Using the API

Before starting, make sure you have followed the instructions in the setup page.

Talking to the API

Generating Video | Generating Images

In this example, we'll use the `gen4_image` model to generate an image of the Eiffel Tower rendered in the style of the painting Starry Night. To do this, we'll pass in reference images of the Eiffel Tower and Starry Night, and use the `promptText` to specify that we want the Eiffel Tower rendered in the style of Starry Night. Reference images can be referenced in the text prompt using at-mention syntax referencing the `tag` of the reference image.

Node | Python | Just testing

```
import RunwayML, { TaskFailedError } from '@runwayml/sdk';

const client = new RunwayML();

// Create a new image generation task using the "gen4_image" model
try {
  let task = await client.textToImage
    .create({
      model: 'gen4_image',
      ratio: '1920:1080',
      promptText: '@EiffelTower painted in the style of @StarryNight',
      referenceImages: [
        {
          uri: 'https://upload.wikimedia.org/wikipedia/commons/8/85/Tour_Eiffel_Wikimedia_Commons_
          tag: 'EiffelTower',
        },
        {
          uri: 'https://upload.wikimedia.org/wikipedia/commons/thumb/e/ea/Van_Gogh_-_Starry_Night_
          tag: 'StarryNight',
        },
      ],
    })
    .waitForTaskOutput();

  console.log('Task complete:', task);
  console.log('Image URL:', task.output[0]);
} catch (error) {
  if (error instanceof TaskFailedError) {
    console.error('The image failed to generate.');
    console.error(error.taskDetails);
  } else {
    console.error(error);
  }
}
```

Uploading base64 encoded images as data URIs

You can also upload base64 encoded images (as a data URI) instead of pointing to an external URL. This can be useful if you're working with a local image file and want to avoid an extra network round trip to upload the image.

To do this, simply pass the base64 encoded image string as a data URI in the `uri` of a `referenceImage` object instead of a URL. For more information about file types and size limits, see the Inputs page.

In this example, we'll use the `gen4_image` model to generate an image of a bunny facing the camera. We provide a reference image of a rabbit that we use at-mention syntax to reference in the `promptText`. We also provide an untagged reference image that the model will use to style the output image with.

Node | Python

```
import fs from 'node:fs';
import path from 'node:path';
import RunwayML, { TaskFailedError } from '@runwayml/sdk';

// Use the `mime-types` package to get the content type of the image file
// Install with `npm install mime-types --save`
import * as mime from 'mime-types';

const client = new RunwayML();

function getImageAsDataUri(imagePath: string) {
  // Read the image file
  const imageBuffer = fs.readFileSync(imagePath);

  // Convert to base64
  const base64String = imageBuffer.toString('base64');

  // Get the MIME type of the image file
  const contentType = mime.lookup(imagePath);

  return `data:${contentType};base64,${base64String}`;
}

// Create a new text-to-image task using the "gen4_image" model
try {
  const textToImage = await client.textToImage
    .create({
      model: 'gen4_image',
      promptText: '@bunny facing the camera',
      ratio: '1920:1080',
      referenceImages: [
        {
          uri: getImageAsDataUri('rabbit.png'),
          tag: 'bunny',
        },
        {
          uri: getImageAsDataUri('style.png'),
        },
      ],
    })
    .waitForTaskOutput();

  console.log('Task complete:', task);
} catch (error) {
  if (error instanceof TaskFailedError) {
    console.error('The image failed to generate.');
    console.error(error.taskDetails);
  } else {
    console.error(error);
  }
}
```