

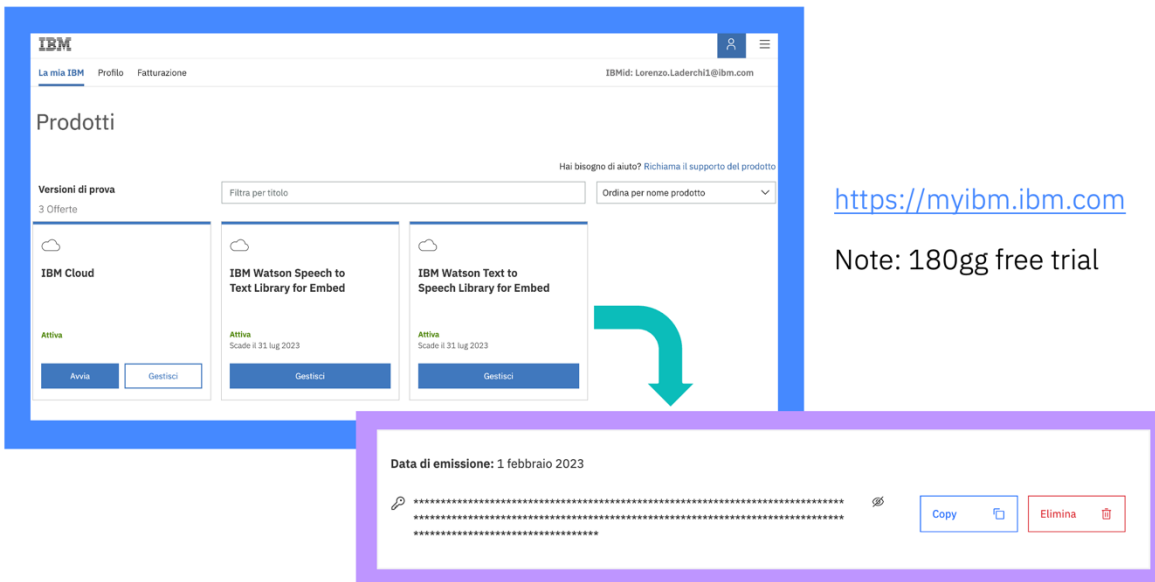
Lab 2: Watson Speech

In questo laboratorio andremo a eseguire le librerie di speech.

Per il laboratorio di oggi utilizzeremo una chiave trial già definita da noi come variabile d'ambiente, ma è possibile ottenerne una dalla durata di 180gg al seguente link:

<https://www.ibm.com/account/reg/us-en/subscribe?formid=urx-51754>

Per ottenerla è sufficiente eseguire il login con la propria IBMid (registrazione gratuita cliccando su “Crea IBMid”). La chiave ha durata 6 mesi.



The screenshot shows the IBM Cloud 'Prodotti' (Products) page. It lists three trial offers under 'Versioni di prova' (Trial versions): IBM Cloud, IBM Watson Speech to Text Library for Embed, and IBM Watson Text to Speech Library for Embed. All are marked as 'Attiva' (Active) and expire on 31 lug 2023. A green arrow points from the 'IBM Watson Speech to Text Library for Embed' offer to a separate box containing a temporary entitlement key.

<https://myibm.ibm.com>

Note: 180gg free trial

Data di emissione: 1 febbraio 2023

```
*****  
*****  
*****
```

Copy Elimina

La chiave temporanea che utilizzeremo oggi è già salvata nella variabile d'ambiente `$IBM_ENTITLEMENT_KEY`; effettuiamo il login sul private registry che contiene le immagini utilizzando il seguente comando:

```
echo $IBM_ENTITLEMENT_KEY | sudo docker login -u cp --password-stdin cp.icr.io
```

Nota: se state eseguendo il laboratorio su un personal computer, potete comunque utilizzare la chiave temporanea per effettuare il login – la trovate nella root della seguente repo: <https://github.com/LorenzOLade/Watson-Speech>

L'ambiente virtuale è unico ma ognuno avrà il suo progetto, pertanto possiamo digitare il seguente comando per creare una cartella personale:

```
mkdir tuo_cognome
```

Controllate la corretta creazione della cartella con il comando:

```
ls
```

Dovreste essere in grado di vedere la vostra cartella. Ora è il momento di accedere alla propria cartella con il comando:

```
cd tuo_cognome
```

A questo punto è necessario clonare una repository github per avere tutto il materiale a disposizione:

```
git clone https://github.com/lorenzolade/Watson-Speech
```

La repository che utilizziamo è stata clonata aggiungendo i file del laboratorio di oggi, ma potete sempre accedere all'originale (sempre aggiornata) al seguente link: <https://github.com/ibm-build-lab/Watson-Speech>

A questo punto entriamo in profondità nella repository appena scaricata entrando prima nella cartella Watson-speech e successivamente nella cartella single-container-stt con il comando:

```
cd Watson-Speech  
cd single-container-stt
```

Di default il Dockerfile clonato dalla repository github non contiene la lingua italiana; possiamo modificarlo per includerla, e successivamente procedere con la build.

Nello specifico apriamo il file con un editor di testo (noi utilizziamo nano)

```
nano Dockerfile
```

Dovremmo ritrovarci con questa videata:



```
GNU nano 2.9.8 Dockerfile

# Model images
FROM cp.icr.io/cp/ai/watson-stt-generic-models:1.5.0 as catalog
# Add additional models here
FROM cp.icr.io/cp/ai/watson-stt-en-us-multimedia:1.5.0 as en-us-multimedia
FROM cp.icr.io/cp/ai/watson-stt-fr-fr-multimedia:1.5.0 as fr-fr-multimedia

# Base image for the runtime
FROM cp.icr.io/cp/ai/watson-stt-runtime:1.5.0 AS runtime

# Configuration file directory
ENV LOCAL_DIR=chuck_var

# Environment variable used for directory where configurations are mounted
ENV CONFIG_DIR=/opt/ibm/chuck.x86_64/var

# Copy in the catalog and runtime configurations
COPY --chown=watson:0 --from=catalog catalog.json ${CONFIG_DIR}/catalog.json
COPY --chown=watson:0 ./${LOCAL_DIR}/* ${CONFIG_DIR}/

# Intermediate image to populate the model cache
FROM runtime as model_cache

# Copy model archives from model images
RUN sudo mkdir -p /models/pool2
# For each additional models, copy the line below with the model image
COPY --chown=watson:0 --from=en-us-multimedia model/* /models/pool2/
COPY --chown=watson:0 --from=fr-fr-multimedia model/* /models/pool2/

# Run script to initialize the model cache from the model archives
COPY ./prepareModels.sh .
RUN ./prepareModels.sh

# Final runtime image with models baked in
FROM runtime as release

COPY --from=model_cache ${CONFIG_DIR}/cache/ ${CONFIG_DIR}/cache/

^G Get Help  ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify
^X Exit      ^R Read File  ^_ Replace    ^U Uncut Text ^T To Spell
```

Sostituiamo la 5ª riga con la seguente, in modo da aggiungere il supporto all'italiano:

```
FROM cp.icr.io/cp/ai/watson-stt-it-it-multimedia:1.5.0 as it-it-multimedia
```

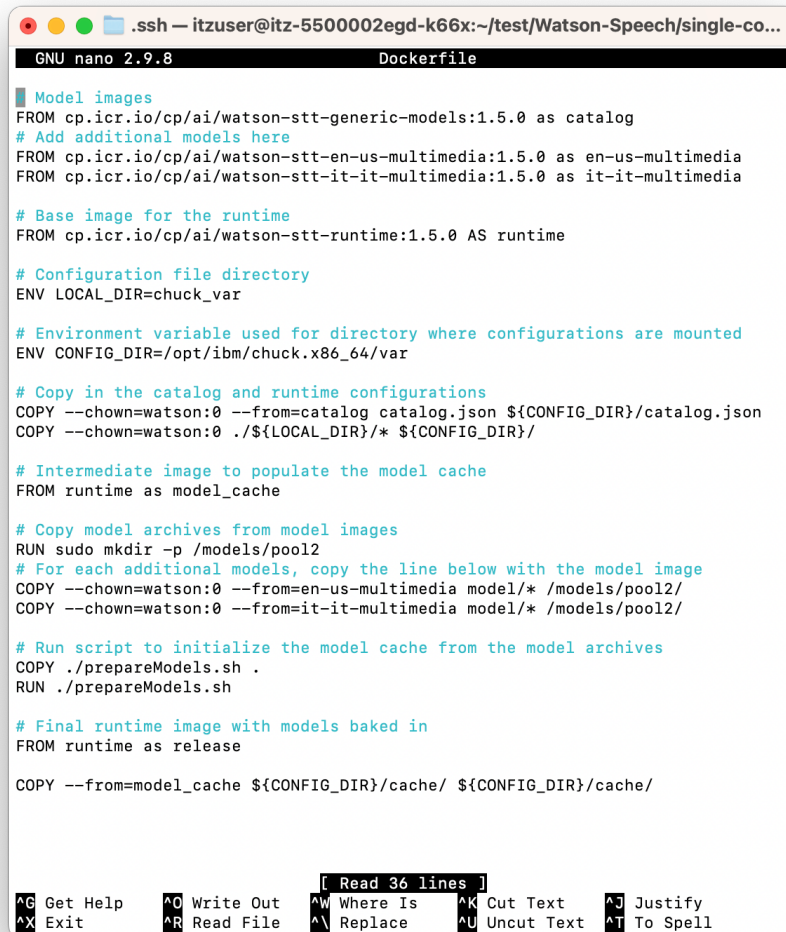
Nota: al seguente link è disponibile l'elenco completo delle lingue e modelli supportati; nello specifico notiamo che per ogni lingua sono presenti le versioni "Multimedia" e "Telephony" – i due modelli sono ottimizzati rispettivamente per file audio in alta qualità e file con frequenze troncate, tipici delle conversazioni telefoniche:

<https://www.ibm.com/docs/en/watson-libraries?topic=home-models-catalog>

Ripetiamo l'operazione con la riga corrispondente nella seguente sezione:

```
# For each additional models, copy the line below with the model image
COPY --chown=watson:0 --from=en-us-multimedia model/* /models/pool2/
COPY --chown=watson:0 --from=it-it-multimedia model/* /models/pool2/
```

Il risultato finale dovrà essere così:



```
GNU nano 2.9.8 Dockerfile

# Model images
FROM cp.icr.io/cp/ai/watson-stt-generic-models:1.5.0 as catalog
# Add additional models here
FROM cp.icr.io/cp/ai/watson-stt-en-us-multimedia:1.5.0 as en-us-multimedia
FROM cp.icr.io/cp/ai/watson-stt-it-it-multimedia:1.5.0 as it-it-multimedia

# Base image for the runtime
FROM cp.icr.io/cp/ai/watson-stt-runtime:1.5.0 AS runtime

# Configuration file directory
ENV LOCAL_DIR=chuck_var

# Environment variable used for directory where configurations are mounted
ENV CONFIG_DIR=/opt/ibm/chuck.x86_64/var

# Copy in the catalog and runtime configurations
COPY --chown=watson:0 --from=catalog catalog.json ${CONFIG_DIR}/catalog.json
COPY --chown=watson:0 ./${LOCAL_DIR}/* ${CONFIG_DIR}/

# Intermediate image to populate the model cache
FROM runtime as model_cache

# Copy model archives from model images
RUN sudo mkdir -p /models/pool2
# For each additional models, copy the line below with the model image
COPY --chown=watson:0 --from=en-us-multimedia model/* /models/pool2/
COPY --chown=watson:0 --from=it-it-multimedia model/* /models/pool2/

# Run script to initialize the model cache from the model archives
COPY ./prepareModels.sh .
RUN ./prepareModels.sh

# Final runtime image with models baked in
FROM runtime as release

COPY --from=model_cache ${CONFIG_DIR}/cache/ ${CONFIG_DIR}/cache/

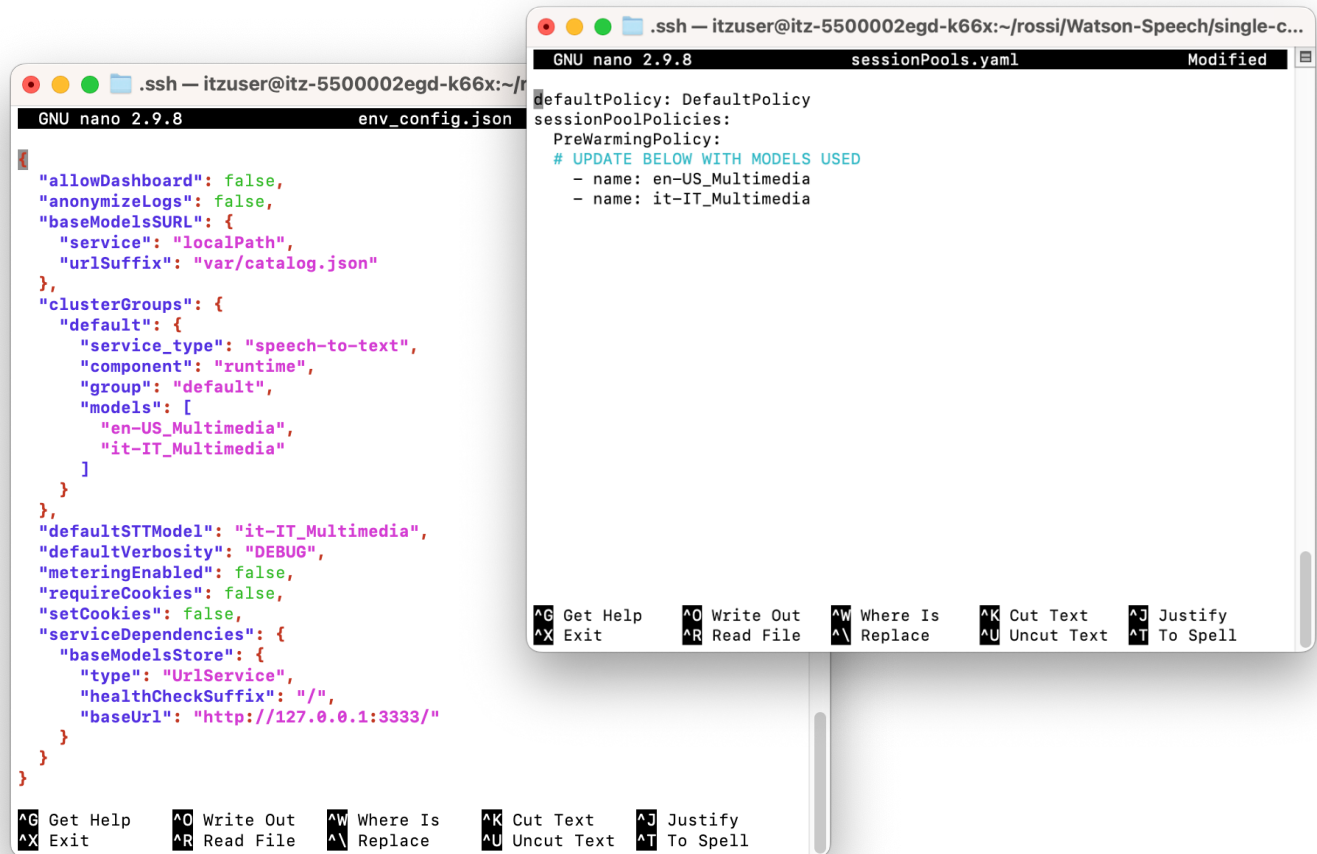
[ Read 36 lines ]
^G Get Help      ^O Write Out    ^W Where Is     ^K Cut Text     ^J Justify
^X Exit          ^R Read File    ^\ Replace      ^U Uncut Text   ^T To Spell
```

Salviamo il Dockerfile appena modificato (con nano: ctr-x, yes, enter).

Avendo modificato il Dockerfile di default, è necessario aggiornare i file **env_config.json** e **sessionPools.yaml** all'interno della directory **chuck_var** in modo da rispecchiare le stesse modifiche di lingua fatte nel Dockerfile (sostituzione della lingua francese con quella italiana).

Nel file **env_config.json** è anche possibile modificare il parametro **defaultSTTModel** per impostare la lingua di default, ad esempio con **it-IT_Multimedia**

Dovremmo avere questa situazione:



```
GNU nano 2.9.8 env_config.json
{
  "allowDashboard": false,
  "anonymizeLogs": false,
  "baseModelsSURL": {
    "service": "localPath",
    "urlSuffix": "var/catalog.json"
  },
  "clusterGroups": {
    "default": {
      "service_type": "speech-to-text",
      "component": "runtime",
      "group": "default",
      "models": [
        "en-US_Multimedia",
        "it-IT_Multimedia"
      ]
    }
  },
  "defaultSTTModel": "it-IT_Multimedia",
  "defaultVerbosity": "DEBUG",
  "meteringEnabled": false,
  "requireCookies": false,
  "setCookies": false,
  "serviceDependencies": {
    "baseModelsStore": {
      "type": "UrlService",
      "healthCheckSuffix": "/",
      "baseUrl": "http://127.0.0.1:3333/"
    }
  }
}

GNU nano 2.9.8 sessionPools.yaml
defaultPolicy: DefaultPolicy
sessionPoolPolicies:
  PreWarmingPolicy:
    # UPDATE BELOW WITH MODELS USED
    - name: en-US_Multimedia
    - name: it-IT_Multimedia

^G Get Help  ^O Write Out  ^W Where Is  ^K Cut Text  ^J Justify
^X Exit      ^R Read File  ^\ Replace   ^U Uncut Text ^T To Spell
```

È ora il momento di costruire l'immagine appena descritta nel Dockerfile; per farlo riposizioniamoci sulla directory superiore e lanciamo il comando docker build:

```
cd ..
sudo docker build . -t stt-tuo_cognome
```

Verifichiamo il successo dell'operazione controllando l'esistenza della nostra immagine:

```
sudo docker images
```

Siamo finalmente in grado di lanciare il nostro container personalizzato; utilizziamo il seguente comando:

```
sudo docker run --rm -it -d --env ACCEPT_LICENSE=true -p 1080:1080 stt-tuo_cognome
```

- rm il container è effimero, viene rimosso al primo stop
- it sessione interattiva
- d detached mode: vogliamo che il servizio giri in background
- p mappiamo la porta dell'host Docker sulla porta interna del servizio

Verifichiamo che il container stia girando:

```
docker ps
```

Testiamo il servizio facendogli elencare i modelli disponibili:

```
curl "http://localhost:1080/speech-to-text/api/v1/models"
```

Dovremmo ottenere qualcosa di simile:

```
{
  "models": [
    {
      "name": "en-US_Multimedia",
      "rate": 16000,
      "language": "en-US",
      "description": "US English multimedia model for broadband audio (16kHz or more)",
      "supported_features": {
        "custom_acoustic_model": false,
        "custom_language_model": true,
        "low_latency": true,
        "speaker_labels": true
      },
      "url": "http://localhost:1080/speech-to-text/api/v1/models/en-US_Multimedia"
    },
    {
      "name": "fr-FR_Multimedia",
      "rate": 16000,
      "language": "fr-FR",
      "description": "French multimedia model for broadband audio (16kHz or more)",
      "supported_features": {
        "custom_acoustic_model": false,
        "custom_language_model": true,
        "low_latency": true,
        "speaker_labels": true
      },
      "url": "http://localhost:1080/speech-to-text/api/v1/models/fr-FR_Multimedia"
    }
  ]
}
```

Se abbiamo lasciato la lingua inglese come predefinita, testiamo il servizio con il sample `audio en-quote-1` (contenuto nella cartella `sample_dataset`) senza specificare la lingua nella costruzione della chiamata:

```
curl "http://localhost:1080/speech-to-text/api/v1/recognize" \
  --header "Content-Type: audio/wav" \
  --data-binary @sample_dataset/en-quote-1.wav
```

Otterremo una risposta json di questo tipo:

```
[[ituser@itz-5500002egd-k66x single-container-stt]$ curl "http://localhost:1080/speech-to-text/api/v1/recognize" \
> --header "Content-Type: audio/wav" \
> --data-binary @sample_dataset/en-quote-1.wav
{
  "result_index": 0,
  "results": [
    {
      "final": true,
      "alternatives": [
        {
          "transcript": "the day will come when there will be no battlefields but markets opening to commerce and minds opening to ideas ",
          "confidence": 0.94
        }
      ]
    }
  ]
}
```

Testiamo infine la lingua italiana con la seguente chiamata:

```
curl "http://localhost:1080/speech-to-text/api/v1/recognize?model=it-IT_Multimedia" \
  --header "Content-Type: audio/wav" \
  --data-binary @sample_dataset/it-quote-1.wav
```