Quickstart

Get up and running with the Venice API in minutes. Generate an API key, make your first request, and start building.

Set up your API key

Add your API key to your environment. You can export it in your shell:

```
export VENICE_API_KEY='your-api-key-here'
```

Or add it to a `.env` file in your project:

```
VENICE_API_KEY=your-api-key-here
```

Install the SDK

Venice is OpenAI-compatible, so you can use the OpenAI SDK. If you prefer to use cURL or raw HTTP requests, you can skip this step.

Choose your model (optional)

Use Venice Parameters

You can choose to enable Venice-specific features like web search using `venice_parameters`:

```
import os
from openai import OpenAI

client = OpenAI(
    api_key=os.environ.get("VENICE_API_KEY"),
    base_url="https://api.venice.ai/api/v1"
)

completion = client.chat.completions.create(
```

```
    model="venice-uncensored",
    messages=[
        {"role": "user", "content": "What are the latest developments in AI?"}
    ],
    extra_body={
        "venice_parameters": {
            "enable_web_search": "auto",
            "include_venice_system_prompt": True
        }
    }
)

print(completion.choices[0].message.content)
```

See all available parameters.

Enable streaming (optional)

Stream responses in real-time using `stream=True`:

```
import os
from openai import OpenAI

client = OpenAI(
    api_key=os.environ.get("VENICE_API_KEY"),
    base_url="https://api.venice.ai/api/v1"
)

stream = client.chat.completions.create(
    model="venice-uncensored",
    messages=[{"role": "user", "content": "Write a short story about AI"}],
    stream=True
)

for chunk in stream:
    if chunk.choices and chunk.choices[0].delta.content is not None:
        print(chunk.choices[0].delta.content, end="")
```

Customize response behavior (optional)

Control how the model responds with parameters like temperature, max tokens, and more:

```
import os
from openai import OpenAI

client = OpenAI(
    api_key=os.environ.get("VENICE_API_KEY"),
    base_url="https://api.venice.ai/api/v1"
)

completion = client.chat.completions.create(
    model="venice-uncensored",
    messages=[
        {"role": "system", "content": "You are a creative storyteller"},
        {"role": "user", "content": "Tell me a creative story"}
    ],
    temperature=0.8,
    max_tokens=500,
    top_p=0.9,
    frequency_penalty=0.5,
    presence_penalty=0.5,
    extra_body={
        "venice_parameters": {
            "include_venice_system_prompt": False
        }
    }
)

print(completion.choices[0].message.content)
```

Check out the Chat Completions docs for more information on all supported parameters.

More Capabilities

Image Generation

Create images from text prompts using diffusion models:

```
import os
import requests

url = "https://api.venice.ai/api/v1/image/generate"

payload = {
    "model": "venice-sd35",
    "prompt": "A cyberpunk city with neon lights and rain",
    "width": 1024,
    "height": 1024,
    "format": "webp"
}

headers = {
    "Authorization": f"Bearer {os.getenv('VENICE_API_KEY')}",
    "Content-Type": "application/json"
}

response = requests.post(url, json=payload, headers=headers)

print(response.json())
```

Note: The response returns base64-encoded images in the `images` array. Decode the base64 string to save or display the image. **Popular Image Models:**

- `qwen-image` - Highest quality image generation
- `venice-sd35` - Default choice, works with all features
- `hidream` - Fast generation for production use

View All Image Models

See all available image models with pricing and capabilities

For more advanced parameter options like `cfg_scale`, `negative_prompt`, `style_preset`, `seed`, `variants`, and more, check out the Images API Reference.

Image Editing

Modify existing images with AI-powered inpainting using the Qwen-Image model:

```
import os
import requests
import base64

url = "https://api.venice.ai/api/v1/image/edit"

with open("image.jpg", "rb") as f:
    image_base64 = base64.b64encode(f.read()).decode('utf-8')

payload = {
    "prompt": "Colorize",
    "image": image_base64
}

headers = {
    "Authorization": f"Bearer {os.getenv('VENICE_API_KEY')}",
    "Content-Type": "application/json"
}

response = requests.post(url, json=payload, headers=headers)

with open("edited_image.png", "wb") as f:
    f.write(response.content)
```

Note: The image editor uses the Qwen-Image model and is an experimental endpoint. Send the input image as a base64-encoded string, and the API returns the edited image as binary data. See the Image Edit API for all

parameters.

Image Upscaling

Enhance and upscale images to higher resolutions:

```
import os
import requests
import base64

url = "https://api.venice.ai/api/v1/image/upscale"

with open("image.jpg", "rb") as f:
    image_base64 = base64.b64encode(f.read()).decode('utf-8')

payload = {
    "image": image_base64,
    "scale": 2
}

headers = {
    "Authorization": f"Bearer {os.getenv('VENICE_API_KEY')}",
    "Content-Type": "application/json"
}

response = requests.post(url, json=payload, headers=headers)

with open("upscaled_image.png", "wb") as f:
    f.write(response.content)
```

Note: Send the input image as a base64-encoded string, and the API returns the upscaled image as binary data. See the Image Upscale API for all parameters.

Text-to-Speech

Convert text to audio with 60+ multilingual voices:

```
import os
import requests

response = requests.post(
    "https://api.venice.ai/api/v1/audio/speech",
    headers={
        "Authorization": f"Bearer {os.getenv('VENICE_API_KEY')}",
        "Content-Type": "application/json"
    },
    json={
        "input": "Hello, welcome to Venice Voice.",
        "model": "tts-kokoro",
        "voice": "af_sky"
    }
)

with open("speech.mp3", "wb") as f:
    f.write(response.content)
```

The `tts-kokoro` model supports 60+ multilingual voices including `af_sky`, `af_nova`, `am_liam`, `bf_emma`, `zf_xiaobei`, and `jm_kumo`. See the TTS API for all voice options.

Embeddings

Generate vector embeddings for semantic search, RAG, and recommendations:

```
import os
import requests

url = "https://api.venice.ai/api/v1/embeddings"

payload = {
    "model": "text-embedding-bge-m3",
    "input": "Privacy-first AI infrastructure for semantic search",
    "encoding_format": "float"
```

```
}

headers = {
    "Authorization": f"Bearer {os.getenv('VENICE_API_KEY')}",
    "Content-Type": "application/json"
}

response = requests.post(url, json=payload, headers=headers)

print(response.json())
```

See the Embeddings API for batch processing and advanced options.

Vision (Multimodal)

Analyze images alongside text using vision-capable models like `mistral-31-24b`:

```
import os
from openai import OpenAI

client = OpenAI(
    api_key=os.getenv("VENICE_API_KEY"),
    base_url="https://api.venice.ai/api/v1"
)

response = client.chat.completions.create(
    model="mistral-31-24b",
    messages=[
        {
            "role": "user",
            "content": [
                {"type": "text", "text": "What is in this image?"},
                {
                    "type": "image_url",
                    "image_url": {"url": "https://www.gstatic.com/webp/gallery/1
                }
```

```
            ]
        }
    ]
)

print(response.choices[0].message.content)
```

Function Calling

Define functions that models can call to interact with external tools and APIs:

```
import os
from openai import OpenAI

client = OpenAI(
    api_key=os.getenv("VENICE_API_KEY"),
    base_url="https://api.venice.ai/api/v1"
)

tools = [
    {
        "type": "function",
        "function": {
            "name": "get_weather",
            "description": "Get the current weather in a location",
            "parameters": {
                "type": "object",
                "properties": {
                    "location": {
                        "type": "string",
                        "description": "The city and state"
                    }
                },
                "required": ["location"]
            }
        }
    }
```

```
]

response = client.chat.completions.create(
    model="zai-org-glm-4.6",
    messages=[{"role": "user", "content": "What's the weather in San Francisco?"
    tools=tools
)

print(response.choices[0].message)
```

Next Steps

Now that you've made your first requests, explore more of what Venice API has to offer:

Additional Resources

Need Help?

- **Discord Community**: Join our Discord server for support and discussions
- **Documentation**: Browse our complete API reference
- **Status Page**: Check service status at veniceai-status.com
- **Twitter**: Follow @AskVenice for updates