Crane + Qwen3-TTS: Run Voice Cloning Locally with Rust

February 23, 2026 · by Mark Bartlett

Download this guide as PDF

> **Quick Answer:** Qwen3-TTS (1.7B params, Apache 2.0, Jan 2026) clones voices from 3 seconds of reference audio and generates speech faster than real-time (RTF 0.87) on an RTX 3060 or better. It outperforms ElevenLabs on speaker similarity benchmarks (0.789 vs 0.646). Crane is a pure Rust inference engine (268 stars, by lucasjinreal) that added Qwen3-TTS support on Feb 23, 2026 — zero Python dependencies, OpenAI-compatible /v1/audio/speech endpoint, CUDA and Metal support. For most users, the official Python package (pip install qwen-tts) is the fastest path. Crane is interesting if you want a single Rust binary with no Python.

📚 **Related:** Voice Chat with Local LLMs · VRAM Requirements · Fine-Tuning with LoRA · Run Your First Local LLM · Planning Tool

ElevenLabs charges $22/month for voice cloning. OpenAI's TTS API costs $15 per million characters. Both send your audio to someone else's servers.

Qwen3-TTS (GitHub, released Jan 22, 2026) clones voices from 3 seconds of reference audio, runs entirely on your hardware, costs nothing after your GPU investment, and is Apache 2.0 licensed. It outperforms ElevenLabs on speaker similarity benchmarks across 10 languages.

Crane (GitHub, 268 stars) is a pure Rust inference engine built on Candle that added Qwen3-TTS support on February 23, 2026. No Python, no pip, no venv. One binary.

Why Qwen3-TTS

The numbers speak for themselves.

Metric	Qwen3-TTS 1.7B	ElevenLabs	OpenAI TTS
Speaker similarity (avg 10 languages)	**0.789**	0.646	N/A
Word Error Rate (English)	**1.24**	Higher	Higher
Voice cloning reference	3 seconds	30+ seconds	Not available
VRAM	~4 GB	Cloud	Cloud

Metric	Qwen3-TTS 1.7B	ElevenLabs	OpenAI TTS
Cost	$0 (after GPU)	$22/month	$15/M chars
Languages	10	29	57
License	Apache 2.0	Proprietary	Proprietary

Qwen3-TTS is a discrete multi-codebook language model with 16 codebooks at 12Hz sampling rate. It supports both streaming and non-streaming output with end-to-end latency as low as 97ms. The model comes in two sizes — 1.7B (recommended) and 0.6B (lightweight).

The catch: English preset voices have a subtle “anime-like” quality from training data bias toward dubbed animation content. Using voice cloning with a native English reference sample solves this. And only 10 languages versus ElevenLabs’ 29.

Model Variants

Model	Params	Voice Clone	Instruction Control	Best For
Qwen3-TTS-12Hz-1.7B-Base	1.7B	Yes (3-sec)	No	Voice cloning
Qwen3-TTS-12Hz-1.7B-CustomVoice	1.7B	No	Yes (9 voices)	Preset voice generation
Qwen3-TTS-12Hz-0.6B-Base	0.6B	Yes (3-sec)	No	Lightweight cloning

For voice cloning, you need the **Base** variant. The CustomVoice variant provides preset voices with instruction control but cannot clone from reference audio.

Option A: Official Python Package (Easiest)

```
conda create -n qwen3-tts python=3.12 -y
conda activate qwen3-tts
pip install -U qwen-tts
pip install -U flash-attn --no-build-isolation  # optional, +10% speed
```

Clone a Voice

```
import torch
import soundfile as sf
from qwen_tts import Qwen3TTSModel

model = Qwen3TTSModel.from_pretrained(
    "Qwen/Qwen3-TTS-12Hz-1.7B-Base",
    device_map="cuda:0",
    dtype=torch.bfloat16,
    attn_implementation="flash_attention_2",
)

# Provide 3+ seconds of reference audio + transcript
wavs, sr = model.generate_voice_clone(
    text="Text you want spoken in the cloned voice.",
    language="English",
    ref_audio="reference.wav",
    ref_text="This is the transcript of the reference audio.",
)
sf.write("output.wav", wavs[0], sr)
```

Batch Generation (Build Prompt Once)

```
prompt_items = model.create_voice_clone_prompt(
    ref_audio="reference.wav",
    ref_text="Transcript of reference.",
    x_vector_only_mode=False,
)

wavs, sr = model.generate_voice_clone(
    text=["Sentence one.", "Sentence two.", "Sentence three."],
    language=["English", "English", "English"],
    voice_clone_prompt=prompt_items,
)
for i, wav in enumerate(wavs):
    sf.write(f"output_{i}.wav", wav, sr)
```

Built-In Web UI

```
qwen-tts-demo Qwen/Qwen3-TTS-12Hz-1.7B-Base --ip 0.0.0.0 --port 8000
```

Open `http://localhost:8000` for a Gradio interface with voice cloning and custom voice design.

Option B: Crane (Rust, No Python)

Crane is a pure Rust inference engine that handles LLMs, vision models, TTS, OCR, and ASR — all powered by Candle. If you want zero Python dependencies and an OpenAI-compatible API, this is the path.

Build

```
git clone https://github.com/lucasjinreal/Crane.git
cd Crane

# CUDA GPU build
cargo build -p crane-oai --release --features cuda

# Or CPU-only
cargo build -p crane-oai --release
```

Requires a recent Rust toolchain. Metal (macOS) is also supported.

Run the TTS Server

```
./target/release/crane-oai \
  --model-path /path/to/Qwen3-TTS-12Hz-1.7B-Base \
  --port 8000
```

This exposes an OpenAI-compatible `/v1/audio/speech` endpoint. Point any OpenAI TTS client at it:

```
curl http://localhost:8000/v1/audio/speech \
  -H "Content-Type: application/json" \
  -d '{"model": "qwen3-tts", "input": "Hello from local TTS.", "voice": "alloy"}' \
  --output speech.wav
```

Crane Performance

On Apple Silicon M1, Crane achieves 5-6x speedup over vanilla PyTorch for LLM inference. The TTS path is newer (added Feb 23, 2026) and benchmarks are limited, but Candle's Rust kernels are competitive with PyTorch for transformer workloads.

Option C: Docker (OpenAI-Compatible Server)

For a production-style deployment with an OpenAI-compatible API:

```
git clone https://github.com/groxaxo/Qwen3-TTS-Openai-Fastapi.git
cd Qwen3-TTS-Openai-Fastapi

# GPU deployment
docker-compose up qwen3-tts-gpu

# Or with vLLM backend (slightly faster)
docker-compose --profile vllm up qwen3-tts-vllm
```

This gives you a drop-in replacement for OpenAI's TTS API that runs entirely on your hardware.

Hardware Requirements

Setup	Minimum GPU	VRAM	RTF (Speed)
1.7B model	RTX 3060 12GB	~4 GB	0.87 (faster than real-time)
1.7B + FlashAttention 2	RTX 3060 12GB	~3 GB	0.87
0.6B model	GTX 1060 6GB	~2-4 GB	0.52-0.68

Setup	Minimum GPU	VRAM	RTF (Speed)
Production (multi-user)	RTX 3090 24GB	~4 GB	0.83 (vLLM)

RTF (Real-Time Factor) below 1.0 means the model generates audio faster than you can listen to it. At 0.87, a 10-second clip takes about 8.7 seconds to generate.

Latency by text length (1.7B on RTX 3090):

Text Length	Median Latency
Short (2 words)	1.01s
Sentence (7 words)	3.29s
Medium (20 words)	8.50s
Long (36 words)	21.16s

The Local TTS Landscape (Feb 2026)

Qwen3-TTS isn't the only option. Here's how it compares:

Model	Params	VRAM	Clone	Languages	RTF	License
Qwen3-TTS 1.7B	1.7B	~4 GB	3-sec zero-shot	10	0.87	Apache 2.0
Qwen3-TTS 0.6B	0.6B	~2-4 GB	3-sec zero-shot	10	0.52	Apache 2.0
Orpheus TTS	3B	~15 GB	Zero-shot	English+	~1.0	Apache 2.0
Chatterbox Turbo	350M	~4-8 GB	5-sec zero-shot	English+	<1.0	Apache 2.0
Kani-TTS-2	400M	3 GB	Zero-shot	2	0.2	Apache 2.0
XTTS-v2 (Coqui fork)	~1.5B	~4-6 GB	6-sec zero-shot	17	~1.0	Non-commercial
Piper	Varies	CPU-only	Training only	30+	0.1	MIT

Orpheus (3B) excels at emotional speech with emotive tags like `<laugh>` and `<sigh>`, but needs 15GB VRAM. **Kani-TTS-2** (400M, released Feb 15, 2026) is remarkably fast at RTF 0.2 on just 3GB VRAM, but only supports English and Portuguese. **Coqui/XTTS-v2** is in maintenance mode after Coqui AI shut down in December 2025 — the community fork at idiap/coqui-ai-TTS still works but there won't be a v3.

For most use cases in February 2026, Qwen3-TTS 1.7B is the best overall choice: best quality-to-VRAM ratio, broadest language support among open models, and an active development team at Alibaba.

Ethics and Legal

Voice cloning raises real concerns. The Biden deepfake robocall in January 2024 resulted in a $6 million FCC penalty. The EU AI Act requires labeling AI-generated audio. California's AI Transparency Act (AB 942, effective January 2026) mandates disclosure.

Qwen3-TTS's model card states: "you agree to inform listeners that speech samples are synthesized" and "you agree to only use voices whose speakers grant permission."

Practical guidelines:

- Only clone voices you have explicit permission to use
- Label generated audio as AI-synthesized
- Clone your own voice for personal projects — it's the safest path
- Chatterbox embeds PerTh neural watermarks in generated audio for provenance tracking, if traceability matters to your use case

---

Bottom Line

Qwen3-TTS gives you ElevenLabs-quality voice cloning on 4GB of VRAM, Apache 2.0, completely free. The Python package (`pip install qwen-tts`) is the fastest path. Crane adds a zero-dependency Rust option with an OpenAI-compatible API, though its TTS support is one day old.

If you have an RTX 3060 or better, you can clone voices locally today. If you're on a Mac, Crane's Metal support makes it the first Rust-native TTS inference engine for Apple Silicon.

The English accent issue is real but solvable — clone from a native English speaker's audio instead of using presets. For everything else, this is the best open-source TTS available.

GitHub: QwenLM/Qwen3-TTS · GitHub: lucasjinreal/Crane
