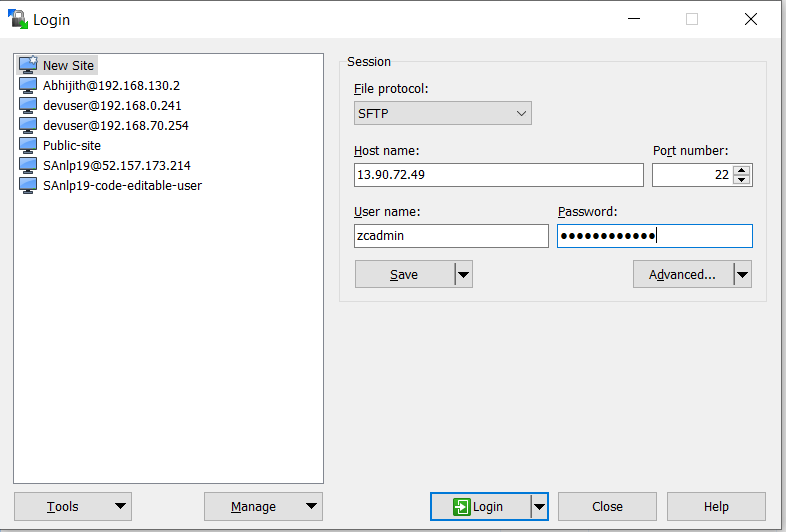
# Training object detection model

1. Open WinSCP
2. Add new site



13.90.72.49

​

Username: zcadmin

Password: \*\*\*\*\*\*\*\*\*\*

​

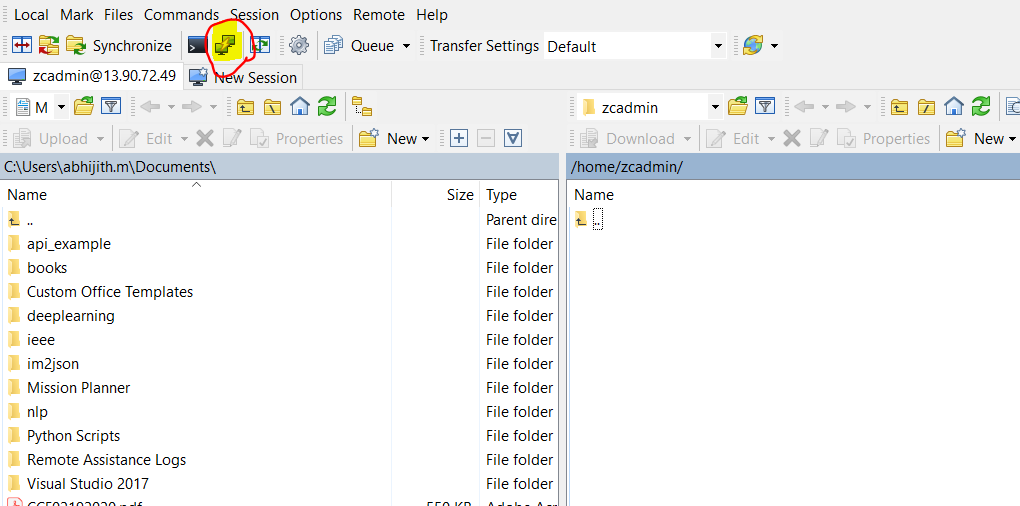
Config: Standard NC12\_Promo (12 vcpus, 112 GiB memory)

# Instalation

Tutorial: <https://www.youtube.com/watch?v=wdufj-pjE5c&list=PLoY9f-5DCBlkvkvDjwa5Dn4WXjhTA5K7v>

Installation written document: <https://gilberttanner.com/blog/installing-the-tensorflow-object-detection-api>

1. Open putty



1. Python version:- 3.6.9
2. Create python [virtual environment](https://stackoverflow.com/questions/29934032/virtualenv-python-3-ubuntu-14-04-64-bit)
   1. > pip3 install virtualenv
   2. > python3 -m virtualenv object\_detection\_env
   3. > source object\_detection\_env/bin/activate
   4. If it is successful then the following output will get



1. pip3 install tensorflow==1.14
2. pip3 install Pillow==6.1.0
3. pip3 install contextlib2==0.5.5
4. pip3 install Cython==0.29.14
5. pip3 install lxml==4.4.0
6. pip3 install matplotlib==3.1.1
7. pip3 install pandas

## Download Tensorflow object detection model

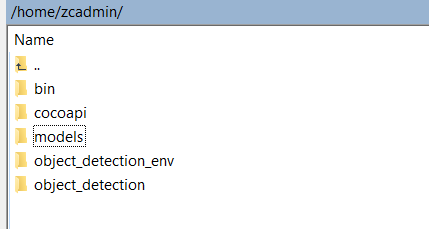
1. sudo apt install git
2. git clone [https://github.com/tensorflow/models --single-branch --branch r1.13.0](https://github.com/tensorflow/models%20--single-branch%20--branch%20r1.13.0)

## Install the COCO API

1. git clone <https://github.com/cocodataset/cocoapi.git>
2. cd cocoapi/PythonAPI
3. make
4. cp -r pycocotools <path\_to\_tensorflow>/models/research/
   1. eg:> cp -r pycocotools /home/zcadmin/models/research/

## Protobuf Installation/Compilation

1. [Download protobuf](https://github.com/protocolbuffers/protobuf/releases/download/v3.12.3/protoc-3.12.3-linux-x86_64.zip) for linux – you can download this file from windows machine using google chrome.
2. After downloading the file extract the folder and move “bin” folder into linux server(location: /home/zcadmin/)
3. Current folder structure



1. Move cmd location to /home/zcadmin/model/research/
2. Create a python file “use\_protobuf.py”in model/research folder

Note: to make folder as wr permission:

In putty: >sudo chown ubuntu:ubuntu .

* + - ls –ld
* this will show output like this

To make all folder permission as ububtu:

* + - sudo chown ubuntu:ubuntu \* -R
    - ls -l

drwxr-xr-x 70 ubuntu ubuntu 4096 Jul 19 06:40 .

import os

import sys

args = sys.argv

directory = args[1]

protoc\_path = args[2]

for file in os.listdir(directory):

    if file.endswith(".proto"):

        os.system(protoc\_path+" "+directory+"/"+file+" --python\_out=.")

1. Run this cmd to extract proto file
2. > python3 use\_protobuf.py object\_detection/protos ../../bin/protoc

## Export python path

* export PYTHONPATH=$PYTHONPATH:/home/zcadmin/models/research
* export PYTHONPATH=$PYTHONPATH:/home/zcadmin/models/research/object\_detection
* export PYTHONPATH=$PYTHONPATH:/home/zcadmin/models/research/slim

## Install object detection api

1. ensure current cmd is in model/research folder

* python3 setup.py build
* python3 setup.py install

1. Now the installation is completed.
2. You can check using the following command in python3 cmd

* import object\_detection

# Training

Written documentation: <https://gilberttanner.com/blog/creating-your-own-objectdetector>

1. open models/research/object\_detection folder
2. create a new file – “xml\_to\_csv.py”

import os

import glob

import pandas as pd

import xml.etree.ElementTree as ET

def xml\_to\_csv(path):

    xml\_list = []

    for xml\_file in glob.glob(path + '/\*.xml'):

        tree = ET.parse(xml\_file)

        root = tree.getroot()

        for member in root.findall('object'):

            value = (root.find('filename').text,

                     int(root.find('size')[0].text),

                     int(root.find('size')[1].text),

                     member[0].text,

                     int(member[4][0].text),

                     int(member[4][1].text),

                     int(member[4][2].text),

                     int(member[4][3].text)

                     )

            xml\_list.append(value)

    column\_name = ['filename', 'width', 'height', 'class', 'xmin', 'ymin', 'xmax', 'ymax']

    xml\_df = pd.DataFrame(xml\_list, columns=column\_name)

    return xml\_df

def main():

    for folder in ['train', 'test']:

        image\_path = os.path.join(os.getcwd(), ('images/' + folder))

        xml\_df = xml\_to\_csv(image\_path)

        xml\_df.to\_csv(('images/'+folder+'\_labels.csv'), index=None)

    print('Successfully converted xml to csv.')

main()

1. Create a new folder “images”
2. Inside images folder create train and test folder
3. Paste training and testing data into this folder

* Python3 xml\_to\_csv.py

1. Create python file “generate\_tfrecords.py”

"""

Usage:

  # From tensorflow/models/

  # Create train data:

  python generate\_tfrecord.py --csv\_input=data/train\_labels.csv  --output\_path=train.record

  # Create test data:

  python generate\_tfrecord.py --csv\_input=data/test\_labels.csv  --output\_path=test.record

"""

from \_\_future\_\_ import division

from \_\_future\_\_ import print\_function

from \_\_future\_\_ import absolute\_import

import os

import io

import pandas as pd

import tensorflow as tf

from PIL import Image

from object\_detection.utils import dataset\_util

from collections import namedtuple, OrderedDict

flags = tf.app.flags

flags.DEFINE\_string('csv\_input', '', 'Path to the CSV input')

flags.DEFINE\_string('output\_path', '', 'Path to output TFRecord')

flags.DEFINE\_string('image\_dir', '', 'Path to images')

FLAGS = flags.FLAGS

# TO-DO replace this with label map

def class\_text\_to\_int(row\_label):

    if row\_label == 'label':

        return 1

    elif row\_label == 'logo':

        return 2

    elif row\_label == 'table':

        return 3

    elif row\_label == 'checkbox':

        return 4

    else:

        return None

def split(df, group):

    data = namedtuple('data', ['filename', 'object'])

    gb = df.groupby(group)

    return [data(filename, gb.get\_group(x)) for filename, x in zip(gb.groups.keys(), gb.groups)]

def create\_tf\_example(group, path):

    with tf.gfile.GFile(os.path.join(path, '{}'.format(group.filename)), 'rb') as fid:

        encoded\_jpg = fid.read()

    encoded\_jpg\_io = io.BytesIO(encoded\_jpg)

    image = Image.open(encoded\_jpg\_io)

    width, height = image.size

    filename = group.filename.encode('utf8')

    image\_format = b'jpg'

    xmins = []

    xmaxs = []

    ymins = []

    ymaxs = []

    classes\_text = []

    classes = []

    for index, row in group.object.iterrows():

        xmins.append(row['xmin'] / width)

        xmaxs.append(row['xmax'] / width)

        ymins.append(row['ymin'] / height)

        ymaxs.append(row['ymax'] / height)

        classes\_text.append(row['class'].encode('utf8'))

        classes.append(class\_text\_to\_int(row['class']))

    tf\_example = tf.train.Example(features=tf.train.Features(feature={

        'image/height': dataset\_util.int64\_feature(height),

        'image/width': dataset\_util.int64\_feature(width),

        'image/filename': dataset\_util.bytes\_feature(filename),

        'image/source\_id': dataset\_util.bytes\_feature(filename),

        'image/encoded': dataset\_util.bytes\_feature(encoded\_jpg),

        'image/format': dataset\_util.bytes\_feature(image\_format),

        'image/object/bbox/xmin': dataset\_util.float\_list\_feature(xmins),

        'image/object/bbox/xmax': dataset\_util.float\_list\_feature(xmaxs),

        'image/object/bbox/ymin': dataset\_util.float\_list\_feature(ymins),

        'image/object/bbox/ymax': dataset\_util.float\_list\_feature(ymaxs),

        'image/object/class/text': dataset\_util.bytes\_list\_feature(classes\_text),

        'image/object/class/label': dataset\_util.int64\_list\_feature(classes),

    }))

    return tf\_example

def main(\_):

    writer = tf.python\_io.TFRecordWriter(FLAGS.output\_path)

    path = os.path.join(FLAGS.image\_dir)

    examples = pd.read\_csv(FLAGS.csv\_input)

    grouped = split(examples, 'filename')

    for group in grouped:

        tf\_example = create\_tf\_example(group, path)

        writer.write(tf\_example.SerializeToString())

    writer.close()

    output\_path = os.path.join(os.getcwd(), FLAGS.output\_path)

    print('Successfully created the TFRecords: {}'.format(output\_path))

if \_\_name\_\_ == '\_\_main\_\_':

    tf.app.run()

* python3 generate\_tfrecord.py --csv\_input=images/train\_labels.csv --image\_dir=images/train --output\_path=train.record
* python3 generate\_tfrecord.py --csv\_input=images/test\_labels.csv --image\_dir=images/test --output\_path=test.record
* create new directory – training
* create new file inside training folder – “labelmap.pbtxt”

item {

id: 1

name: 'label'

}

item {

id: 2

name: 'logo'

}

item {

id: 3

name: 'table'

}

item {

id: 4

name: 'checkbox'

}

## Edit config file

1. copy config file in sample/ faster\_rcnn\_inception\_v2\_pets.config to training folder
2. open training/ faster\_rcnn\_inception\_v2\_pets.config file
3. download this [model](http://download.tensorflow.org/models/object_detection/faster_rcnn_inception_v2_coco_2018_01_28.tar.gz) file
4. edit config file using this [documentation](https://gilberttanner.com/blog/creating-your-own-objectdetector) and run training