Setting up the Environment

Welcome to the Rasa Certification Workshop - we're glad to have you joining us! Before we begin the workshop, we have a few things to set up to make sure you're ready to go on day 1. In this session, we'll set up the environment, by installing all required tools and dependencies and downloading the starter project.

Use this handout to follow along with the video walkthrough.

I. Pre-workshop checklist:

Before you begin, you'll need:

- ❑ IDE or Text editor
- ❑ Python 3.6 or 3.7

Setting up the project:

- ❑ Virtual environment
- ❑ Financial Demo bot
  - ❑ Fork the repo
  - ❑ Clone a local copy
  - ❑ Install dependencies (including Rasa Open Source)
- ❑ Run the bot!
- ❑ Telegram account
- ❑ Install ngrok

II. Setting up the Virtual Environment

Why set up a virtual environment? Virtual environments let you scope Python packages to a project directory, instead of installing the package on the system (global) level. This means you can use different versions of a package for different projects and helps to prevent conflicts.

We're using venv in this demo to create the virtual environment, because it's built into the Python standard library. This is the process we recommend for Mac and Linux users.

Note: If you're using **Windows**, we recommend using Anaconda to set up your virtual environment instead of venv. You can watch a video walkthrough of the Windows installation process here. The important parts to note are installing ujson, tensorflow, and the Visual Studio C++ build tools.

For the workshop, we'll be working with an existing Rasa project instead of creating a new one using `rasa init`, so be sure to return to the workshop setup video to see how to clone the project and install its dependencies.

Steps (Mac/Linux):

A. Create a new directory for the workshop project
    a. mkdir rasa-workshop
B. Navigate into the directory
    a. cd rasa-workshop
C. Create the virtual environment
    a. python3 -m venv ./venv
D. Activate the virtual environment
    a. source ./venv/bin/activate

III. Clone the starter project

For this workshop, we're using the Financial Services demo bot, an open source banking assistant.

Steps:

A. Clone the repo to create a local copy on your computer:
    a. git clone https://github.com/RasaHQ/financial-demo.git

IV. Git Cheatsheet

We'll be using Git throughout the workshop to switch between versions of the project at different stages of development. Here are a few common commands we'll be using:

git remote -v	Show remote repositories
git fetch <remote name, e.g. origin>	Download new changes from remote repo
git branch -a	List all branches
git checkout <existing branch name>	Switch to a different branch
git status	Track uncommitted changes
git stash	Temporarily shelve uncommitted changes (so you can switch to a different branch)

V. Install Dependencies

To install Rasa Open Source (along with the other packages required by the Financial Demo assistant), we'll use the requirements.txt included in the repository.

Steps:

A. Navigate into the Financial Services demo assistant folder
    a. cd financial-demo
B. Install dependencies
    a. pip install -r requirements.txt

VI. Download the SpaCy Language Model

Steps:

A. Download the model:
    a. python3 -m spacy download en_core_web_md
B. Link the model:
    a. python -m spacy link en_core_web_md en

VII. Set up Duckling

Duckling is a pre-trained entity extractor that we'll be using to extract numbers, times, and amounts of money. It runs outside of Rasa Open Source, as a separate service.

You can run Duckling using Docker, but you can also run it on an external server, which is the method we'll be using. We've already set up a Duckling server at this address:

http://duckling.rasa.com:8000

Steps:

A. Open the config.yml file and locate this line in the Duckling config:
   `url: http://localhost:8000`
B. Change the URL to the following:
   `url: http://duckling.rasa.com:8000`

VIII. Run the Assistant

Before we can start talking to the assistant, we need to train the model and start the servers.

Steps:

A. Train the model
    a. rasa train
B. Once the model has finished training, start the Rasa Open Source server. This will also start a session to chat with the assistant on the command line.
    a. rasa shell
C. Open a second terminal window and run this command to start the action server. Note, **this bot runs the action server on 5056**, which isn't the default port. *Tip: be sure your virtual environment is active. You may need to run the source ./venv/bin/activate command to start the virtual environment before starting the action server.*
    a. rasa run actions --port 5056
D. Talk to the bot! Check the README to see the questions the assistant can recognize.
E. To stop the Rasa Open Source server and the chat session, type `/stop`
F. To stop the action server, you can close the terminal window where it's running.

IX. Create a Telegram account

Steps:

A. Download Telegram for desktop or mobile
B. Create your account
C. Log in

X. Install ngrok

Follow the instructions here to download and install ngrok: https://ngrok.com/download

Note: If you're on a Mac, the easiest way to install ngrok is using homebrew. This method installs globally, so you can run the ngrok command from any directory.

```
brew cask install ngrok
```

If you installed ngrok by downloading the file directly (not using homebrew), you'll need to provide the path to where you unzipped the ngrok executable (relative to your project folder) when you start ngrok (e.g. `./ngrok http 5005`). Alternatively, you can add the ngrok executable to your $PATH so you can run the ngrok command globally.

To test ngrok, make sure you have a process running on localhost (e.g. the `rasa shell` command starts Rasa Open Source on port 5005). Open a new terminal window and run `ngrok http 5005`. You should see the tunnel URLs generated by ngrok. If you get command not found but you've

installed ngrok, provide the path to the executable in your start command, e.g. `./ngrok http 5005`.