Setup & Python Basics

Data Science Fall 2016

Overview

- Setup working environment
- Review of Python basics
- Q & A
- Quiz (10 minutes)

VM Setup: pros & cons

- Why you may want to follow these steps
  - If you don’t have a Linux working environment, then it provides you one.
  - Everyone is working on the same environment, issues can be resolved easier.
  - We pre-install most required software packages in an virtual machine image.

- Potential issues
  - Running a VM is usually slower in also everything. We suggest you use a machine with at least i3 dual core, 4gb ram or equivalent. Exact requirement varies.

VM Setup: Download & Install

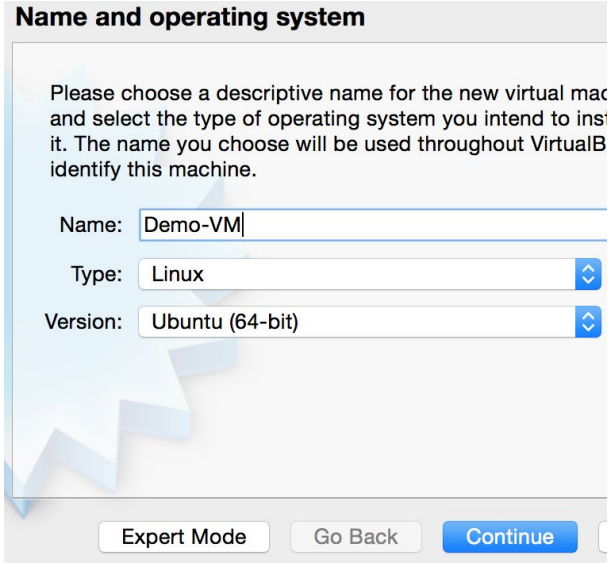
Download VirtualBox (~100M): https://www.virtualbox.org

Download VM Image (~2GB):
https://drive.google.com/open?id=0Byru7AR76OHxcXJKMlItOEdIOFU

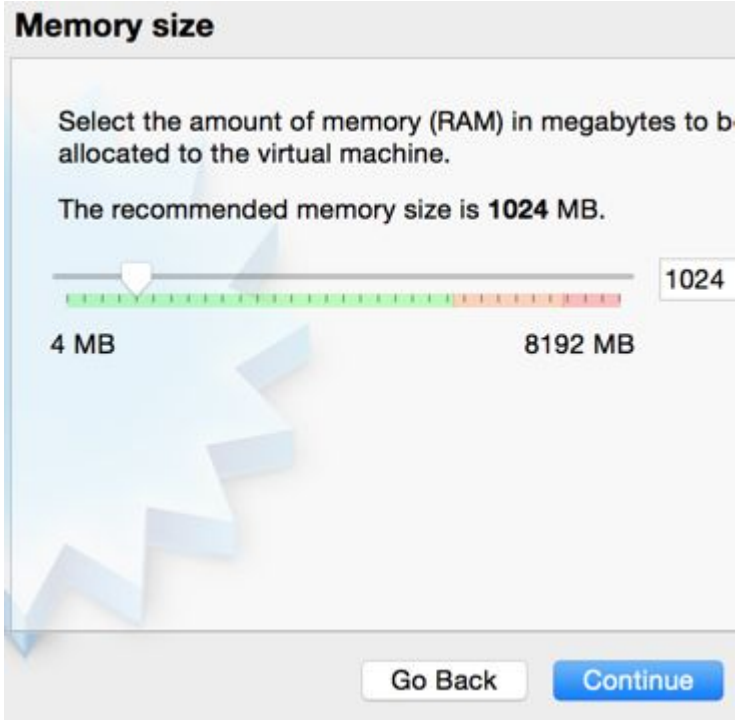
Unzip VM Image: you get a .**vdi** file.

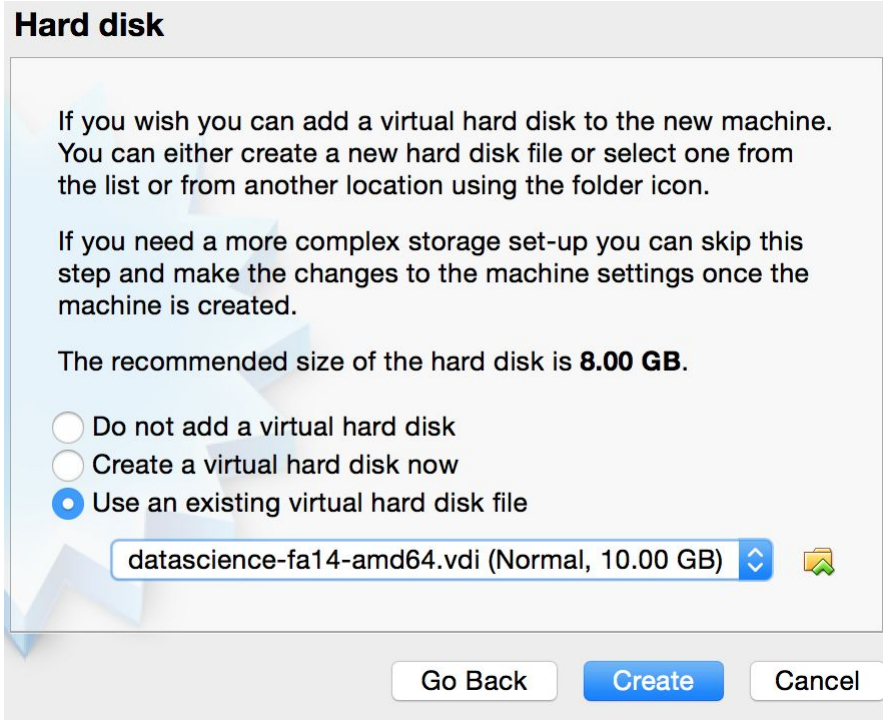
Install VirtualBox: follow the installer.

VM Setup: Create VM

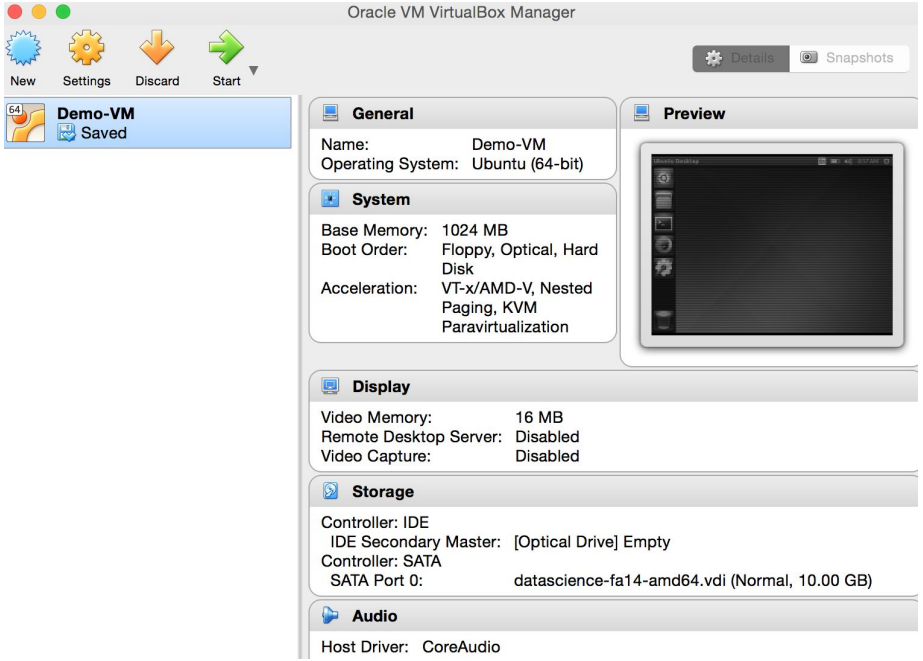


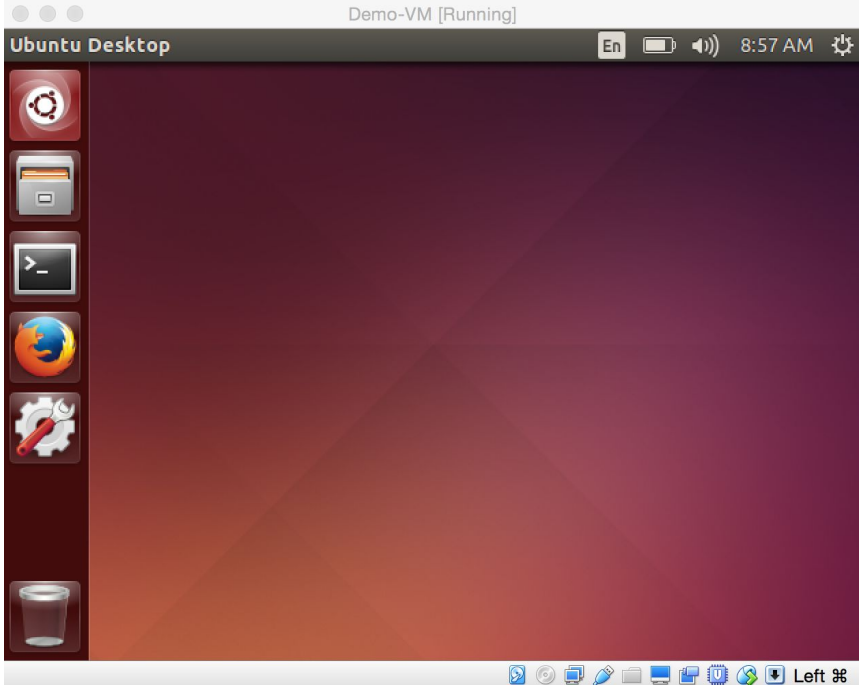
VM Setup: Create VM (continued)





VM Setup: Create VM (continued)





Python: Why you need it as a Data Scientist?

- For these reasons:
  - Rich Ecosystem: lots of open source projects (e.g. nltk, scikit, tensorflow, ...)
  - Widely accessible (and free): comes with Linux/Unix systems.
  - Easy: you can write codes very FAST!
  - Quite efficient: Cpython implemented with C/C++.

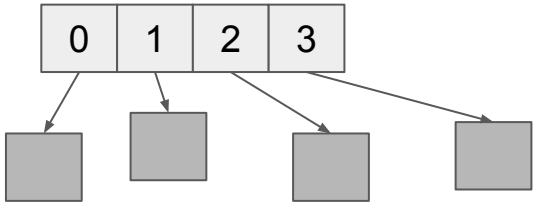
Python Basics: A hello world program!

```cpp
#include <iostream>

using namespace std;

int main()
{
    cout << "Hello world!" << endl;
    return 0;
}
```

```python
# Inline comments starts with hash (pound) symbol.
"""
Example comments of multiple lines.
How to run: python helloworld.py
"""
import os

if __name__ == "__main__":
    # This block of codes only execute as "main"
    # Python code blocks are recognized by indenture!
    cwd = os.getcwd()
    print cwd + ": Hello World!"
```

Python Basics: Lists

- CPython Implementation
  - Array of pointers to objects (not C/C++ list!)



- Creation
  - L1 = [1, 2]
  - L2 = [1] * 2 # [1, 1]
  - L3 = L1 + L2 # [1, 2, 1, 1]
  - L4 = range(0, 3) # [0, 1, 2]
- Mutation
  - L1.append(3) # [1, 2, 3] -- O(1)
  - L1[0] = 0 # [0, 2, 3] -- O(1)
  - L1.remove(2) # [0, 2] -- O(n)

Python Basics: Dictionaries

- Usage
  - Store key->value pairs.
- CPython Implementation
  - Hashtables
- Creation & Mutation
  - D1 = {“key1”: “val1”, 2: “val2”, 3: [1,2,3,4,5]}
  - D2 = {i:i*i for i in range(3)} # {0: 0, 1: 1, 2: 4}
  - D2[3] = 9 # {0: 0, 1: 1, 2: 4, 3: 9}
  - del D2[3] # {0: 0, 1: 1, 2: 4}
  - D1 + D2 = illegal operation!
- Access all key-value pairs.
  - for key, val in **D1.items()**:

    print key, val

Python Basics: Sets

- Usage
  - Store distinct set of elements.
- CPython Implementation
  - Hashtables (like Dictionary, without value)
- Creation & Mutation
  - D1 = {“key1”, 2, 3} # vs. dictionary {“key1”: “val1”, 2: “val2”, 3: [1,2,3,4,5]}

Python Basics: Tuples

- Creation
  - T1 = (1, ‘2’)
  - T2 = (3,)
  - T3 = T1 + T2 # (1, ‘2’, 3)
- Mutation
  - No Mutation is allowed for tuples!
- Tuples as keys
  - Dictionary[(1,2)]
- Comparison against lists
  - List is not hashable: Dictionary[[1,2]] -- cannot use list as keys of a dictionary!
  - Tuple is not mutable: T[0] = 0 -- illegal: cannot mutate a tuple in any way!
  - Tuple is slightly more efficient than list (fewer statuses to store):
    - L = [1,2,3] -- L.__sizeof__() => 64
    - T = (1,2,3) -- T.__sizeof__() => 48

Python Basics: Functions

- Define/Call a function:

```
def fn(x, y):
    return x + y

fn(2,3) => 5
fn(“a”, “b”) => “ab”
fn(“1”, 1) => TypeError: cannot concatenate 'str' and 'int' objects
```

- Anonymous function: lambda x: x*x

```
def afn(f, arg):
    return (f(arg), f(arg))

afn(lambda x: x*x, 2) = > (4, 4)
afn(lambda x: abs(x), -2) => (2, 2)
```

Python Basics: Summary

- We reviewed some important details for Python data structure (Chapter 2).
- Other important concepts:
  - Modules
  - Control flow
  - Strings
  - Arithmetic

Q & A

Quiz!

- Navigate to Quizzes section on Canvas.
- You have 10 minutes to complete 5 multiple choice questions.
- Passcode: ds101