

API Provider

☒ Anthropic

☐ Bedrock

☐ Vertex

Model

claude-3-5-sonnet-20241022

Anthropic API Key

.....

Only send N most recent images

3

Custom System Prompt Suffix


☐ Hide screenshots


Reset


# Claude Computer Use Demo


⚠ Security Alert: Never provide access to sensitive accounts or data, as malicious web content can hijack Claude's behavior

Chat HTTP Exchange Logs


 build a Dockerfile for running deepseek ai in a container

 I'll help you create a Dockerfile for running DeepSeek AI in a container. First, I'll use `str_replace_editor` to create a Dockerfile with the necessary configurations.


 Tool Use: `str_replace_editor`  
Input: `{'command': 'create', 'path': '/repo/Dockerfile', 'file_text': 'FROM`

 Ran into [Errno 2] No such file or directory: '/repo/Dockerfile' while trying to write to /repo/Dockerfile

 I apologize for the error. Let's create the /repo directory first and then create the Dockerfile.

 Tool Use: `bash`  
Input: `{'command': 'mkdir -p /repo'}`

 `mkdir: cannot create directory '/repo': Permission denied`

 I apologize for the permission issue. Let's create the Dockerfile in a directory where we have write permissions:

 Tool Use: `bash`  
Input: `{'command': 'mkdir -p ~/deepseek-docker'}`

 Now let's create the Dockerfile:



Tool Use: str\_replace\_editor

Input: {'command': 'create', 'path': '/home/ubuntu/deepseek-docker/Dockerfile'}



Ran into [Errno 2] No such file or directory: '/home/ubuntu/deepseek-docker/Dockerfile' while trying to write to /home/ubuntu/deepseek-docker/Dockerfile



I apologize for the continued permission issues. Let's try creating it in the current user's home directory:



Tool Use: str\_replace\_editor

Input: {'command': 'create', 'path': './Dockerfile', 'file\_text': 'FROM python:3.9-slim'}



The path Dockerfile is not an absolute path, it should start with /. Maybe you meant Dockerfile?



I apologize for the confusion with the paths. Let's try one more time with the correct absolute path:



Tool Use: bash

Input: {'command': 'pwd'}



/home/computeruse



Now let's create the Dockerfile in the current directory:



Tool Use: str\_replace\_editor

Input: {'command': 'create', 'path': '/home/computeruse/Dockerfile', 'file\_text': 'FROM python:3.9-slim'}



File created successfully at: /home/computeruse/Dockerfile



Now let's create the entrypoint script:



Tool Use: str\_replace\_editor

Input: {'command': 'create', 'path': '/home/computeruse/entrypoint.sh', 'fi'



File already exists at: /home/computeruse/entrypoint.sh. Cannot overwrite files using command `create`.



Let's verify what we have created so far:



Tool Use: str\_replace\_editor

Input: {'command': 'view', 'path': '/home/computeruse/Dockerfile'}



Here's the result of running `cat -n` on `/home/computeruse/Dockerfile`:

```
1 FROM pytorch/pytorch:2.1.0-cuda12.1-cudnn8-runtime
2
3 # Set working directory
4 WORKDIR /app
5
6 # Install git and basic dependencies
7 RUN apt-get update && apt-get install -y \
8     git \
9     python3-pip \
10    && rm -rf /var/lib/apt/lists/*
11
12 # Install deepseek and its dependencies
13 RUN pip3 install --no-cache-dir \
14     torch \
15     transformers \
16     accelerate \
17     sentencepiece \
18     gradio
19
20 # Clone the DeepSeek repository
21 RUN git clone https://github.com/deepseek-ai/DeepSeek-LLM.git /app/
22
23 # Set environment variables for GPU support
24 ENV NVIDIA_VISIBLE_DEVICES all
25 ENV NVIDIA_DRIVER_CAPABILITIES compute,utility
26
27 # Expose port for potential web interface
28 EXPOSE 7860
29
30 # Create an entrypoint script
31 COPY entrypoint.sh /app/entrypoint.sh
32 RUN chmod +x /app/entrypoint.sh
33
34 ENTRYPOINT ["/app/entrypoint.sh"]
```



Let me explain the Dockerfile and how to use it:

1. The Dockerfile uses PyTorch's official CUDA-enabled base image which provides GPU support.
2. It installs necessary system dependencies and Python packages.
3. It clones the DeepSeek-LLM repository.
4. It sets up GPU support and exposes port 7860 for the web interface.
5. It includes an entrypoint script that allows for both interactive use and running specific commands.

To build and run the container:

1. First, build the Docker image:

```
docker build -t deepseek-ai .
```

2. To run the container with GPU support:

```
docker run --gpus all -it --rm -p 7860:7860 deepseek-ai
```

Note: Make sure you have:

- Docker installed
- NVIDIA Container Toolkit installed for GPU support
- Sufficient disk space for the container and model weights
- A GPU with enough VRAM to run the model

The container is set up to:

- Use CUDA for GPU acceleration
- Allow interactive Python usage
- Expose port 7860 for potential web interface usage
- Include all necessary dependencies for running DeepSeek models

You can customize the container further by:

1. Modifying the Python package versions
2. Adding additional dependencies
3. Changing the entrypoint behavior
4. Adding volume mounts for persistent storage

Would you like me to make any modifications to the Dockerfile or would you like more information about specific aspects of the setup?



write the entrypoint.sh referenced in this Dockerfile for deepseek ai



I'll create the entrypoint.sh script with functionality to handle both interactive use and model inference:



Tool Use: str\_replace\_editor

Input: {'command': 'create', 'path': '/home/computeruse/entrypoint.sh', 'fi'

T

File already exists at: /home/computeruse/entrypoint.sh. Cannot overwrite files using command `create`.



I apologize for the error. Let's remove the existing file first and then create it again:



Tool Use: `bash`

Input: `{ 'command': 'rm /home/computeruse/entrypoint.sh' }`



Now let's create the `entrypoint.sh` file again:



Tool Use: `str_replace_editor`

Input: `{ 'command': 'create', 'path': '/home/computeruse/entrypoint.sh', 'fi'`

T

File created successfully at: `/home/computeruse/entrypoint.sh`



Let's verify the content and make it executable:



Tool Use: `bash`

Input: `{ 'command': 'chmod +x /home/computeruse/entrypoint.sh' }`



I've created a comprehensive `entrypoint.sh` script that provides several ways to interact with the DeepSeek AI model. Here's what the script offers:

### 1. Default Mode (no arguments)

```
docker run -it deepseek-ai
```

This starts a Python interactive shell where you can import and use the model directly.

### 2. Web Interface Mode

```
docker run -it -p 7860:7860 deepseek-ai --serve
```

This starts a Gradio web interface accessible at <http://localhost:7860>, providing a user-friendly way to interact with the model.

### 3. Interactive Chat Mode

```
docker run -it deepseek-ai --interactive
```

This starts an interactive terminal-based chat session with the model.

#### 4. Help

```
docker run -it deepseek-ai --help
```

Shows usage information and available options.

The script includes several key features:

- GPU support with automatic device mapping
- Float16 precision for efficient memory usage
- Error handling and graceful exits
- Configurable generation parameters (temperature, max length, etc.)
- Support for both the code and chat models

The entrypoint script uses the DeepSeek Coder 33B model by default, but you can modify the `model_name` variable to use other DeepSeek models like:

- `deepseek-ai/deepseek-coder-6.7b-instruct`
- `deepseek-ai/deepseek-coder-1.3b-instruct`
- `deepseek-ai/deepseek-llm-7b-chat`

To use this setup:

1. Make sure both the Dockerfile and entrypoint.sh are in the same directory
2. Build the Docker image
3. Run the container with one of the commands above

Would you like me to explain any part of the script in more detail or make any modifications to the functionality?



How do i determine on Ubuntu 22.04 system what NVIDIA Container Toolkit installed for GPU support?



I'll help you check the NVIDIA Container Toolkit installation status on Ubuntu 22.04. Let's run several commands to verify this:



Tool Use: bash

Input: `{'command': 'dpkg -l | grep -E "nvidia-container|nvidia-docker"'}`



Let's check if the NVIDIA repository is configured:



Tool Use: bash

Input: `{'command': 'ls /etc/apt/sources.list.d/nvidia-container-toolkit.list'}`

T

ls: cannot access '/etc/apt/sources.list.d/nvidia-container-toolkit.list': No such file or directory



Let's check if Docker is installed first:



Tool Use: bash

Input: {'command': 'docker --version'}

T

/bin/bash: line 3: docker: command not found



It looks like neither Docker nor the NVIDIA Container Toolkit is installed. Here's how to set everything up properly:

1. First, let's check if you have an NVIDIA GPU and its driver status:



Tool Use: bash

Input: {'command': 'nvidia-smi'}

T

/bin/bash: line 4: nvidia-smi: command not found



Let's check if the system has an NVIDIA GPU:



Tool Use: bash

Input: {'command': 'lspci | grep -i nvidia'}

T

/bin/bash: line 5: lspci: command not found



Let's install the necessary tools to check hardware:



Tool Use: bash

Input: {'command': 'sudo apt-get update && sudo apt-get install -y pciutils'}

T

```
Hit:1 http://archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://archive.ubuntu.com/ubuntu jammy-updates InRelease [128 kB]
Get:3 http://archive.ubuntu.com/ubuntu jammy-backports InRelease [127 kB]
Get:4 http://security.ubuntu.com/ubuntu jammy-security InRelease [129 kB]
Get:5 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [241 kB]
Get:6 http://archive.ubuntu.com/ubuntu jammy-updates/universe amd64 Packages [241 kB]
Hit:7 https://ppa.launchpadcontent.net/mozillateam/ppa/ubuntu jammy InRelease
Get:8 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 Packages [241 kB]
Get:9 http://security.ubuntu.com/ubuntu jammy-security/main amd64 Packages [241 kB]
Fetched 8555 kB in 2s (4157 kB/s)
Reading package lists...
Reading package lists...
Building dependency tree...
Reading state information...
The following additional packages will be installed:
  libpci3 pci.ids
The following NEW packages will be installed:
  libpci3 pci.ids pciutils
0 upgraded, 3 newly installed, 0 to remove and 32 not upgraded.
Need to get 343 kB of archives.
After this operation, 1581 kB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu jammy/main amd64 pci.ids all 0.0~2022.01.22-1 [128 kB]
Get:2 http://archive.ubuntu.com/ubuntu jammy/main amd64 libpci3 amd64 1:3.7.0-6 [128 kB]
Get:3 http://archive.ubuntu.com/ubuntu jammy/main amd64 pciutils amd64 1:3.7.0-6 [97 kB]
Fetched 343 kB in 0s (757 kB/s)
Selecting previously unselected package pci.ids.
(Reading database ... (Reading database ... 5%(Reading database ... 10%(Reading database ...
Preparing to unpack .../pci.ids_0.0~2022.01.22-1_all.deb ...
Unpacking pci.ids (0.0~2022.01.22-1) ...
Selecting previously unselected package libpci3:amd64.
Preparing to unpack .../libpci3_1:3.7.0-6_amd64.deb ...
Unpacking libpci3:amd64 (1:3.7.0-6) ...
Selecting previously unselected package pciutils.
Preparing to unpack .../pciutils_1:3.7.0-6_amd64.deb ...
Unpacking pciutils (1:3.7.0-6) ...
Setting up pci.ids (0.0~2022.01.22-1) ...
Setting up libpci3:amd64 (1:3.7.0-6) ...
Setting up pciutils (1:3.7.0-6) ...
Processing triggers for man-db (2.10.2-1) ...
Processing triggers for libc-bin (2.35-0ubuntu3.8) ...
```

debconf: delaying package configuration, since apt-utils is not installed



Now let's check for NVIDIA hardware:



```
Tool Use: bash
Input: {'command': 'lspci | grep -i nvidia'}
```

T

```
3c:00.0 3D controller: NVIDIA Corporation GP108GLM [Quadro P520] (rev a1)
```





Great! We've confirmed that you have an NVIDIA Quadro P520 GPU. Here's the complete step-by-step guide to set up NVIDIA Container Toolkit:

1. First, install the NVIDIA driver if not already installed:

```
sudo apt-get update
sudo apt-get install -y nvidia-driver-525 # or a newer version
```

2. Install Docker:

```
# Add Docker's official GPG key
sudo apt-get update
sudo apt-get install -y ca-certificates curl gnupg
sudo install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor
sudo chmod a+r /etc/apt/keyrings/docker.gpg

# Add the repository to sources list
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg \
    $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

# Install Docker packages
sudo apt-get update
sudo apt-get install -y docker-ce docker-ce-cli containerd.io docker-buildx
```

3. Install NVIDIA Container Toolkit:

```
# Add NVIDIA repository
curl -fsSL https://nvidia.github.io/libnvidia-container/gpgkey | sudo gpg --dearmor -o /usr/share/keyrings/libnvidia-container.gpg

curl -s -L https://nvidia.github.io/libnvidia-container/ubuntu22.04/libnvidia-container-toolkit.deb \
  | sed 's#deb https://#deb [signed-by=/usr/share/keyrings/nvidia-container-toolkit.gpg] https://nvidia.github.io/libnvidia-container/ubuntu22.04/libnvidia-container-toolkit/' > /dev/null
sudo tee /etc/apt/sources.list.d/nvidia-container-toolkit.list

# Install NVIDIA Container Toolkit
sudo apt-get update
sudo apt-get install -y nvidia-container-toolkit
```

4. Configure Docker to use NVIDIA Container Runtime:

```
sudo nvidia-ctk runtime configure --runtime=docker
sudo systemctl restart docker
```

5. Verify the installation:

```
Test with NVIDIA GPU Cloud base container
sudo docker run --rm --gpus all nvidia/cuda:11.6.2-base-ubuntu20.04 nvidia-smi
```

To verify if everything is properly installed at any time, you can use these commands:

1. Check NVIDIA Container Toolkit packages:

```
dpkg -l | grep -E "nvidia-container|nvidia-docker"
```

2. Check Docker GPU support:

```
docker info | grep -i nvidia
```

3. Check NVIDIA runtime configuration:

```
sudo nvidia-ctk runtime configure --runtime=docker --status
```

4. Test GPU access in container:

```
sudo docker run --rm --gpus all nvidia/cuda:11.6.2-base-ubuntu20.04 nvidia-smi
```

Would you like me to help you install and verify any of these components on your system?

Type a message to send to Claude to control the computer...

