Remote Computing using Python

Haris Jabbar
haris.jabbar@tum.de

May 2018

1 Introduction

With the advent of cloud computing platforms and access to remote high performance compute resources, it is imperative that a Python developer can make use of these resources while developing locally. A naive method could be to write a python script locally and then transfer the file to remote machine; but this method is sub-optimal. A much more productive method is to run a python interpreter on the remote machine (either in cloud e.g AWS or at a compute cluster like at LRZ or at a research chair) and be able to access it on your local laptop/desktop. This guide aims to demystify this remote computing paradigm for python development. It is divided into following sections:

- Setup
- SSH Port Forwarding (aka SSH Tunneling)
- Jupyter Notebooks
- Step by Step Guide

2 Setup

This guide will use three computers A, B and C. The IP addresses are as follows :

- Computer A : 192.188.0.1 (local client e.g. your laptop/desktop)
- Computer B : 192.168.0.2 (SSH Server)
- Computer C : 192.168.0.3 (remote server in cloud e.g. with GPUs etc.)

3 SSH Port Forwarding

Accessing remote machines over the internet (or even intranet) is inherently a security risk. Hence adequate precautions must be taken to secure the connection. SSH Tunneling is one of the most widely used methods for such secure connections. Port forwarding is a mechanism when a port on a computer 'A' is mapped to a port on a remote computer 'C'. This allows the user on 'A' to use remote port locally and everything s/he does is executed on the remote port on computer 'C'.

There are three types of SSH port forwarding:

- Local port forwarding - connections from an SSH client (A) are forwarded, via the SSH server (B), to a destination server (C).
- Remote port forwarding - connections from an SSH server (C) are forwarded, via the SSH client (A), to a destination server (B)
- Dynamic port forwarding - connections from various programs are forwarded, via the SSH client to an SSH server, and finally to several destination servers. (We will not cover this)

3.1 The Intuition

Port forwarding is a very useful tool but it's important to understand the difference between the two mechanisms.

Local Port Forwarding

Suppose you (A) are on a private network which blocks access to *Facebook* (C) but you have SSH access to another machine (B) outside your private network that has access to *Facebook*. Then you could use local port forwarding like so

$ ssh -L 9000:facebook.com:80 user@192.168.0.2

Now open your browser and go to http://localhost:9000 and you will be (magically) transported to facebook.com

Another more relevant example is the current use case of remote python development on a server (B). In this scenario the jupyter notebook runs as a local server (C) at port 8888 (configurable). This jupyter server (C) is available only to localhost and not remotely. Here both B and C are same physical machines. B refers to the ssh server and C refers to the jupyter server; both running on a single machine. To connect to this server:port from your machine (A), we use the following command

$ ssh -L 9000:localhost:8888 user@192.168.0.2

Now when you open http://localhost:9000, you are taken to the jupyter server, running locally on 192.168.0.2

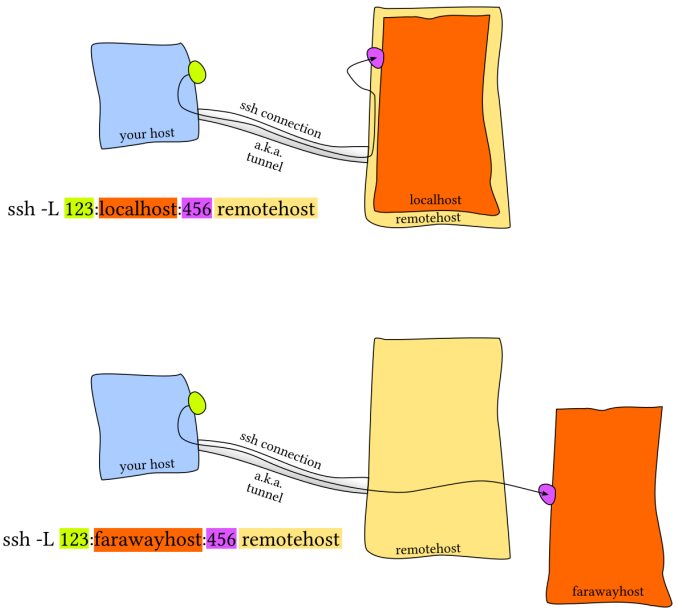

Figure 1: Local Port Forwarding

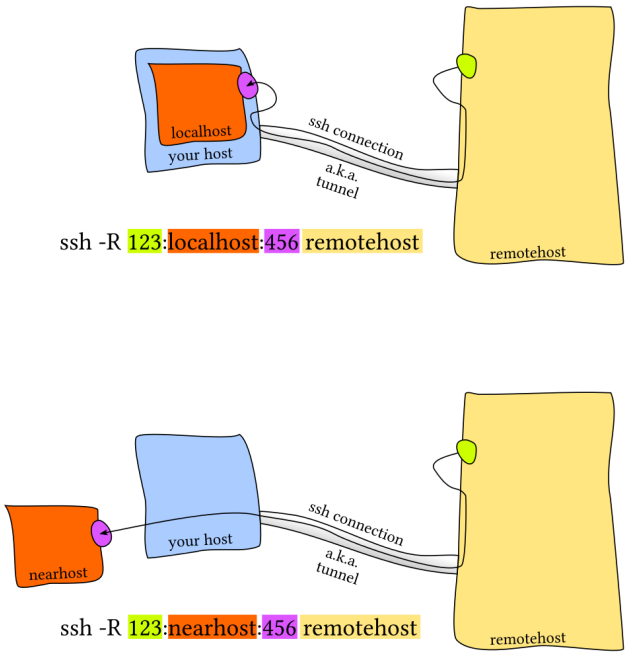

Figure 2: Remote Port Forwarding

Remote Port Forwarding

Suppose you are training your neural network and want to see/share its progress on the go. There are many applications (e.g tensorboard and visdom) that run a local python server which you can access. Ideally you will need a fixed IP address on this server machine that you can share. But if you don't have one, you can use remote port forwarding to fix this problem.

In this setup, you will need access to a computer (B) with fixed IP address and for which you have ssh access. The computer on which you are training the network and running the tensorboard/visdom server will be C. Suppose your tensorboard runs on port 6006. Then you will run the following command on C.

$ ssh -R 9000:localhost:6006 user@192.168.0.2

Now anyone opening http://192.168.0.2:9000 will be forwarded to machine C, port 6006 where your tensorboard is running.

4 Jupyter Notebooks

Jupter Notebook is an open source web application that allows running remote python instances over the web. The interface allows seamless integration of code, text, and visualizations.

JupyterLab (http://jupyterlab.readthedocs.io) is the next version of Jupyter Notebooks. Although still in beta, but it's stable for daily use and has all the features (plus many more) of jupyter notebooks.

For all the steps below, you can use either jupyter-notebook or jupyter-lab; although Jupyter Lab will give you a far richer experience.

5 Step by Step Guide

Now we will give step by step instruction to use a jupyter notebook with remote machine. The idea is that you (user: s007) have access to a GPU machine (e.g 'atbeetz21.informatik.tu-muenchen.de')

In this setup, we will use local port forwarding. Your laptop is ssh client (A). The remote GPU machine acts as both (B) and (C). More specifically, the ssh server is B and the jupyter notebook server is C.

The ssh command will be (See figure 3):

$ ssh -p 58022 -L 9000:localhost:8888 s007@atbeetz21.informatik.tu-muenchen.de

You can use different port number instead of 9000. The port 8888 is the default jupyter notebook server port. We are assuming that no other jupyter server is running on GPU machine. If that is not the case, jupyter server will allocate a different port to every subsequent run (See Figure 4). We can also specify the port to be used by jupyter server.

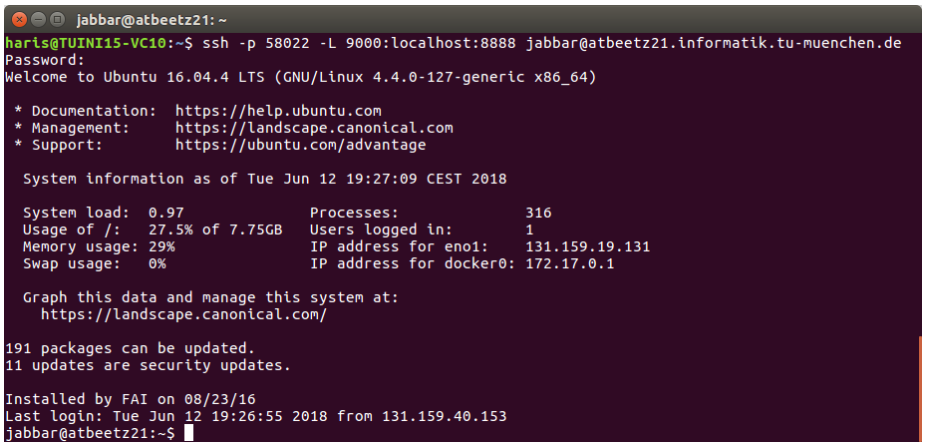

Figure 3: SSH Login with Port Forwarding

The flag '-p 58022' is the SSH server port instead of the default port 22. This is fixed by admin and has no bearing on port forwarding; other than specifying here.

Assuming the ssh was success, you will be logged in to the remote machine. It is always a good practice to see if anyone else is using the GPU(s) on the machine. This can be done by

$nvidia-smi

This will give you information about the NVIDIA driver, the available GPUs and some other useful statistics. If someone is already using the GPU, you will typically see information about the running process as well. Other ways to see are if GPU utilization is non zero.

Assuming no one is using the GPU, you can go ahead and start the jupyter notebook by

$jupyter-notebook –no-browser –port 8888

The first flag tells the server not to launch the browser automatically. The second flag is where you can specify a port other than default 8888.

Above command will give you information like shown in Figure 4. You can either click on the link or use copy/paste to open it in your own local browser. Please note that in our example we mapped port 8888 to port 9000, so here we will have to use 9000 instead of 8888.

If everything goes well, you will see the familiar jupyter webpage in your local browser.

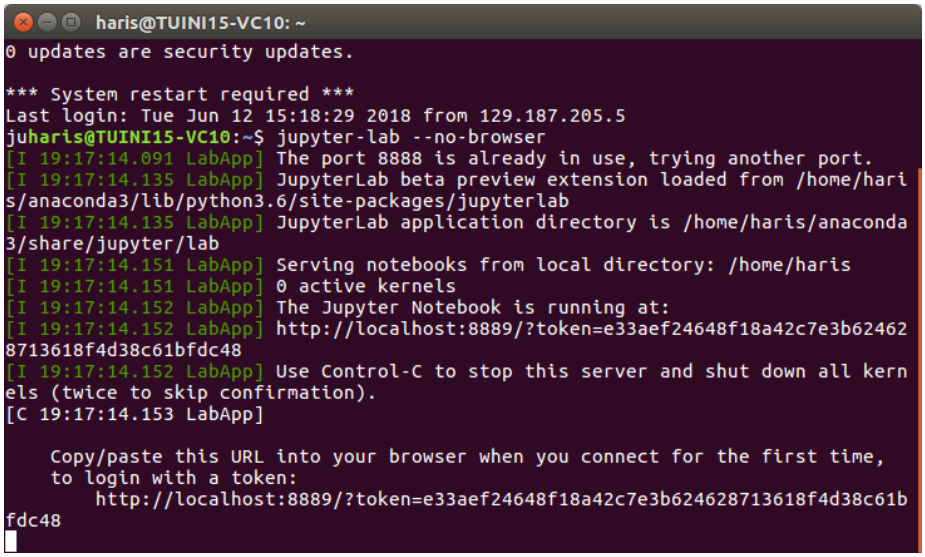

Figure 4: Running Jupyter Notebook