to master will deploy a new version of this app. **Deploys happen automatically**; be sure that this branch in GitHub is always in a deployable state and any tests have passed before you push. [Learn more](#).

☐ Wait for GitHub checks to pass before deploy

Only enable this option if you have a Continuous Integration service configured on your repo.

Disable Automatic Deploys

3. 헤로쿠 앱 만들기

Config Vars

Config vars change the way your app behaves. In addition to creating your own, some add-ons come with their own.

Config Vars

Hide Config Vars

There are no config vars for this app yet
[Learn about config vars](#) in the Dev Center.

KEY

VALUE

Add

Config Vars

Config vars change the way your app behaves. In addition to creating your own, some add-ons come with their own.

Config Vars

숨겨야 하는 비밀 키들은 Config Vars에 추가

Hide Config Vars

API_KEY

sk-proj-U0a-JLTMTxf-MwNtP9AJu5bEk_x8o0IRT



KEY

VALUE

Add

4. 헤로쿠 CLI로 배포 완성

- `heroku login` #터미널에서 헤로쿠 로그인 수행
- `heroku apps` #내 헤로쿠 계정이 가지고 있는 앱들이 나타남
- `heroku git:remote -a ai-app` #깃 폴더 위치로 가서 수행
- `heroku buildpacks:add --index 1 heroku-community/apt`
- `heroku buildpacks:add --index 2 heroku/python`
- `heroku buildpacks` #빌드팩의 목록 확인
- `heroku buildpacks:remove heroku/python` #특정한 이름의 빌드팩 지우기



4. 헤로쿠 CLI로 배포 완성

```
C:\Users\Wjeong\Desktop\특강 준비\Heroku Back>heroku logs
> Warning: heroku update available from 10.1.0 to 10.6.1.
2025-04-23T08:06:29.893522+00:00 app[api]: Release v1 created by user 47ag925park@gmail.com
2025-04-23T08:06:29.893522+00:00 app[api]: Initial release by user 47ag925park@gmail.com
2025-04-23T08:06:30.098321+00:00 app[api]: Enable Logplex by user 47ag925park@gmail.com
2025-04-23T08:06:30.098321+00:00 app[api]: Release v2 created by user 47ag925park@gmail.com
2025-04-23T08:16:05.000000+00:00 app[api]: Build started by user 47ag925park@gmail.com
2025-04-23T08:16:39.000000+00:00 app[api]: Build failed -- check your build output: https://dashboard.heroku.com/apps/d0a14752-c512-416c-9cc8-4a8232b00c2c/activity/builds/8ea93771-e9e7-4fda-be7d-df33405e5809
```

- heroku ps #정상 배포 확인 명령어

```
C:\Users\Wjeong\Desktop\OpenAISupport\Project>heroku ps
=== web (Basic): streamlit run main.py --server.port=$PORT --server.address=0.0.0.0 (1)

web.1: up 2025/02/16 11:36:29 +0900 (~ 1h ago)
```

- heroku logs --tail #설치 및 동작 로그 확인

```
C:\Users\Wjeong\Desktop\OpenAISupport\Project>heroku logs --tail
```

- heroku ps:scale web=1 #설치는 잘 되었는데 실행이 안되었다면 아래와 같이 수행

```
C:\Users\Wjeong\Desktop\OpenAISupport\Project>heroku ps:scale web=1_
```


4. 헤로쿠 CLI로 배포 완성

에러 확인 후, 각 상황에 맞는 대응이 필요함

```
C:\Users\Wjeong\Desktop\특강 준비\HerokuBack>heroku logs
>> Warning: heroku update available from 10.1.0 to 10.6.1.
2025-04-23T08:06:29.893522+00:00 app[api]: Release v1 created by user 47ag925park@gmail.com
2025-04-23T08:06:29.893522+00:00 app[api]: Initial release by user 47ag925park@gmail.com
2025-04-23T08:06:30.098321+00:00 app[api]: Enable Logplex by user 47ag925park@gmail.com
2025-04-23T08:06:30.098321+00:00 app[api]: Release v2 created by user 47ag925park@gmail.com
2025-04-23T08:16:05.000000+00:00 app[api]: Build started by user 47ag925park@gmail.com
2025-04-23T08:16:39.000000+00:00 app[api]: Build failed -- check your build output: https://dashboard.heroku.com/apps/d0a14752-c512-4f6c-9cc8-4a8232b00c2c/activity/builds/8ea93771-e9e7-4fda-be7d-df33405e5809
```