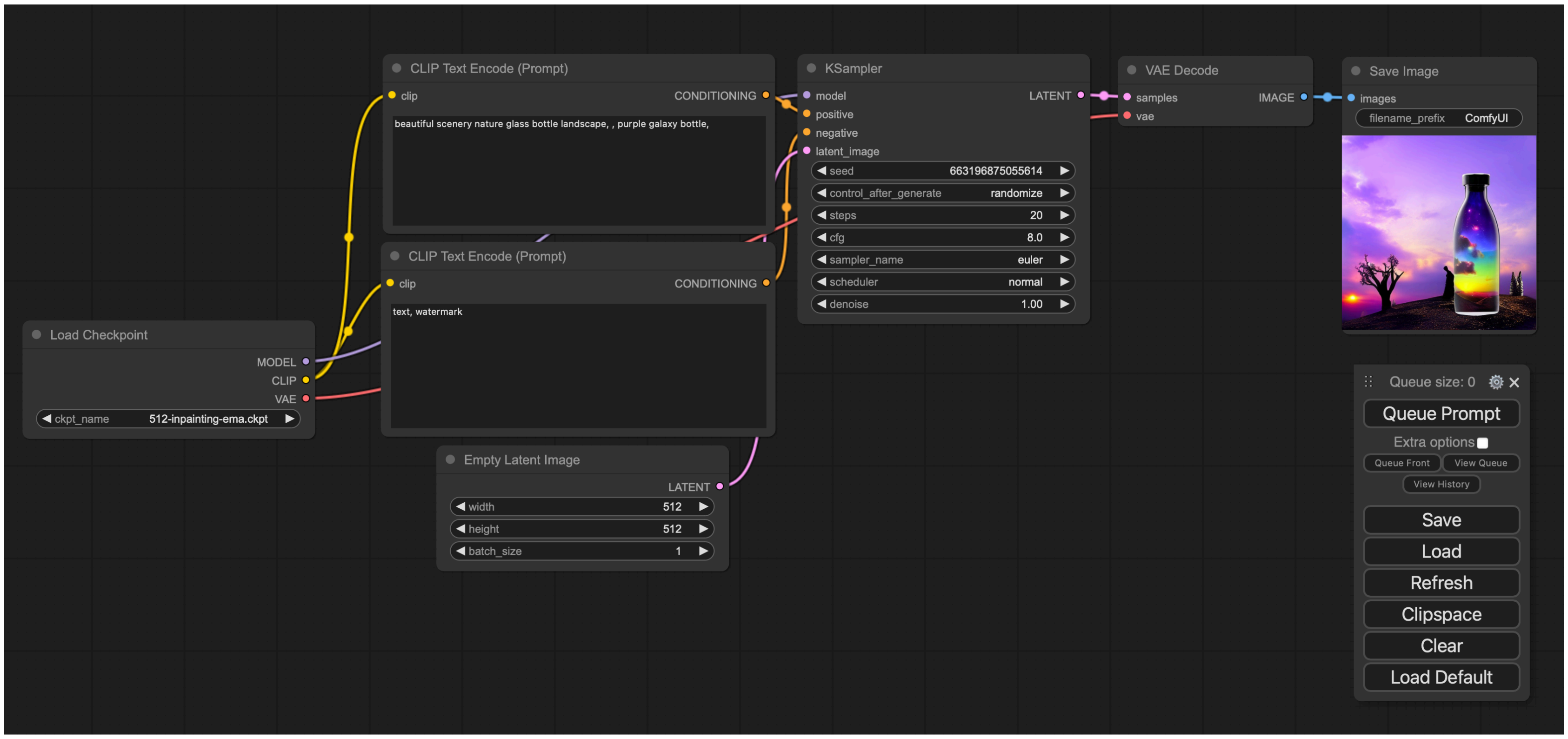
# Run ComfyUI interactively and as an API

View on GitHub

ComfyUI is a no-code Stable Diffusion GUI that allows you to design and execute advanced image generation pipelines.



beautiful scenery nature glass bottle landscape, , purple galaxy bottle,

text, watermark

seed 663196875055614
control_after_generate randomize
steps 20
cfg 8.0
sampler_name euler
scheduler normal
denoise 1.00

filename_prefix ComfyUI

ckpt_name 512-inpainting-ema.ckpt

width 512
height 512
batch_size 1

In this example, we show you how to

1. run ComfyUI interactively to develop workflows

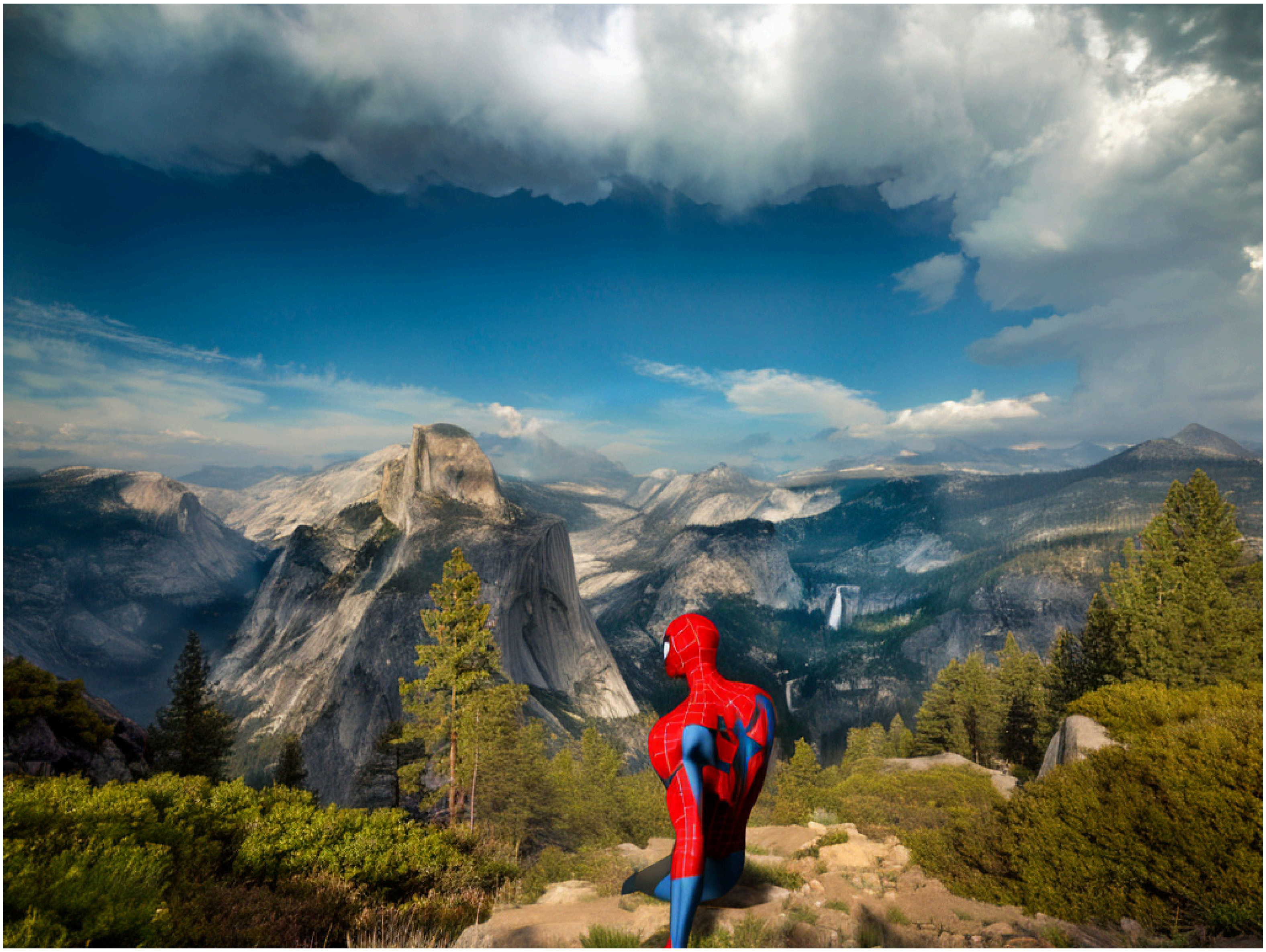
2. serve a ComfyUI workflow as an API

Combining the UI and the API in a single app makes it easy to iterate on your workflow even after deployment. Simply head to the interactive UI, make your changes, export the JSON, and redeploy the app.

An alternative approach is to port your ComfyUI workflow from the JSON format to Python code. The Python approach further reduces inference latency by a few hundred milliseconds to a second, but introduces some extra complexity. You can read more about it in this blog post.

## Quickstart

This example serves the ComfyUI inpainting example workflow, which "fills in" part of an input image based on a prompt. For the prompt `"Spider-Man visits Yosemite, rendered by Blender, trending on artstation"` on this input image, we got this output:



1. Stand up the ComfyUI server in development mode:

```
modal serve 06_gpu_and_ml/comfyui/comfyapp.py
```

2. In another terminal, run inference:

```
python 06_gpu_and_ml/comfyui/comfyclient.py --dev --modal-workspace your-modal-workspace -
```

You can find your Modal workspace name by running `modal profile current`.

The first inference will take a bit longer because the server will need to boot up (~20-30s). Successive inference calls while the server is up should take a few seconds or less.

## Setup

First, we define the environment we need to run ComfyUI — system software, Python package, etc.

You can add custom checkpoints and plugins to this environment by editing the `model.json` file in this directory.

```
import json
import pathlib
import subprocess
from typing import Dict

import modal

comfyui_commit_sha = "0fecfd2b1a2794b77277c7e256c84de54a63d860"

comfyui_image = (  # build up a Modal Image to run ComfyUI, step by step
    modal.Image.debian_slim(  # start from basic Linux with Python
        python_version="3.11"
    )
    .apt_install("git")  # install git to clone ComfyUI
    .run_commands(  # install ComfyUI
        "cd /root && git init .",
        "cd /root && git remote add --fetch origin https://github.com/comfyanonymous/Comfy
        f"cd /root && git checkout {comfyui_commit_sha}",
        "cd /root && pip install xformers!=0.0.18 -r requirements.txt --extra-index-url ht
    )
    .pip_install("httpx", "tqdm", "websocket-client")  # add web dependencies
    .copy_local_file(  # copy over the ComfyUI model definition JSON and helper Python mod
        pathlib.Path(__file__).parent / "model.json", "/root/model.json"
    )
    .copy_local_file(
        pathlib.Path(__file__).parent / "helpers.py", "/root/helpers.py"
    )
)

app = modal.App(name="example-comfyui")
```

Some additional code for managing ComfyUI lives in `helpers.py`. This includes functions like downloading checkpoints and plugins to the right directory on the ComfyUI server.

```
with comfyui_image.imports():
    from helpers import connect_to_local_server, download_to_comfyui, get_images
```

## Running ComfyUI interactively and as an API on Modal

Below, we use Modal's class syntax to run our customized ComfyUI environment and workflow on Modal.

Here's the basic breakdown of how we do it:

1. We add another step to the image `build` with `download_models`, which adds the custom checkpoints and plugins defined in `model.json`.
2. We stand up a "headless" ComfyUI server with `prepare_comfyui` when our app starts.
3. We serve a `ui` (by decorating with `@web_server`), so that we can interactively develop our ComfyUI workflow.
4. We stand up an `api` with `web_endpoint`, so that we can run our workflows as a service.

For more on how to run web services on Modal, check out this guide.

```
@app.cls(
    allow_concurrent_inputs=100,
    gpu="any",
    image=comfyui_image,
    timeout=1800,
    container_idle_timeout=300,
    mounts=[
        modal.Mount.from_local_file(
            pathlib.Path(__file__).parent / "workflow_api.json",
            "/root/workflow_api.json",
        )
    ],
)
class ComfyUI:
    @modal.build()
    def download_models(self):
        models = json.loads(
            (pathlib.Path(__file__).parent / "model.json").read_text()
        )
        for m in models:
            download_to_comfyui(m["url"], m["path"])

    def _run_comfyui_server(self, port=8188):
        cmd = f"python main.py --dont-print-server --listen --port {port}"
        subprocess.Popen(cmd, shell=True)

    @modal.enter()
    def prepare_comfyui(self):
        # runs on a different port as to not conflict with the UI instance
        self._run_comfyui_server(port=8189)

    @modal.web_server(8188, startup_timeout=30)
    def ui(self):
        self._run_comfyui_server()

    @modal.web_endpoint(method="POST")
    def api(self, item: Dict):
        from fastapi import Response

        # download input images to the container
        download_to_comfyui(item["input_image_url"], "input")
        workflow_data = json.loads(
            (pathlib.Path(__file__).parent / "workflow_api.json").read_text()
        )

        # insert the prompt
        workflow_data["3"]["inputs"]["text"] = item["prompt"]

        # send requests to local headless ComfyUI server (on port 8189)
        server_address = "127.0.0.1:8189"
        ws = connect_to_local_server(server_address)
        images = get_images(ws, workflow_data, server_address)
        return Response(content=images[0], media_type="image/jpeg")
```

## The workflow for developing workflows

When you run this script with `modal deploy 06_gpu_and_ml/comfyui/comfyapp.py` , you'll see a link that includes `ComfyUI.ui` . Head there to interactively develop your ComfyUI workflow. All of your custom checkpoints/plugins from `model.json` will be loaded in.

To serve the workflow after you've developed it, first export it as "API Format" JSON:

1. Click the gear icon in the top-right corner of the menu
2. Select "Enable Dev mode Options"
3. Go back to the menu and select "Save (API Format)"

Save the exported JSON to the `workflow_api.json` file in this directory.

Then, redeploy the app with this new workflow by running `modal deploy 06_gpu_and_ml/comfyui/comfyapp.py` again.

## Further optimizations

There is more you can do with Modal to further improve performance of your ComfyUI API endpoint.

For example:

- Apply keep_warm to the `ComfyUI` class to always have a server running
- Cache downloaded checkpoints/plugins to a Volume to avoid full downloads on image rebuilds

If you're interested in building a platform for running ComfyUI workflows with dynamically-defined dependencies and workflows, please reach out to us on Slack.