Intègre un assistant vocal dans ta webapp

11 juin 2025

Godefroy de Compreignac

CEO @ Lonestone

Présentation de l'intervenant

Godefroy de Compreignac

Co-fondateur CEO @ Lonestone

Entrepreneur et développeur depuis 20 ans.

Early adopter enthousiaste de l’IA générative.

Agence de développement d'outils métier et de SaaS intégrant de l'IA

Île de Nantes

30+ experts salariés

Début 2024...

✅ ChatGPT était déjà bien adopté

❌ Le mode vocal n'était pas encore une option

À l'origine de mes pérégrinations

En quête d'un ghostwriter...

Pourquoi pas une IA vocale pour m'aider à me questionner et à écrire ?

Raconte

lonestone

Je commence à coder un proto.

Et je teste toutes les startups d'IA vocale qui sortent.

Vapi Fixed Cost · talkscriber · gpt 4o · eleven multilingual v2 · web

Web

Cost ~$0.13 /min

Latency ~2150 ms

MODEL

Model

Configure the behavior of the assistant.

Provider: OpenAI

Model: GPT 4o Cluster

First Message Mode: Assistant speaks first

First Message: Bonjour, je suis l'IA de Lonestone, le product studio qui crée vos apps et IA pour propulser votre business. Comment puis-je vous aider ?

System Prompt — Generate

Expertises de Lonestone :
- Design et stratégie
- Développement web et mobile
- Intelligence artificielle

Offres de Lonestone :
- Shape : On s'immerge dans votre métier et vos objectifs, on crée un prototype cliquable pour le tester. Vous repartez avec une roadmap claire, le design et des recommandations techniques.
- Build : On constitue une équipe Product sur mesure pour développer votre produit en agilité avec vous. On déploie en continu pour tester rapidement et itérer afin de tenir vos objectifs.
- Run : On s'occupe de la maintenance de votre produit: hébergement, mises à jour, corrections et évolutions à la demande
- Scale : On vous propose les devs et designers à la demande en fonction de vos besoins du moment pour rejoindre votre équipe à distance, pour une courte ou une longue durée, sans engagement.
- Audit : On analyse votre produit: sa sécurité, son code ou son UX. Puis on vous concote un rapport détaillé avec des recommandations activables et priorisées. Vous pouvez ensuite nous en confier la mise en oeuvre.

Files

Select Files

Max Tokens

Temperature

VOICE

Voice Configuration

Select a voice from the list, or sync your voice library if it's missing. If errors persist, enable custom voice and add a voice ID.

You are using fr language for your transcriber. Make sure you choose a voice compatible with fr.

Provider: 11labs

Voice: Tht5kcbeypx3keuqqhph

Add Voice ID Manually

Model

This is the model that will be used.

Eleven_multilingual_v2

Additional Configuration

Configure additional settings for the voice of your assistant.

TRANSCRIBER

Transcriber

This section allows you to configure the transcription settings for the assistant.

Provider: Talkscriber

Language: Fr

Model: Whisper

Background Denoising Enabled

Filter background noise while the user is talking.

TOOLS

TOOLS

Tools

Tools enable voicebots to perform actions during calls. Add tools from the Tools Library to connect with Make.com or GHL workflows, or create custom tools with your backend.

Note: Tools have different Request and Response format as compared to Functions. Check our tools guide for more details

Select Tools

Predefined Functions

We've pre-built functions for common use cases. You can enable them and configure them below.

Enable End Call Function
This will allow the assistant to end the call from its side. *(Best for gpt-4 and bigger models.)* *Read More*

Dial Keypad
This sets whether the assistant can dial digits on the keypad. *Read More*

Forwarding Phone Number

+33 6 11 89 13 84

Custom Functions

Define your custom functions here to enhance your assistant's capabilities. You can use your own code or tools like **Pipedream** or **Make** for the setup.

ANALYSIS

Summary

This is the prompt that's used to summarize the call. The output is stored in **call.analysis.summary**. You can also find the summary in the Call Logs Page.

Prompt

You are an expert note-taker. You will be given a transcript of a call. Summarize the call in 2-3 sentences, if applicable.

Même fonctionnement partout :

Orchestration de plusieurs modèles

Rappel : quelques types de modèles

LLM - Large Language Model (Text-to-Text)

Traitements textuels :

- Répondre à une question
- Planifier
- Appeler un outil
- Générer du code

STT - Speech-to-text

Retranscrire de l'audio en texte.

TTS - Text-to-Speech

Générer de la voix, avec clonage, émotion, etc.

Text-to-Image

Générer des images

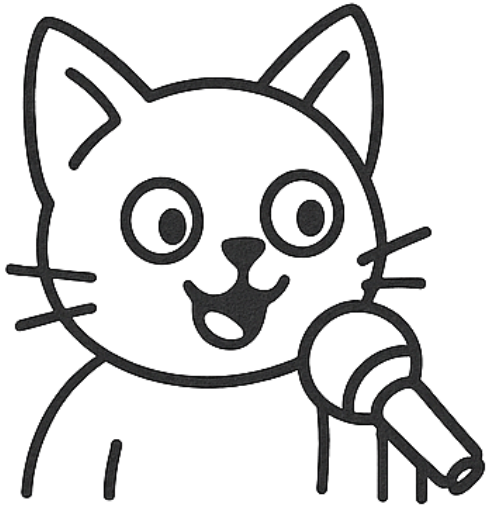
Image-to-Text

Analyser des images

Multimodal

Plusieurs capacités simultanées : texte, images, audio...

... et bien d'autres : Text-to-Video, Image-to-Video, Image-to-3D, Voice Activity Detection, etc.



L'utilisateur parle

STT

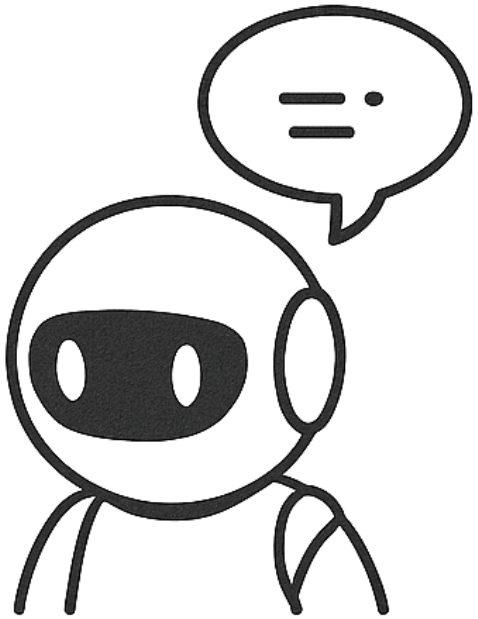
Transcription

LLM

Génération de réponse texte

TTS

Génération de la voix



L'assistant répond

lonestone

Le gros sujet : la latence

Latence entre deux humains

< 200 ms : très réactif

200–500 ms : naturel

> 700 ms : hésitation, latence gênante

> 1 seconde : signal d'émotion (surprise, désaccord, réflexion) ou problème technique (dans les appels)

➡️ Objectif : viser 500ms max

Latence

Orchestration de modèles	vs	Modèle multimodal
Détection de fin de speech + Transcription (STT) + Génération réponse (LLM) + Génération voix (TTS) + Echanges réseau		Speech-to-Speech (accessible depuis fin 2024)
> 2 secondes (sans optimisations)		350-800 ms

Démo
OpenAI Realtime

lonestone

Multimodal = problème résolu ?

Pourquoi on s'embête encore
avec de l'orchestration de modèles ?

Limitations du multimodal

💰 Très coûteux (3-5x plus)

🗣️ Nombre de voix limitées

🔗 Dépendance aux géants de la tech

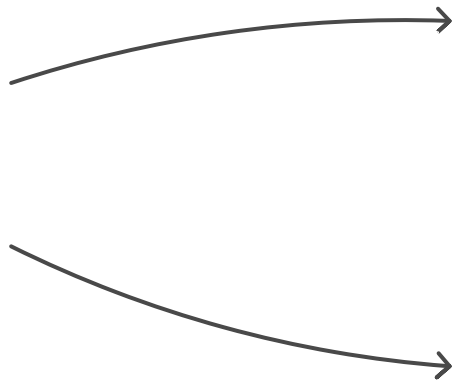
🔧 Peu de contrôle

OpenAI Gemini

L'orchestration de modèles permet :

🕊️ Choix des providers

🔧 Contrôle quasi total



🤑 Maîtrise des coûts

🗣️ Grand choix de voix, clonage

📐 Précision dans la langue visée

🕊️ Souverain / self-hosting / local

♾️ Cas d'usage illimités

🔬 Evaluation plus simple et moins chère

👌 UX aux petits oignons

La latence, c'est pas si grave

Pour plein de cas d'usage.
Quand l'utilisateur sait qu'il parle à une IA.
Quand on veut laisser le temps à l'utilisateur de répondre.

Exemple : questionnaire vocal

La latence, ça s'optimise

- Choix de fournisseurs performants
- Choix de modèles rapides
- Optimisation de prompt
- Streaming à chaque étape
- Détection de voix sémantique

La preuve

Kyutai 🇫🇷 a annoncé le 22 mai 2025
sa solution d'orchestration et ses modèles STT et TTS
à très faible latence

Démo :

https://unmute.sh/

Bientôt open source...

Démo
Unmute

lonestone

Fournisseurs IA vocales

Quelques exemples parmi les meilleurs

Speech-to-Text

- Gladia 🇫🇷
- OpenAI
- Deepgram

Text-to-Speech

- IIElevenLabs
- OpenAI
- Cartesia

Frameworks d'orchestration

SaaS (propriétaire)

- VAPI
- Voiceflow
- Retell AI
- PLAY.HT
- BLAND.AI

Open source

- LiveKit
- Pipecat
- vocode

La plupart des frameworks d'orchestration
sont propriétaires, prévus pour la téléphonie,
trop complexes ou trop limités, en python...

donc j'ai créé Micdrop

Micdrop

https://github.com/lonestone/micdrop

- ✅ Pour le web
- ✅ Open source
- ✅ Full Typescript
- ✅ Extensible
- ✅ Dépendances npm client/server

utilisé sur

Raconte

CIBLI

Mettez une étoile ⭐ sur Github svp 🙏

Démo
Micdrop

lonestone

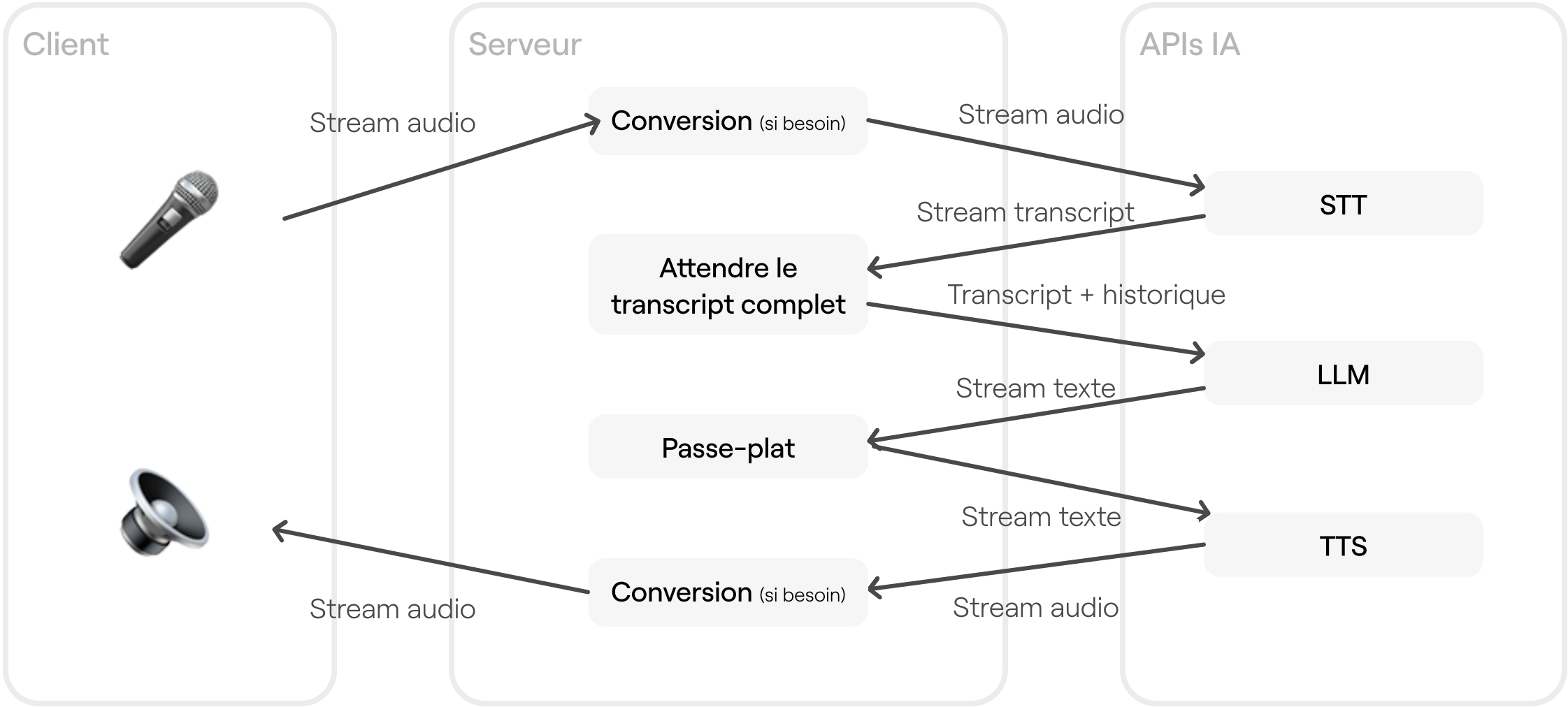
Quelques apprentissages...

Latence : au delà des modèles...

Une fois qu'on a choisi des modèles rapides
qui tournent sur des infrastructures performantes, comment optimiser ?

La clé, c'est le streaming.

Streaming de bout-en-bout



lonestone

Transport : quel protocole utiliser ?

WebRTC

Flux audio/vidéo + texte
entre pairs (P2P)

- ✅ Latence ultra-faible
- ❌ UDP : perte de paquets possible
- 🤔 Plus compliqué à mettre en œuvre

VS

WebSocket

Echange bi-directionnelle
de texte et binaire (client/serveur)

- 🤔 Latence un peu plus élevée
- ✅ TCP : pas de perte
- ✅ Très facile à installer et maintenir

➡️ **Choix des APIs d'IA**
➡️ **Choix pour Micdrop**

VAD - Voice Activity Detection

Pour déterminer quand transcrire le flux audio du micro
et quand c'est à l'assistant de parler

Seuil de Volume

✅ Simple et performant

🤔 Insuffisant seul, confond bruits parasites et voix

+

Détection voix humaine

✅ Indispensable pour bien détecter la voix

🤔 Un peu plus lourd (modèle ML)

🤔 Insuffisant seul, capte trop les voix lointaines

silero

➡️ Micdrop combine les deux

VAD - Voice Activity Detection

À qui la responsabilité ?

Côté client

✅ Optimisation bande passante
✅ Moins lourd pour le serveur
✅ Interruption plus rapide
🤔 Potentiellement un peu lourd côté client
🤔 Compatibilité navigateurs anciens/exotiques ?

➡️ Idéal pour une conversation longue où l'utilisateur prend son temps.
➡️ **Choix pour Micdrop**

VS

Côté serveur

🤔 Envoi continu, bande passante importante
🤔 Lourd pour le serveur (cumul les users)
🤔 Délai d'interruption potentiellement désagréable
✅ Léger côté client

➡️ **Choix de la plupart des frameworks**

lonestone

Interruptions de l'assistant par l'utilisateur

💡 Décider si c'est souhaitable ou pas pour le cas d'usage.

(option `disableInterruption` dans micdrop)

Mécanisme d'abandon du processus en cours en cas de détection de voix :

- 🔲 Arrêt immédiat de l'audio
- 🔲 Arrêt du stream audio du serveur vers le client
- 🔲 Arrêt de la génération de voix (TTS)
- 🔲 Arrêt de la génération de réponse (LLM)

Eviter les interruptions de l'utilisateur par l'assistant pendant une pause dans une phrase non terminée

Détection par le LLM (prompt) de deux cas de figure :

1️⃣ Transcription à ignorer

Pas de valeur, on supprime et on ne répond pas.

Exemple : onomatopées ("Oh", "Ahem"...)

2️⃣ Réponse à retarder

La phrase ne semble pas sémantiquement complète.

On garde mais on ne répond pas tout de suite.

Exemple : "Je pense que", "Est-ce que tu peux"

➡️ La commande détectée est passée à Micdrop

```
If the last user message is just an interjection or a
sound that expresses emotion, hesitation, or
reaction (ex: "Uh", "Ahem", "Hmm", "Ah") but doesn't
carry any clear meaning like agreeing, refusing, or
commanding, just say ${CANCEL_LAST_USER_MESSAGE}.

If the last user message is an incomplete sentence,
just say ${SKIP_ANSWER}.
```

lonestone

Phrases de remplissage quand l'assistant "réfléchit"

Tool calling = souvent trop long

Exemples : Recherche sur le web, enregistrement d'une réservation.

💡 On peut jouer des sons pré-enregistrés

Exemples : "Ok je vois", "Laisse-moi réfléchir", "Je cherche ça..."

Gérer la fin de conversation

Détection par le LLM (prompt) quand l'utilisateur demande à finir.

➡️ La commande détectée est passée à Micdrop

```
If the user asks to end the call,
say goodbye and say ${END_CALL}.
```

Gestion des devices

🎤 Choix du microphone
📶 Test du micro (indicateur de volume)

🔈 Choix du speaker
⏯️ Test du speaker

🔃 Détection de changement de devices
(ex : nouvel appareil bluetooth connecté)

⚠️ Contourner la sécurité anti pub intempestive

```ts
// Inspired https://www.mattmontag.com/web/unlock-web-audio-in-safari-for-ios-and-macos
export function unlock(audioContext: AudioContext): void {
  const events = ['touchstart', 'touchend', 'mousedown', 'keydown']

  const unlock = async () => {
    unlockSafari(audioContext)
    if (audioContext.state === 'suspended') {
      await audioContext.resume()
    }
    // Clean events
    events.forEach((e) => document.body.removeEventListener(e, unlock))
  }

  // Add events
  events.forEach((e) => document.body.addEventListener(e, unlock, false))
}

// safari/ios hack - inspired by https://codepen.io/kslstn/pen/pagLqL
// unlock audioContext - call when interactive (button click)
function unlockSafari(audioContext: AudioContext): void {
  // create empty buffer and play it to unlock audioContext when interactive
  const source = audioContext.createBufferSource()
  source.buffer = audioContext.createBuffer(1, 1, 22050)
  source.connect(audioContext.destination)
  const onEnded = () => {
    source.disconnect()
    source.buffer = null
    source.removeEventListener('ended', onEnded)
  }
  source.addEventListener('ended', onEnded)
  source.start()
}
```

Installer Micdrop

Micdrop côté client

```
npm install @micdrop/client
```

```
import { CallClient } from '@micdrop/client'

const call = CallClient.getInstance()
call.url = 'wss://localhost/call'

await call.start()
```

Micdrop côté serveur

```
npm install @micdrop/server
```

```
import { WebSocketServer } from 'ws'
import { CallServer, CallConfig } from '@micdrop/server'

const wss = new WebSocketServer({ port: 8080 })

wss.on('connection', (ws) => {
  new CallServer(ws, {
    systemPrompt: 'You are a helpful assistant',
    firstMessage: 'Hello!',

    async generateAnswer(conversation) {
      return 'Assistant response'
    },

    async speech2Text(audioBlob, lastMessagePrompt) {
      return 'Transcribed text'
    },

    async text2Speech(text) {
      return new ArrayBuffer(0)
    },
  })
})
```

Bientôt la V2

⚡Meilleure gestion du streaming

🔌 Plus d'intégrations clé en main

💥 Phrases de remplissage

(voir branche v2 sur Github, WIP)

lonestone

Merci pour votre attention !

Godefroy de Compreignac

CEO @ Lonestone

Restons discuter 🙌

(et ajoutez-moi sur Linkedin)