



# JARVIS SETUP GUIDE - INSTALLAZIONE DEFINITIVA



## GUIDA AGGIORNATA POST-SVILUPPO

Questa guida è stata aggiornata basandosi sui **test reali** e riflette il **processo di installazione effettivamente validato**.

---



## QUICK START - 15 MINUTI



### PREREQUISITI VERIFICATI:

- **OS:** Windows 10/11, macOS 10.15+, Ubuntu 20.04+
  - **Python:** 3.9-3.11 (testato su 3.11)
  - **RAM:** 8GB minimum, 16GB recommended
  - **Storage:** 5GB liberi
  - **Internet:** Solo per installazione iniziale
- 



## STEP 1: INSTALLAZIONE BASE

### 1.1 Clone Repository:

```
bash

git clone [repository-url]
cd Jarvis
```

### 1.2 Crea Virtual Environment:

```
bash

# Windows
python -m venv jarvis_env
jarvis_env\Scripts\activate

# macOS/Linux
python3 -m venv jarvis_env
source jarvis_env/bin/activate
```

### 1.3 Installa Requirements:

```
bash
```

```
pip install --upgrade pip
pip install -r requirements.txt
```

⚠ **Se PyAudio fallisce su Windows:**

```
bash

pip install pipwin
pipwin install pyaudio
```

## 🧠 **STEP 2: INSTALLAZIONE OLLAMA + MISTRAL**

### **2.1 Installa Ollama:**

- **Windows/macOS:** Scarica da <https://ollama.ai>
- **Linux:** `curl -fsSL https://ollama.com/install.sh | sh`

### **2.2 Installa Modello Mistral:**

```
bash

ollama pull mistral:7b
ollama serve
```

### **2.3 Verifica Ollama:**

```
bash

# Test connessione
curl http://localhost:11434/api/generate -d '{"model":"mistral:7b","prompt":"Hello"}'
```

## 🔊 **STEP 3: INSTALLAZIONE COMPONENTI VOICE**

### **3.1 Verifica Dipendenze Voice:**

```
bash

python -c "import whisper, piper_tts, pyaudio; print('✅ Voice stack OK')"
```

### **3.2 Test Voice Manager:**

```
bash
```

```
cd core
python voice_manager.py
```

## Output atteso:

- ✓ Whisper disponibile per STT locale
- ✓ Piper disponibile per TTS neurale locale
- ✓ PyAudio disponibile
- 🔊 Voice Manager inizializzato completamente

## 🌐 STEP 4: AVVIO SISTEMA COMPLETO

### 4.1 Avvia WebSocket Server:

```
bash

cd core
python websocket_server.py
```

### 4.2 Apri Frontend:

```
bash

# Apri browser su:
http://localhost:8765
```

### 4.3 Test Sistema:

1. ✓ **Frontend si carica** con animazioni Jarvis
2. ✓ **Metrics si aggiornano** (CPU, RAM, etc.)
3. ✓ **Chat funziona** (scrivi messaggio)
4. ✓ **Voice funziona** (di' "Ehi Jarvis, ciao")

## 🔧 RISOLUZIONE PROBLEMI COMUNI

### ✗ Problema: PyAudio Import Error

```
bash
```

```
# Windows - Soluzione wheel precompilato
pip uninstall pyaudio
# Scarica da: https://www.lfd.uci.edu/~gohlke/pythonlibs/#pyaudio
pip install PyAudio-0.2.11-cp311-cp311-win_amd64.whl

# macOS
brew install portaudio
pip install pyaudio

# Linux
sudo apt-get install portaudio19-dev
pip install pyaudio
```

## ✗ Problema: Piper TTS Non Funziona

```
bash

# Fallback a gTTS (richiede internet)
pip install gtts

# Oppure fallback a pyttsx3 (qualità inferiore)
pip install pyttsx3
```

## ✗ Problema: Ollama Non Risponde

```
bash

# Verifica stato
ollama list
ollama serve

# Se problema persiste, riavvia
pkill ollama
ollama serve
```

## ✗ Problema: WebSocket Connection Failed

```
bash

# Verifica porta libera
netstat -an | grep 8765

# Se occupata, termina processo
# Windows: taskkill /f /im python.exe
# macOS/Linux: pkill -f websocket_server
```

## ❌ Problema: ChromaDB Import Error

**Soluzione:** Il sistema funziona senza ChromaDB (usa SQLite)

```
bash
```

```
# Opzionale - per installare ChromaDB
```

```
pip install --upgrade chromadb
```

## ⚙️ CONFIGURAZIONE AVANZATA

### 🚀 Hardware Limitato (<8GB RAM):

```
json
```

```
// config/master_config.json
{
  "voice": {
    "whisper_model": "tiny", // Invece di "base"
    "language": "it"
  },
  "llm": {
    "model": "mistral:7b",
    "max_tokens": 1024 // Invece di 2048
  }
}
```

### 🚀 Hardware Potente (>16GB RAM):

```
json
```

```
// config/master_config.json
{
  "voice": {
    "whisper_model": "medium", // Qualità superiore
    "language": "it"
  },
  "llm": {
    "model": "qwen2.5:14b", // Modello più grande
    "max_tokens": 4096
  }
}
```

### 🔧 Debug Mode:

```
json
```

```
// config/master_config.json
{
  "debug": true,
  "log_level": "DEBUG",
  "verbose_logging": true
}
```




## VERIFICA INSTALLAZIONE

### Checklist Sistema Funzionante:

#### Backend Components:

- ☐ Ollama serve attivo (localhost:11434)
- ☐ Mistral 7B scaricato (`ollama list`)
- ☐ WebSocket server attivo (localhost:8765)
- ☐ Database SQLite creato (`data/jarvis_memory.db`)

#### Voice System:

- ☐ Whisper model caricato (log: " Whisper model caricato")
- ☐ Piper TTS funzionante (log: " Piper TTS inizializzato")
- ☐ PyAudio microfono attivo (log: " PyAudio disponibile")
- ☐ Wake words detection (test: "Ehi Jarvis")

#### Frontend:

- ☐ UI si carica correttamente
- ☐ Animazioni ring funzionanti
- ☐ Metrics real-time aggiornate
- ☐ Chat input/output funziona

#### Integration Test:

- ☐ Chat testuale: scrivi "Ciao" → ricevi risposta
- ☐ Voice command: di "Ehi Jarvis, come stai?" → risposta vocale
- ☐ System metrics: CPU/RAM si aggiornano real-time

## PRIMI COMANDI DA PROVARE

### Chat Testuale:

"Ciao Jarvis"  
"Come funzioni?"  
"Che ore sono?"  
"Dimmi qualcosa di interessante"

### **Comandi Vocali:**

"Ehi Jarvis, mi senti?"  
"Jarvis, come stai?"  
"Ciao Jarvis, parlami di te"  
"OK Jarvis, che tempo fa?"

### **Comandi Sistema:**

"Jarvis, mostra le tue statistiche"  
"Jarvis, come sono le performance?"  
"Jarvis, salva la conversazione"

---

## **MONITORING E PERFORMANCE**

### **Logs Location:**

```
data/logs/  
├── jarvis.log           # Log principale  
├── voice_manager.log    # Voice system  
├── websocket_server.log # WebSocket communication  
└── llm_manager.log      # LLM responses
```

### **Performance Metrics:**

- **RAM Usage:** 4-6GB normale, <8GB sotto carico
- **CPU Usage:** 20-40% idle, 60-80% durante processing
- **Response Time:** <5s chat, <10s voice end-to-end
- **Boot Time:** 20-30s startup completa

### **Performance Tuning:**






```
bash
```

```
# Monitor risorse in tempo reale
htop # Linux/macOS
# Task Manager > Performance tab # Windows




# Log performance
tail -f data/logs/jarvis.log | grep "processata in"
```

## SICUREZZA E PRIVACY

### Garanzie Privacy:

-  Nessuna trasmissione dati online
-  Audio processing solo in memoria
-  Conversazioni salvate solo localmente
-  Nessuna API key richiesta
-  Database locale criptabile

### Network Security:

-  WebSocket solo localhost
-  Nessuna porta esposta esternamente
-  Ollama solo connessioni locali

## BACKUP E MANUTENZIONE

### File da Salvare:

```
bash

# Database conversazioni
data/jarvis_memory.db

# Configurazioni
config/master_config.json
config/user_preferences.json

# Log importanti (opzionale)
data/logs/jarvis.log
```

### Pulizia Sistema:

```
bash
```



```
# Cancella cache (ricreabile)
rm -rf data/whisper_cache/
rm -rf data/piper_models/

# Cancella log vecchi
find data/logs/ -name "*.log" -mtime +7 -delete
```

## ↑ Aggiornamenti:

```
bash

# Aggiorna requirements
pip install --upgrade -r requirements.txt

# Aggiorna modello Mistral
ollama pull mistral:7b

# Riavvia sistema
python core/websocket_server.py
```

## 🎉 SISTEMA PRONTO!

**Congratulazioni! Il tuo Jarvis AI Assistant è ora completamente operativo.**

## 🚀 Prossimi Passi:

1. **Personalizza le configurazioni** in `config/master_config.json`
2. **Esplora i comandi vocali** per familiarizzare
3. **Monitora le performance** durante l'uso quotidiano
4. **Considera l'espansione** con plugin system (v1.1)

## 📞 Supporto:

- **Log Files:** `data/logs/` per debugging
- **Configuration:** `config/` per personalizzazioni
- **Database:** `data/jarvis_memory.db` per dati conversazioni

**Il sistema è progettato per essere autosufficiente e offline-first. Goditi il tuo assistente AI personale completamente locale! 🎯**