TM351 VM Installation Test

# TM351 VM Installation Test[¶](#TM351-VM-Installation-Test)

This notebook provides a series of tests to ensure that the virtual machine is running correctly.

Run each cell in turn by clicking the play button or keyboard shortcut shift-return. (A full list of keyboard shortcuts can be found from the *Help* menu or via the keyboard shortcut ESC-h.)

The cells should run without error.

## Versions[¶](#Versions)

Display the VM build version and build time, as well as database service versions and pandas version.

In [1]:

!cat /opt/version.txt

Version: TM351\_17J\_0.01 - build time: 2017-06-07/09:31.03.

In [2]:

! psql --version

psql (PostgreSQL) 9.5.7

In [3]:

! mongod --version

db version v3.4.4  
git version: 888390515874a9debd1b6c5d36559ca86b44babd  
OpenSSL version: OpenSSL 1.0.2g 1 Mar 2016  
allocator: tcmalloc  
modules: none  
build environment:  
 distmod: ubuntu1604  
 distarch: x86\_64  
 target\_arch: x86\_64

In [4]:

import pandas as pd  
pd.\_\_version\_\_

Out[4]:

'0.20.2'

## Test Core Packages[¶](#Test-Core-Packages)

In [5]:

import pandas as pd

In [6]:

import matplotlib.pyplot as plt

In [7]:

#When this cell is run, a simple line chart should be displayed  
plt.plot([1,2,3,4])  
plt.ylabel('some numbers')  
plt.show()

![](data:image/png;base64;base64,)

## Database tests[¶](#Database-tests)

Check that the database services are running as required.

In [8]:

#SET DATABASE CONNECTION STRINGS  
import os  
if os.environ.get('DOCKERBUILD')!='1':  
 #Database connection strings for monolithic VM  
 PGCONN='postgresql://test:test@localhost:5432/tm351test'  
 MONGOHOST='localhost'  
 MONGOPORT=27351  
else:  
 #Database connection strings for docker build  
 PGCONN='postgresql://postgres:PGPass@postgres:5432/tm351test'  
 MONGOHOST='mongodb'  
 MONGOPORT=27017  
MONGOCONN='mongodb://{MONGOHOST}:{MONGOPORT}/'.format(MONGOHOST=MONGOHOST,MONGOPORT=MONGOPORT)

### PostgreSQL[¶](#PostgreSQL)

Check the connection to the PostgreSQL server.

In [9]:

from sqlalchemy import create\_engine  
engine = create\_engine(PGCONN)

In [10]:

#Run a simple query on a default table  
from pandas import read\_sql\_query as psql  
  
psql("SELECT table\_schema,table\_name FROM information\_schema.tables \  
 ORDER BY table\_schema,table\_name LIMIT 3;", engine)  
#A table containing three rows should appear

Out[10]:

|  |  |  |
| --- | --- | --- |
|  | table\_schema | table\_name |
| 0 | information\_schema | administrable\_role\_authorizations |
| 1 | information\_schema | applicable\_roles |
| 2 | information\_schema | attributes |

#### SQL Cell Magic[¶](#SQL-Cell-Magic)

We can use cell magics to allow the writing of SQL statements within a code cell flagged appropriately.

To invoke the cell magic in a cell, we need to run the following (though we could perhaps autoload this in every notebook?)

In the following example, magic SQL cells will be configured to run as a the root user:

In [11]:

%load\_ext sql  
%sql {PGCONN}

Out[11]:

'Connected: test@tm351test'

In [12]:

%%sql  
SELECT table\_schema,table\_name FROM information\_schema.tables ORDER BY table\_schema,table\_name LIMIT 1;

1 rows affected.

Out[12]:

|  |  |
| --- | --- |
| table\_schema | table\_name |
| information\_schema | administrable\_role\_authorizations |

Test the ability to pull the result of a SQL query directly into a dataframe:

In [13]:

demo=%sql SELECT table\_schema FROM information\_schema.tables LIMIT 3  
demo

3 rows affected.

Out[13]:

|  |
| --- |
| table\_schema |
| pg\_catalog |
| pg\_catalog |
| pg\_catalog |

### MongDB[¶](#MongDB)

Test that the mongoDB database is running... This example also shows how to connect to the database.

In [14]:

import pymongo  
from pymongo import MongoClient

In [15]:

#If connecting to the default port, you can omit the second (port number) parameter  
# Open a connection to the Mongo server, open the accidents database and name the collections of accidents and labels  
c = pymongo.MongoClient(MONGOCONN)

By default, this database should contain an accidents database along with any default databases.

In [16]:

c.database\_names()

Out[16]:

['accidents', 'admin', 'local']

In [17]:

db = c.accidents  
accidents = db.accidents  
accidents.find\_one()

Out[17]:

{'1st\_Road\_Class': 3,  
 '1st\_Road\_Number': 308,  
 '2nd\_Road\_Class': 5,  
 '2nd\_Road\_Number': 0,  
 'Accident\_Index': '201201BS70001',  
 'Accident\_Severity': 3,  
 'Carriageway\_Hazards': 0,  
 'Casualties': [{'Age\_Band\_of\_Casualty': 7,  
 'Bus\_or\_Coach\_Passenger': 0,  
 'Car\_Passenger': 0,  
 'Casualty\_Class': 1,  
 'Casualty\_Home\_Area\_Type': 1,  
 'Casualty\_Reference': 1,  
 'Casualty\_Severity': 3,  
 'Casualty\_Type': 1,  
 'Pedestrian\_Location': 0,  
 'Pedestrian\_Movement': 0,  
 'Pedestrian\_Road\_Maintenance\_Worker': 0,  
 'Sex\_of\_Casualty': 1,  
 'Vehicle\_Reference': 2}],  
 'Date': '19/01/2012',  
 'Datetime': datetime.datetime(2012, 1, 19, 20, 35),  
 'Day\_of\_Week': 5,  
 'Did\_Police\_Officer\_Attend\_Scene\_of\_Accident': 1,  
 'Junction\_Control': 2,  
 'Junction\_Detail': 6,  
 'LSOA\_of\_Accident\_Location': 'E01002821',  
 'Latitude': 51.493429,  
 'Light\_Conditions': 4,  
 'Local\_Authority\_(District)': 12,  
 'Local\_Authority\_(Highway)': 'E09000020',  
 'Location\_Easting\_OSGR': 527200,  
 'Location\_Northing\_OSGR': 178760,  
 'Longitude': -0.169101,  
 'Number\_of\_Casualties': 1,  
 'Number\_of\_Vehicles': 2,  
 'Pedestrian\_Crossing-Human\_Control': 0,  
 'Pedestrian\_Crossing-Physical\_Facilities': 5,  
 'Police\_Force': 1,  
 'Road\_Surface\_Conditions': 1,  
 'Road\_Type': 6,  
 'Special\_Conditions\_at\_Site': 0,  
 'Speed\_limit': 30,  
 'Time': '20:35',  
 'Urban\_or\_Rural\_Area': 1,  
 'Vehicles': [{'1st\_Point\_of\_Impact': 4,  
 'Age\_Band\_of\_Driver': 4,  
 'Age\_of\_Vehicle': 5,  
 'Driver\_Home\_Area\_Type': 1,  
 'Driver\_IMD\_Decile': 7,  
 'Engine\_Capacity\_(CC)': 1390,  
 'Hit\_Object\_in\_Carriageway': 0,  
 'Hit\_Object\_off\_Carriageway': 0,  
 'Journey\_Purpose\_of\_Driver': 6,  
 'Junction\_Location': 8,  
 'Propulsion\_Code': 1,  
 'Sex\_of\_Driver': 2,  
 'Skidding\_and\_Overturning': 0,  
 'Towing\_and\_Articulation': 0,  
 'Vehicle\_Leaving\_Carriageway': 0,  
 'Vehicle\_Location-Restricted\_Lane': 0,  
 'Vehicle\_Manoeuvre': 9,  
 'Vehicle\_Reference': 1,  
 'Vehicle\_Type': 9,  
 'Was\_Vehicle\_Left\_Hand\_Drive?': 1},  
 {'1st\_Point\_of\_Impact': 1,  
 'Age\_Band\_of\_Driver': 7,  
 'Age\_of\_Vehicle': -1,  
 'Driver\_Home\_Area\_Type': 1,  
 'Driver\_IMD\_Decile': 3,  
 'Engine\_Capacity\_(CC)': -1,  
 'Hit\_Object\_in\_Carriageway': 0,  
 'Hit\_Object\_off\_Carriageway': 0,  
 'Journey\_Purpose\_of\_Driver': 6,  
 'Junction\_Location': 8,  
 'Propulsion\_Code': -1,  
 'Sex\_of\_Driver': 1,  
 'Skidding\_and\_Overturning': 0,  
 'Towing\_and\_Articulation': 0,  
 'Vehicle\_Leaving\_Carriageway': 0,  
 'Vehicle\_Location-Restricted\_Lane': 0,  
 'Vehicle\_Manoeuvre': 18,  
 'Vehicle\_Reference': 2,  
 'Vehicle\_Type': 1,  
 'Was\_Vehicle\_Left\_Hand\_Drive?': 1}],  
 'Weather\_Conditions': 1,  
 '\_id': ObjectId('52a9c97c92c4e16686d13265'),  
 'loc': {'coordinates': [-0.169101, 51.493429], 'type': 'Point'}}

### Sharded MongoDB server[¶](#Sharded-MongoDB-server)

A sharded mongo server, populated with content, is also provided:

* start the sharded server: !/etc/mongo-shards-up
* stop the sharded server: !/etc/mongo-shards-up

In [18]:

#Quick way to kill all mongo processes...  
#!killall mongod  
#!killall mongos  
#...then bring the base mongo server as service on 27351 back up  
#!systemctl restart mongodb

In [19]:

!/etc/mongo-shards-down  
!/etc/mongo-shards-up

/etc/mongo-shards-down: line 4: /vagrant/logs/mongocluster\_\*pid: No such file or directory  
Killing process found in /vagrant/logs/mongocluster\_\*pid  
kill: usage: kill [-s sigspec | -n signum | -sigspec] pid | jobspec ... or kill -l [sigspec]  
Wait a mo to check processes are down...  
...ok  
Starting config server...  
about to fork child process, waiting until server is ready for connections.  
forked process: 3731  
child process started successfully, parent exiting  
Sleep for 5...  
...done  
Configuring config replica set  
MongoDB shell version v3.4.4  
connecting to: mongodb://127.0.0.1:57050/  
MongoDB server version: 3.4.4  
{  
 "\_id" : "c1",  
 "members" : [  
 {  
 "\_id" : 0,  
 "host" : "localhost:57050"  
 }  
 ]  
}  
{  
 "info" : "try querying local.system.replset to see current configuration",  
 "ok" : 0,  
 "errmsg" : "already initialized",  
 "code" : 23,  
 "codeName" : "AlreadyInitialized"  
}  
bye  
Sleep for 5...  
...done  
2017-08-01T11:04:14.811+0000 W SHARDING [main] Running a sharded cluster with fewer than 3 config servers should only be done for testing purposes and is not recommended for production.  
about to fork child process, waiting until server is ready for connections.  
forked process: 3823  
child process started successfully, parent exiting  
Sleep for 5...  
...done  
starting servers for shard 0  
about to fork child process, waiting until server is ready for connections.  
forked process: 3847  
child process started successfully, parent exiting  
Sleep for 5...  
...done  
Configuring s0 replica set  
MongoDB shell version v3.4.4  
connecting to: mongodb://127.0.0.1:37017/  
MongoDB server version: 3.4.4  
{  
 "\_id" : "s0",  
 "members" : [  
 {  
 "\_id" : 0,  
 "host" : "localhost:37017"  
 }  
 ]  
}  
{  
 "info" : "try querying local.system.replset to see current configuration",  
 "ok" : 0,  
 "errmsg" : "already initialized",  
 "code" : 23,  
 "codeName" : "AlreadyInitialized"  
}  
bye  
starting servers for shard 1  
about to fork child process, waiting until server is ready for connections.  
forked process: 3942  
child process started successfully, parent exiting  
Sleep for 5...  
...done  
Configuring s1 replica set  
MongoDB shell version v3.4.4  
connecting to: mongodb://127.0.0.1:47017/  
MongoDB server version: 3.4.4  
{  
 "\_id" : "s1",  
 "members" : [  
 {  
 "\_id" : 0,  
 "host" : "localhost:47017"  
 }  
 ]  
}  
{  
 "info" : "try querying local.system.replset to see current configuration",  
 "ok" : 0,  
 "errmsg" : "already initialized",  
 "code" : 23,  
 "codeName" : "AlreadyInitialized"  
}  
bye  
starting servers for shard 2  
about to fork child process, waiting until server is ready for connections.  
forked process: 4037  
child process started successfully, parent exiting  
Sleep for 5...  
...done  
Configuring s2 replica set  
MongoDB shell version v3.4.4  
connecting to: mongodb://127.0.0.1:57017/  
MongoDB server version: 3.4.4  
{  
 "\_id" : "s2",  
 "members" : [  
 {  
 "\_id" : 0,  
 "host" : "localhost:57017"  
 }  
 ]  
}  
{  
 "info" : "try querying local.system.replset to see current configuration",  
 "ok" : 0,  
 "errmsg" : "already initialized",  
 "code" : 23,  
 "codeName" : "AlreadyInitialized"  
}  
bye  
Waiting 10 seconds for the replica sets to fully come online...  
...done: should be online now

Once again, an *accidents* database should be available as well as administrative databases.

In [20]:

c2 = pymongo.MongoClient('mongodb://localhost:27017/')  
c2.database\_names()

Out[20]:

['accidents', 'admin', 'config']

In [21]:

#Test a query on the sharded database  
db = c2.accidents  
accidents = db.accidents  
accidents.find\_one()

Out[21]:

{'1st\_Road\_Class': 5,  
 '1st\_Road\_Number': 0,  
 '2nd\_Road\_Class': 5,  
 '2nd\_Road\_Number': 0,  
 'Accident\_Index': '200901BS70002',  
 'Accident\_Severity': 2,  
 'Carriageway\_Hazards': 0,  
 'Casualties': [{'Age\_Band\_of\_Casualty': 7,  
 'Bus\_or\_Coach\_Passenger': 0,  
 'Car\_Passenger': 0,  
 'Casualty\_Class': 1,  
 'Casualty\_Home\_Area\_Type': 1,  
 'Casualty\_Reference': 1,  
 'Casualty\_Severity': 3,  
 'Casualty\_Type': 19,  
 'Pedestrian\_Location': 0,  
 'Pedestrian\_Movement': 0,  
 'Pedestrian\_Road\_Maintenance\_Worker': -1,  
 'Sex\_of\_Casualty': 1,  
 'Vehicle\_Reference': 1},  
 {'Age\_Band\_of\_Casualty': 7,  
 'Bus\_or\_Coach\_Passenger': 0,  
 'Car\_Passenger': 0,  
 'Casualty\_Class': 1,  
 'Casualty\_Home\_Area\_Type': 1,  
 'Casualty\_Reference': 2,  
 'Casualty\_Severity': 3,  
 'Casualty\_Type': 11,  
 'Pedestrian\_Location': 0,  
 'Pedestrian\_Movement': 0,  
 'Pedestrian\_Road\_Maintenance\_Worker': -1,  
 'Sex\_of\_Casualty': 1,  
 'Vehicle\_Reference': 2},  
 {'Age\_Band\_of\_Casualty': 11,  
 'Bus\_or\_Coach\_Passenger': 4,  
 'Car\_Passenger': 0,  
 'Casualty\_Class': 2,  
 'Casualty\_Home\_Area\_Type': 1,  
 'Casualty\_Reference': 3,  
 'Casualty\_Severity': 2,  
 'Casualty\_Type': 11,  
 'Pedestrian\_Location': 0,  
 'Pedestrian\_Movement': 0,  
 'Pedestrian\_Road\_Maintenance\_Worker': -1,  
 'Sex\_of\_Casualty': 2,  
 'Vehicle\_Reference': 2},  
 {'Age\_Band\_of\_Casualty': 7,  
 'Bus\_or\_Coach\_Passenger': 4,  
 'Car\_Passenger': 0,  
 'Casualty\_Class': 2,  
 'Casualty\_Home\_Area\_Type': 1,  
 'Casualty\_Reference': 4,  
 'Casualty\_Severity': 2,  
 'Casualty\_Type': 11,  
 'Pedestrian\_Location': 0,  
 'Pedestrian\_Movement': 0,  
 'Pedestrian\_Road\_Maintenance\_Worker': -1,  
 'Sex\_of\_Casualty': 1,  
 'Vehicle\_Reference': 2},  
 {'Age\_Band\_of\_Casualty': 6,  
 'Bus\_or\_Coach\_Passenger': 4,  
 'Car\_Passenger': 0,  
 'Casualty\_Class': 2,  
 'Casualty\_Home\_Area\_Type': 1,  
 'Casualty\_Reference': 5,  
 'Casualty\_Severity': 3,  
 'Casualty\_Type': 11,  
 'Pedestrian\_Location': 0,  
 'Pedestrian\_Movement': 0,  
 'Pedestrian\_Road\_Maintenance\_Worker': -1,  
 'Sex\_of\_Casualty': 2,  
 'Vehicle\_Reference': 2},  
 {'Age\_Band\_of\_Casualty': 4,  
 'Bus\_or\_Coach\_Passenger': 4,  
 'Car\_Passenger': 0,  
 'Casualty\_Class': 2,  
 'Casualty\_Home\_Area\_Type': 1,  
 'Casualty\_Reference': 6,  
 'Casualty\_Severity': 3,  
 'Casualty\_Type': 11,  
 'Pedestrian\_Location': 0,  
 'Pedestrian\_Movement': 0,  
 'Pedestrian\_Road\_Maintenance\_Worker': -1,  
 'Sex\_of\_Casualty': 2,  
 'Vehicle\_Reference': 2},  
 {'Age\_Band\_of\_Casualty': 10,  
 'Bus\_or\_Coach\_Passenger': 4,  
 'Car\_Passenger': 0,  
 'Casualty\_Class': 2,  
 'Casualty\_Home\_Area\_Type': 1,  
 'Casualty\_Reference': 7,  
 'Casualty\_Severity': 3,  
 'Casualty\_Type': 11,  
 'Pedestrian\_Location': 0,  
 'Pedestrian\_Movement': 0,  
 'Pedestrian\_Road\_Maintenance\_Worker': -1,  
 'Sex\_of\_Casualty': 2,  
 'Vehicle\_Reference': 2},  
 {'Age\_Band\_of\_Casualty': 6,  
 'Bus\_or\_Coach\_Passenger': 4,  
 'Car\_Passenger': 0,  
 'Casualty\_Class': 2,  
 'Casualty\_Home\_Area\_Type': 1,  
 'Casualty\_Reference': 8,  
 'Casualty\_Severity': 3,  
 'Casualty\_Type': 11,  
 'Pedestrian\_Location': 0,  
 'Pedestrian\_Movement': 0,  
 'Pedestrian\_Road\_Maintenance\_Worker': -1,  
 'Sex\_of\_Casualty': 2,  
 'Vehicle\_Reference': 2},  
 {'Age\_Band\_of\_Casualty': 10,  
 'Bus\_or\_Coach\_Passenger': 4,  
 'Car\_Passenger': 0,  
 'Casualty\_Class': 2,  
 'Casualty\_Home\_Area\_Type': -1,  
 'Casualty\_Reference': 9,  
 'Casualty\_Severity': 3,  
 'Casualty\_Type': 11,  
 'Pedestrian\_Location': 0,  
 'Pedestrian\_Movement': 0,  
 'Pedestrian\_Road\_Maintenance\_Worker': -1,  
 'Sex\_of\_Casualty': 1,  
 'Vehicle\_Reference': 2},  
 {'Age\_Band\_of\_Casualty': 7,  
 'Bus\_or\_Coach\_Passenger': 4,  
 'Car\_Passenger': 0,  
 'Casualty\_Class': 2,  
 'Casualty\_Home\_Area\_Type': 1,  
 'Casualty\_Reference': 10,  
 'Casualty\_Severity': 3,  
 'Casualty\_Type': 11,  
 'Pedestrian\_Location': 0,  
 'Pedestrian\_Movement': 0,  
 'Pedestrian\_Road\_Maintenance\_Worker': -1,  
 'Sex\_of\_Casualty': 2,  
 'Vehicle\_Reference': 2},  
 {'Age\_Band\_of\_Casualty': 10,  
 'Bus\_or\_Coach\_Passenger': 4,  
 'Car\_Passenger': 0,  
 'Casualty\_Class': 2,  
 'Casualty\_Home\_Area\_Type': 1,  
 'Casualty\_Reference': 11,  
 'Casualty\_Severity': 3,  
 'Casualty\_Type': 11,  
 'Pedestrian\_Location': 0,  
 'Pedestrian\_Movement': 0,  
 'Pedestrian\_Road\_Maintenance\_Worker': -1,  
 'Sex\_of\_Casualty': 1,  
 'Vehicle\_Reference': 2}],  
 'Date': '05/01/2009',  
 'Datetime': datetime.datetime(2009, 1, 5, 10, 59),  
 'Day\_of\_Week': 2,  
 'Did\_Police\_Officer\_Attend\_Scene\_of\_Accident': 1,  
 'Junction\_Control': 4,  
 'Junction\_Detail': 6,  
 'LSOA\_of\_Accident\_Location': 'E01002886',  
 'Latitude': 51.514399,  
 'Light\_Conditions': 1,  
 'Local\_Authority\_(District)': 12,  
 'Local\_Authority\_(Highway)': 'E09000020',  
 'Location\_Easting\_OSGR': 525050,  
 'Location\_Northing\_OSGR': 181040,  
 'Longitude': -0.199248,  
 'Number\_of\_Casualties': 11,  
 'Number\_of\_Vehicles': 2,  
 'Pedestrian\_Crossing-Human\_Control': 0,  
 'Pedestrian\_Crossing-Physical\_Facilities': 1,  
 'Police\_Force': 1,  
 'Road\_Surface\_Conditions': 2,  
 'Road\_Type': 6,  
 'Special\_Conditions\_at\_Site': 0,  
 'Speed\_limit': 30,  
 'Time': '10:59',  
 'Urban\_or\_Rural\_Area': 1,  
 'Vehicles': [{'1st\_Point\_of\_Impact': 1,  
 'Age\_Band\_of\_Driver': 7,  
 'Age\_of\_Vehicle': 3,  
 'Driver\_Home\_Area\_Type': 1,  
 'Driver\_IMD\_Decile': 3,  
 'Engine\_Capacity\_(CC)': 1753,  
 'Hit\_Object\_in\_Carriageway': 0,  
 'Hit\_Object\_off\_Carriageway': 0,  
 'Journey\_Purpose\_of\_Driver': 1,  
 'Junction\_Location': 8,  
 'Propulsion\_Code': 2,  
 'Sex\_of\_Driver': 1,  
 'Skidding\_and\_Overturning': 0,  
 'Towing\_and\_Articulation': 0,  
 'Vehicle\_Leaving\_Carriageway': 0,  
 'Vehicle\_Location-Restricted\_Lane': 3,  
 'Vehicle\_Manoeuvre': 18,  
 'Vehicle\_Reference': 1,  
 'Vehicle\_Type': 19,  
 'Was\_Vehicle\_Left\_Hand\_Drive?': 1},  
 {'1st\_Point\_of\_Impact': 4,  
 'Age\_Band\_of\_Driver': 7,  
 'Age\_of\_Vehicle': 1,  
 'Driver\_Home\_Area\_Type': 1,  
 'Driver\_IMD\_Decile': 2,  
 'Engine\_Capacity\_(CC)': 6700,  
 'Hit\_Object\_in\_Carriageway': 10,  
 'Hit\_Object\_off\_Carriageway': 10,  
 'Journey\_Purpose\_of\_Driver': 1,  
 'Junction\_Location': 8,  
 'Propulsion\_Code': 2,  
 'Sex\_of\_Driver': 1,  
 'Skidding\_and\_Overturning': 0,  
 'Towing\_and\_Articulation': 0,  
 'Vehicle\_Leaving\_Carriageway': 7,  
 'Vehicle\_Location-Restricted\_Lane': 0,  
 'Vehicle\_Manoeuvre': 18,  
 'Vehicle\_Reference': 2,  
 'Vehicle\_Type': 11,  
 'Was\_Vehicle\_Left\_Hand\_Drive?': 1}],  
 'Weather\_Conditions': 1,  
 '\_id': ObjectId('52a9c96192c4e16686ca08a4'),  
 'loc': {'coordinates': [-0.199248, 51.514399], 'type': 'Point'}}

In [22]:

#Turn the sharded server off  
!/etc/mongo-shards-down

Killing process 3731 found in /vagrant/logs/mongocluster\_cfg-a2.pid  
Killing process 3823 found in /vagrant/logs/mongocluster\_mongos-2.pid  
Killing process 3847 found in /vagrant/logs/mongocluster\_s0-r0.pid  
Killing process 3942 found in /vagrant/logs/mongocluster\_s1-r0.pid  
Killing process 4037 found in /vagrant/logs/mongocluster\_s2-r0.pid  
Wait a mo to check processes are down...  
...ok

## Chart Tests[¶](#Chart-Tests)

Viewing data in charts provides a handy way of actually looking at your data...

In [23]:

import seaborn  
  
from numpy.random import randn  
data = randn(75)  
plt.hist(data);  
#Running this cell should produce a histogram.

![](data:image/png;base64;base64,)

In [24]:

ts = pd.Series(np.random.randn(1000), index=pd.date\_range('1/1/2000', periods=1000))  
ts = ts.cumsum()  
ts.plot();  
#Running this cell should produce a line chart.

![](data:image/png;base64;base64,)

## Maps[¶](#Maps)

Several of the data investigations may benefit from displaying data on a map. Test that the mapping functions work:

In [26]:

import folium  
#Note - this will not display a map if you are offline.  
  
#A network connection is required to retrieve the map tiles  
osmap = folium.Map(location=[52.01, -0.71], zoom\_start=13,height=500,width=800)  
folium.Marker([52.0250, -0.7056], popup='The <b>Open University</b> campus.').add\_to(osmap)  
osmap.save('test.html')  
osmap

Out[26]:

## Other VM Services[¶](#Other-VM-Services)

* [OpenRefine - by default on host port 35181](http://127.0.0.1:35181)

From the notebook home page (/tree) you should also be able to launch a terminal as well as a new notebook.