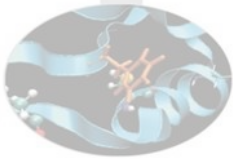
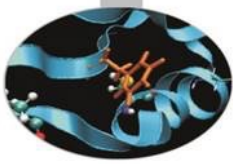
Cosa sappiamo di python adesso



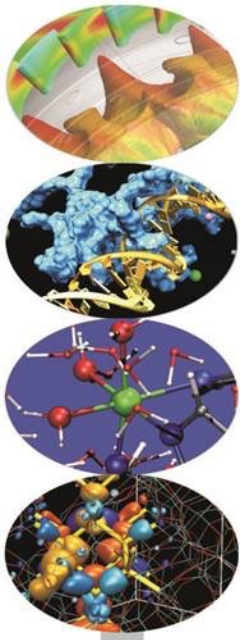
- Cos'è l'interprete Python
- La sintassi
- Utilizzo da shell
- Utilizzo codice da file (batch mode)
- Introspezione
- Strutture dati (es. liste e dizionari)
- Accesso ai file (lettura e scrittura)
- Creare e utilizzare funzioni
- Creare e utilizzare moduli

CINECA SCAI
SuperComputing Applications and Innovation



Becoming a professional

L'ambiente di sviluppo

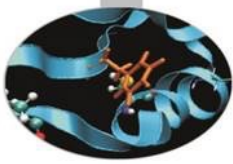


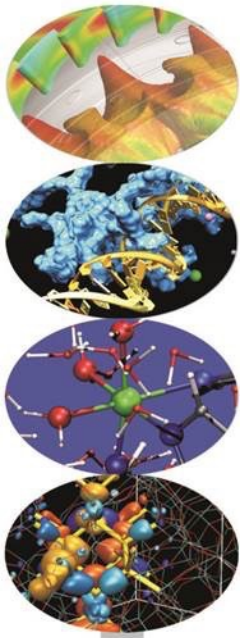
Materiale del corso

http://j.mp/cineca_scpy

Paolo D'Onorio De Meo
p.donoriodemeo@cineca.it

CINECA

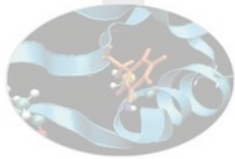




Ambiente di sviluppo

Riuscire ad evolversi

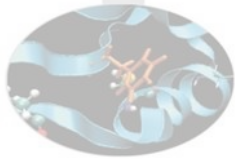
L'ambiente di sviluppo



Sistema software che fornisce la condizioni per scrivere ed eseguire il codice che costituisce una certa applicazione

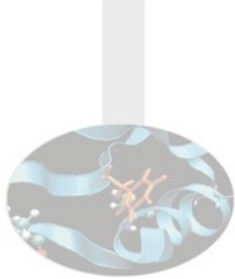
- Es. sito web
  - Server web (Apache?)
  - Interprete PHP (Apache php_mod)
  - Server database (Mysql?)
  - Database manager (phpmyadmin?)
- Es. applicazione java desktop
  - Java Virtual Machine ultima versione
  - Funzionante su OS di riferimento

Il vostro ambiente di sviluppo



- **Python 2**, ultima versione (current 2.7.8)
  - Librerie matematiche (es. lapack, blas)
  - Librerie numeriche (module numpy)
  - Librerie scientifiche (module scipy)
  - Compilatori di sistema
    - C
    - Fotran
  - Shell interattiva (Ipython)
  - Condividere codice
    - Server web
    - Modulo python (Ipython notebook)

Esempio dell'evoluzione: sviluppo web



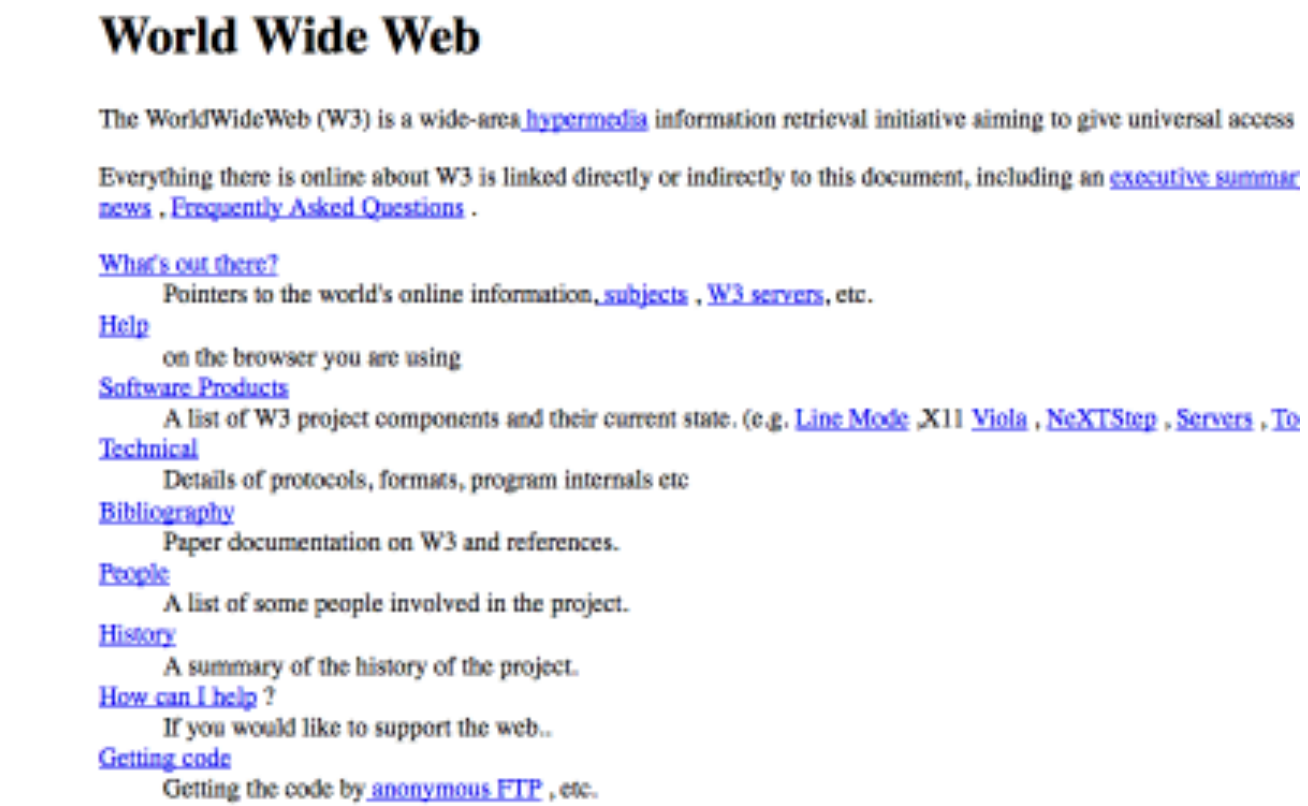
Siti web *statici*

1990-2000

- HTML puro
- Sviluppo in locale
- Trasferimento sul server in produzione tramite FTP

Esempio dell'evoluzione: sviluppo web

Siti web *statici* 1990-2000



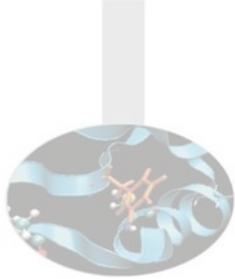
World Wide Web

The WorldWideWeb (W3) is a wide-area hypermedia information retrieval initiative aiming to give universal access

Everything there is online about W3 is linked directly or indirectly to this document, including an executive summar news , Frequently Asked Questions .

What's out there?
Pointers to the world's online information, subjects , W3 servers, etc.
Help
on the browser you are using
Software Products
A list of W3 project components and their current state. (e.g. Line Mode ,X11 Viola , NeXTStep , Servers , To
Technical
Details of protocols, formats, program internals etc
Bibliography
Paper documentation on W3 and references.
People
A list of some people involved in the project.
History
A summary of the history of the project.
How can I help ?
If you would like to support the web..
Getting code
Getting the code by anonymous FTP , etc.

Esempio dell'evoluzione: sviluppo web



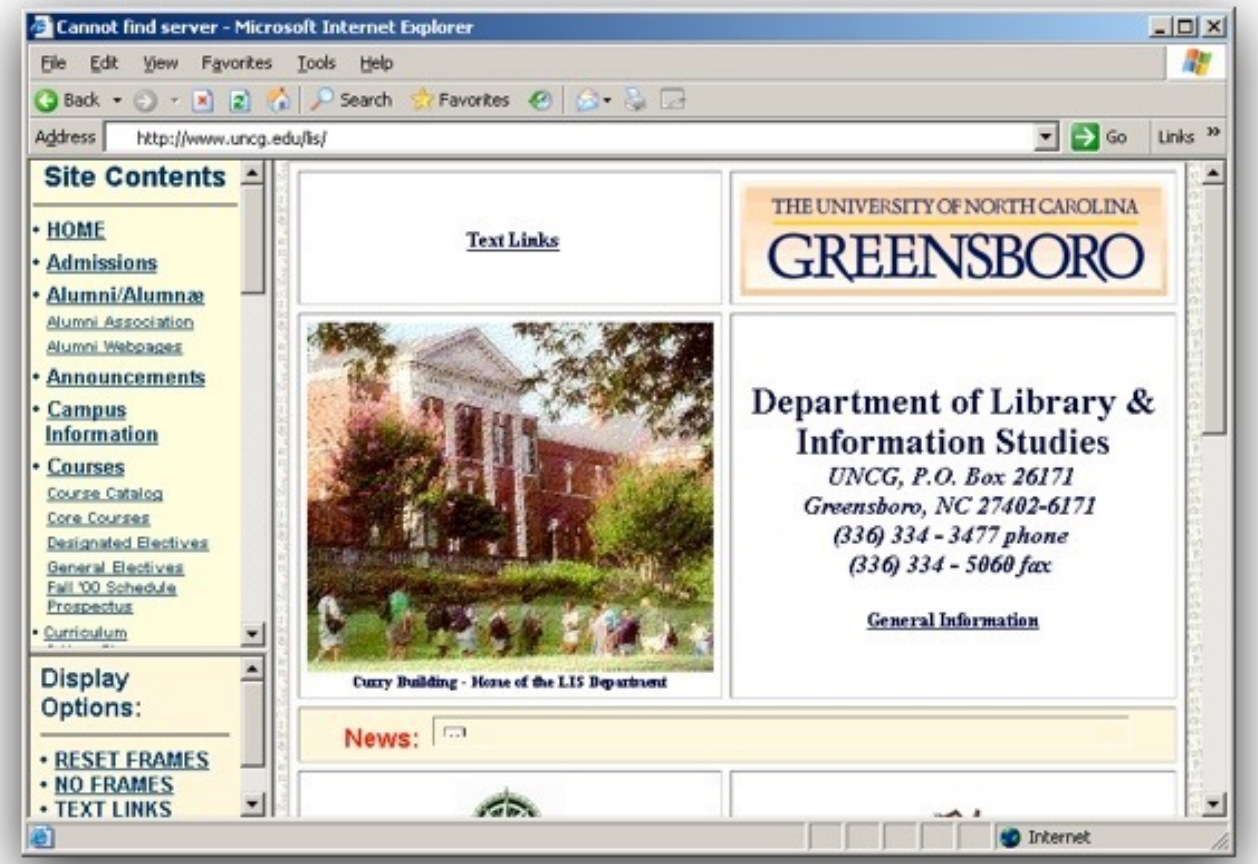
Siti web *dinamici*

2000-2005

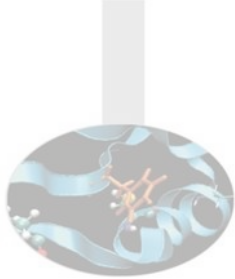
- ASP, PHP
- Database
- Sviluppo direttamente sul server remoto
- Editor su computer locale connesso via SFTP su cartella remota
- ….oppure editor sul server remoto (VIM / Emacs)
  - hard core :)

Esempio dell'evoluzione: sviluppo web

Siti web *dinamici* 2000-2005



Esempio dell'evoluzione: sviluppo web



Siti web *dinamici riproducibili*

2005-2010

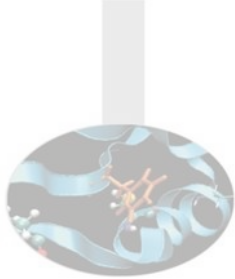
- XAMP stack
  - Cambio computer?
- Sviluppo di nuovo in locale
  - Sviluppo concorrente (Subversioning)
- Trasferimento sul server in produzione per le Release Canditate
  - heavy debugging…

Esempio dell'evoluzione: sviluppo web

Siti web *dinamici*

2005-2010

Esempio dell'evoluzione: sviluppo web

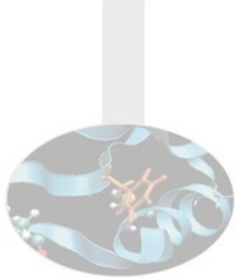


Siti web *2.0*

2010-2014

- Sviluppo di nuovo in locale
  - Front-end (UI-design + Codice client)
- Sviluppo anche in remoto
  - Back-end (Database + Cluster computation)
- Sviluppo concorrente: GIT /Mercurial
  - Branch di sviluppo e workflow di revisione codice

Esempio dell'evoluzione: sviluppo web

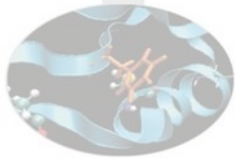


Applicazioni web

2014-2018

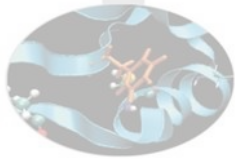
- Sviluppo in locale?

Situazione ideale



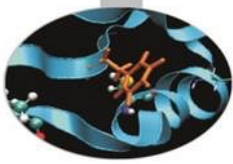
Un ambiente di sviluppo **unico**

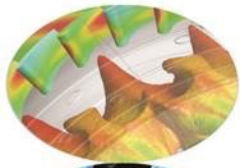
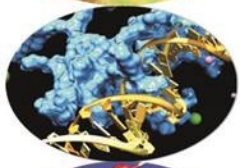
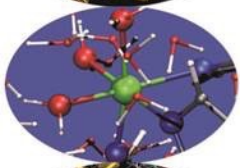
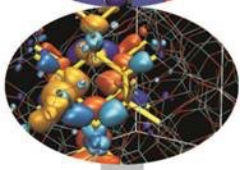
- identico per ogni sviluppatore
- *simula* verosimilmente tutte le condizioni dell'ambiente sul server in produzione
- duplicato facilmente su uno o più server remoti gemelli
  - alta affidabilità
  - scalabilità
  - distribuzione del carico



Calcolo scientifico: situazione (quasi) ideale

- Un nodo di login con scheduler server
  - Simulazione di un cluster di calcolo
- Due o più nodi identici
  - OpenMPI
- File system condiviso
  - Esportando una cartella condivisa dal computer host di docker
- Compilatori e moduli
  - Ambienti diversi (e.g. gcc in diverse versioni, gfortran, etc.)

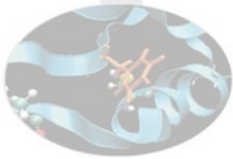


Corso python

Ambiente Virtuale

Macchina virtuale



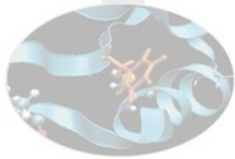
…attraverso un processo di virtualizzazione, crea un **ambiente virtuale** che emula tipicamente il comportamento di una macchina fisica

grazie all'assegnazione di risorse hardware (porzioni di disco rigido, RAM e risorse di processamento)

ed in cui alcune applicazioni possono essere eseguite come se interagissero con tale macchina;

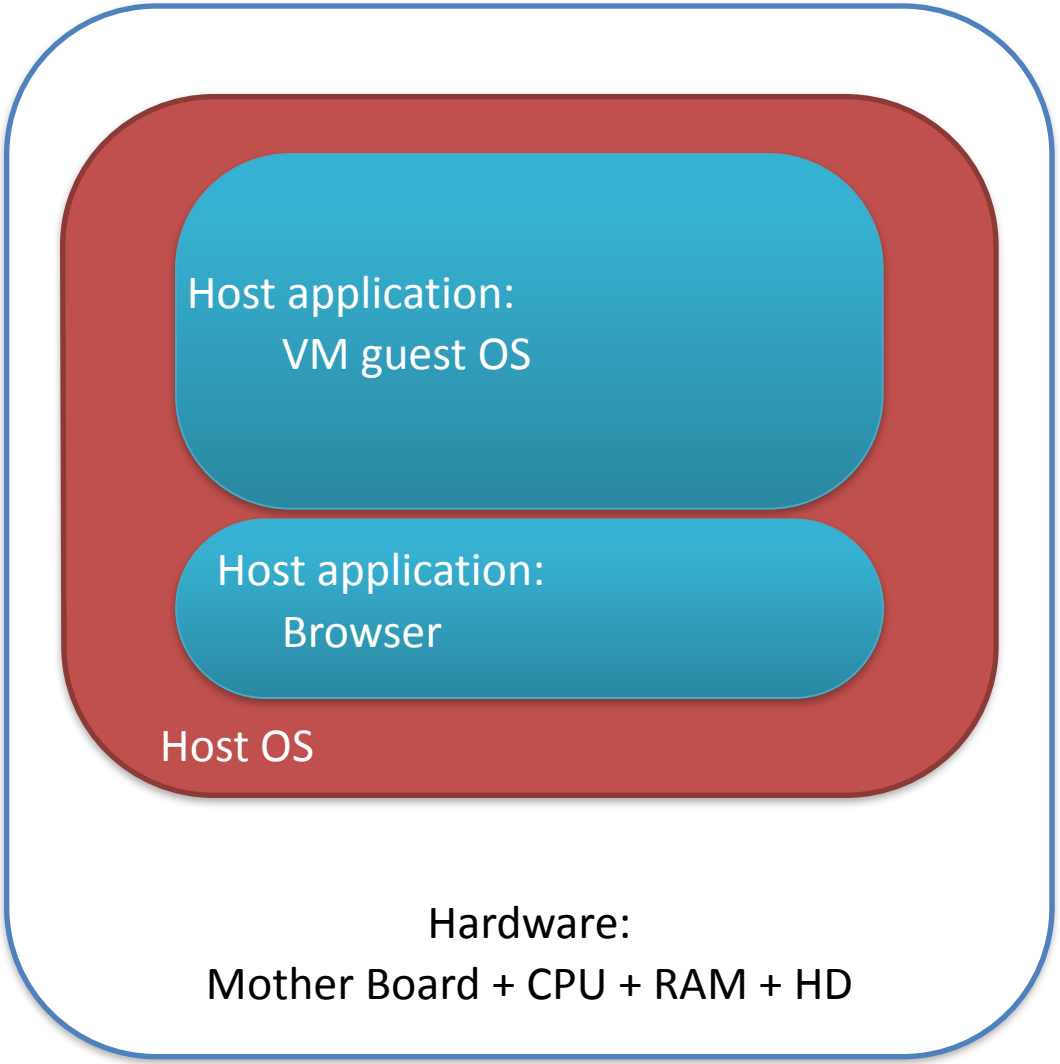
fonte: Wikipedia

Ambiente virtuale

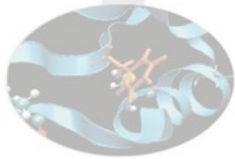


- Un software di virtualizzazione permette di creare un “finto computer” all’interno di un computer reale
  - Host: computer reale
  - Guest: macchina virtuale
- Vantaggi
  - Avere a disposizione un altro sistema operativo (windows, mac, linux) senza dover riavviare il nostro computer
    - la macchina virtuale guest è un comune programma all’interno dell’host
  - Tutto quello che accade all’interno della macchina virtuale non ha ripercussioni fuori di essa.
- Svantaggi (dipende dalle risorse del computer)
  - Host diventa più lento
  - Guest non veloce come se fosse un computer reale

Ambiente virtuale

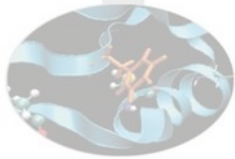


Utilizzo classico per workstation



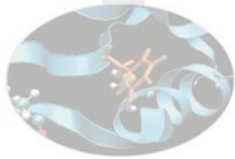
- Strumento di virtualizzazione
  - VirtualBox
  - VMware
- Immagine virtuale
  - Windows, Linux
    - Test applicativi su diversi sistemi operativi
- Cartella condivisa tra Host e Guest
- Condivisione di una parte delle risorse
  - CPU e/o GPU
  - RAM
  - Spazio disco

Utilizzo classico per cluster



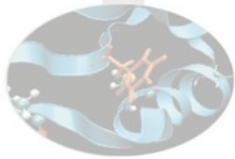
- Spazio disco
  - Condiviso
  - Backup
- Strumento di distribuzione immagini
  - Via spazio disco condiviso
- Immagine virtuale (prefabbricata)
  - Linux enterprise
    - Riproduzione di una o più immagini su tutto il cluster

Utilizzo “enterprise”



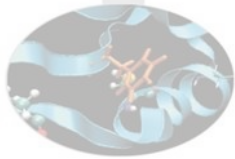
- Riproduzione di sotto-ambienti chiusi
  - rete privata
  - ip virtuali
  - riavvio con dimensioni risorse variabili
- Hosting virtuale
  - Cloud
  - Amazon webservices
  - Amazon S3

Il presente e il futuro



- Computer desktop performanti
  - più dei server del passato…
  - su elaborazione medio-piccole molto vicino a i server di produzione
- Tante immagini virtuali?
  - Eseguire più macchine virtuali in locale
    - E emulare rete, nomi, albero della struttura disco originali
  - Le diverse “immagini” non condividono nulla
    - a livello di sistema operativo
    - anche quando sono distribuzioni simili o identiche
  - Problema: “Divora” risorse

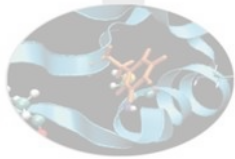
Docker



Dal sito web di Docker

- Docker solves the isolation problem.
- consistent, reproducible, disposable containers
  - make components appear to be running on different machines,
  - while sharing CPU and memory underneath,
  - and provides TCP/IP forwarding
  - and filesystems that can be shared between containers.

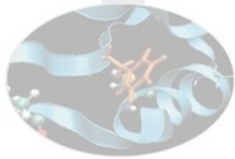
Docker



Progetto open-source per automatizzare lo sviluppo di applicazione

- dentro “contenitori” di software
- provvede un addizionale livello di astrazione (e automatizzazione)
  - virtualizzazione del livello di sistema operativo **Linux**
- Docker sfrutta delle features del *kernel* Linux
  - risorse di “isolamento”
    - cgroups
    - namespaces
  - possiamo eseguire contenitori indipendenti
    - all’interno della stessa istanza Linux

Kernel



Costituisce il nucleo di un sistema operativo.

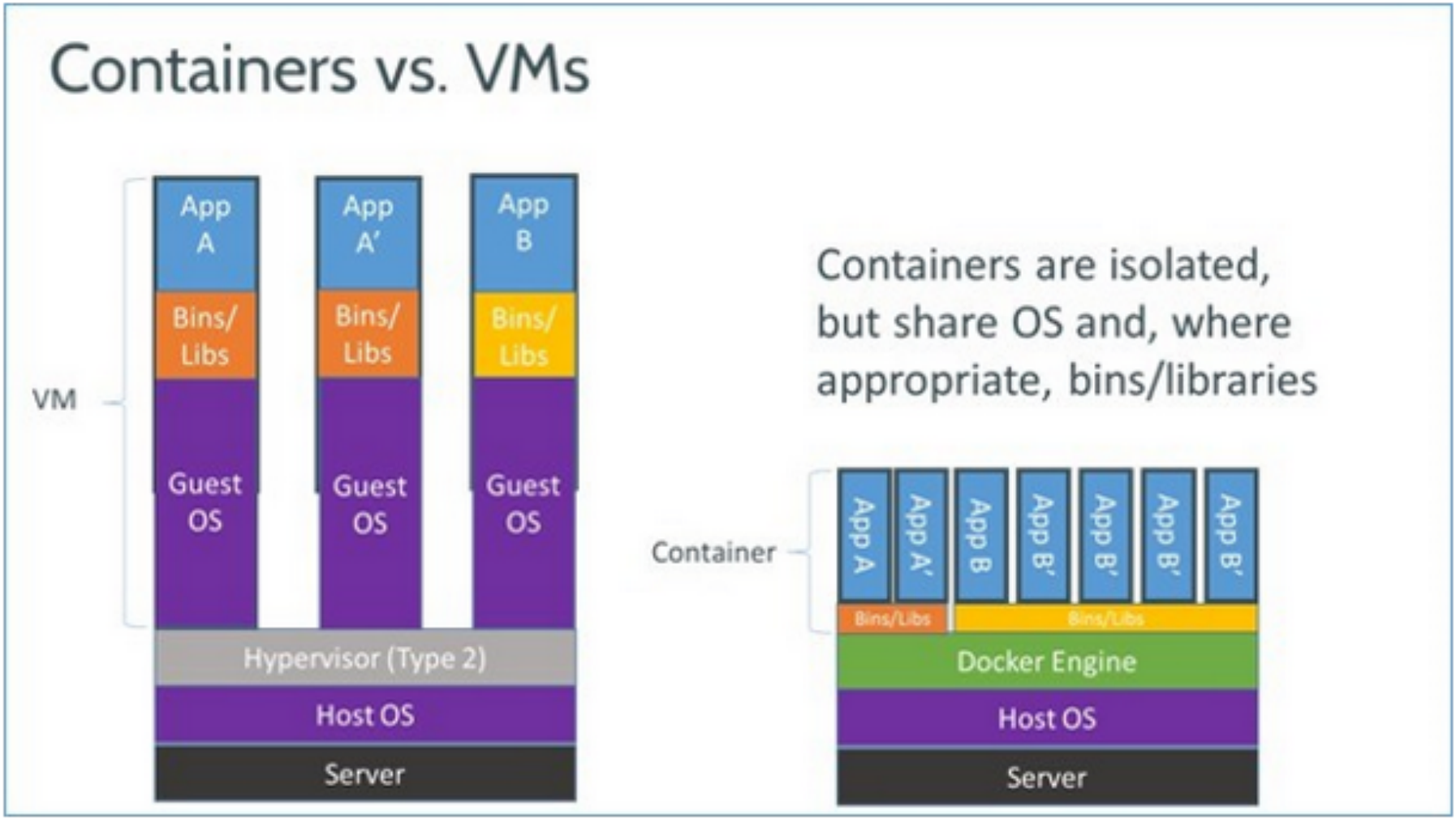
Ha il compito di fornire ai processi in esecuzione sull'elaboratore un accesso sicuro e controllato all'hardware.

fonte: Wikipedia

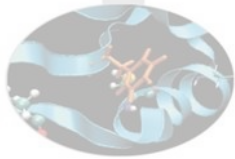
Governa l'utilizzo dell'hardware

e ne distribuisce le risorse

Docker: “containers”

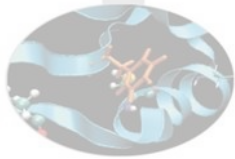


Docker: “containers”



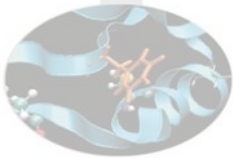
- Container
  - Concetto sorto nel 2000 con FreeBSD
  - Formati proprietari (Zone di Oracle Solaris)
  - Formati open-source (LXC Linux Containers e OpenVZ)
    - Google
      - technology Lmctfy (Let Me Contain That For You)
      - Search, Gmail, Google Docs
        - ad ogni utilizzo di ogni persona, viene lanciato un container…
    - Parallels, Docker
- Hypervisor
  - astrazione di un intero “device”
- Docker
  - astrazione del kernel del sistema operativo
  - funziona su LXC
  - Numero di instance **da 6 a 4 volte** maggiori se si usano Containers al posto di VM

Docker



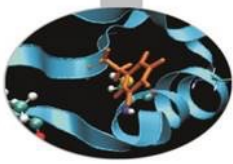
- Gratuito
  - Per Windows e Mac: funziona su *Virtualbox* (boot2docker VM)
- Non richiede più risorse per avviare macchine simili
  - Condivide il “cuore” (kernel) del sistema
  - Permette assegnamento di nomi (hostname) ai containers
- Non richiede alcuna conoscenza dei protocolli di rete
  - accesso tra i containers tramite i nomi

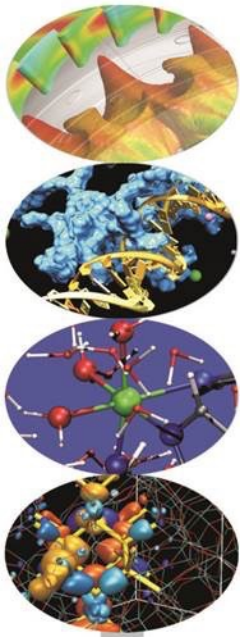
Docker: FAQ



- **How much does Docker cost?**
  - Docker is 100% free, it is open source, so you can use it without paying.
- **Does Docker run on Mac OS X or Windows?**
  - Not at this time, Docker currently only runs on Linux, but you can use VirtualBox to run Docker in a virtual machine on your box, and get the best of both worlds. Check out the Mac OS X and Microsoft Windows installation guides. The small Linux distribution boot2docker can be run inside virtual machines on these two operating systems.
- **How do containers compare to virtual machines?**
  - They are complementary. VMs are best used to allocate chunks of hardware resources. Containers operate at the process level, which makes them very lightweight and perfect as a unit of software delivery.

CINECA SCAI
SuperComputing Applications and Innovation



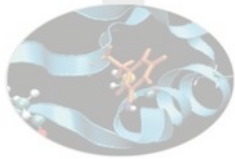


Corso python

Costruire il vostro ambiente virtuale

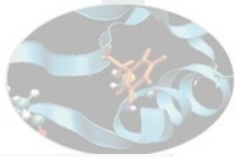
CINECA

Costruire un container



- Un container viene eseguito a partire da una “immagine”
  - Una immagine docker è composta di uno stack di comandi
    - Ogni comando è salvato con un “commit” (GIT-like)
      - Il comando più utilizzato è RUN
      - Ogni comando RUN è un comando di shell su Linux
    - I comandi sono raccolti in un file di configurazione (Dockerfile)

Dockerfile



```
FROM ubuntu:14.04
# Install ubuntu packages (compilers and other tools)
RUN apt-get update && apt-get install -y expect build-essential gfortran
# Install anaconda light
RUN mkdir -p /opt
WORKDIR /opt
ADD Miniconda*sh /opt/installer.sh
ADD expect.sh /opt/
# execute and clean
RUN cd /opt && ./expect.sh
# Add to path
ENV PATH $PATH:/opt/miniconda/bin
# Install minimal packages - reply with yes to install
RUN yes | conda create -n scientific ipython-notebook numpy scipy matplotlib cython
# Set up scripts for your needs
RUN echo "ipython notebook --ip=0.0.0.0 –port=8888 --no-browser" > /opt/start_notebook.sh
# notebook port
EXPOSE 8888
# Activate virtualenv
RUN rm /bin/sh && ln -s /bin/bash /bin/sh
RUN echo "source /opt/miniconda/bin/activate scientific" >> /root/.bashrc
```

Build & run

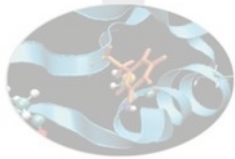
Dockerfile

```
FROM ubuntu

RUN apt-get install php
```

Bash

```
$ docker build -t myimage .

$ docker run -it myimage bash

myimage$ php —version
```

Build & run: versioni diverse



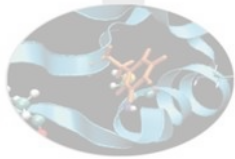
Dockerfile

```
FROM ubuntu:12.04

RUN apt-get install php
```

Bash

```
$ docker build -t myimage .

$ docker run -it myimage bash

myimage$ php —version
```

Build & run: compilatori



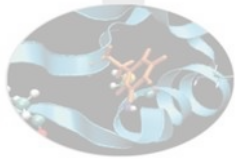
Dockerfile

```
FROM ubuntu:14.04

RUN apt-get update
RUN apt-get install -y gfortran
```

Bash

```
$ docker build -t myimage .

$ docker run -it myimage bash

myimage$ fortran compiler
```

Build & run: scientific python



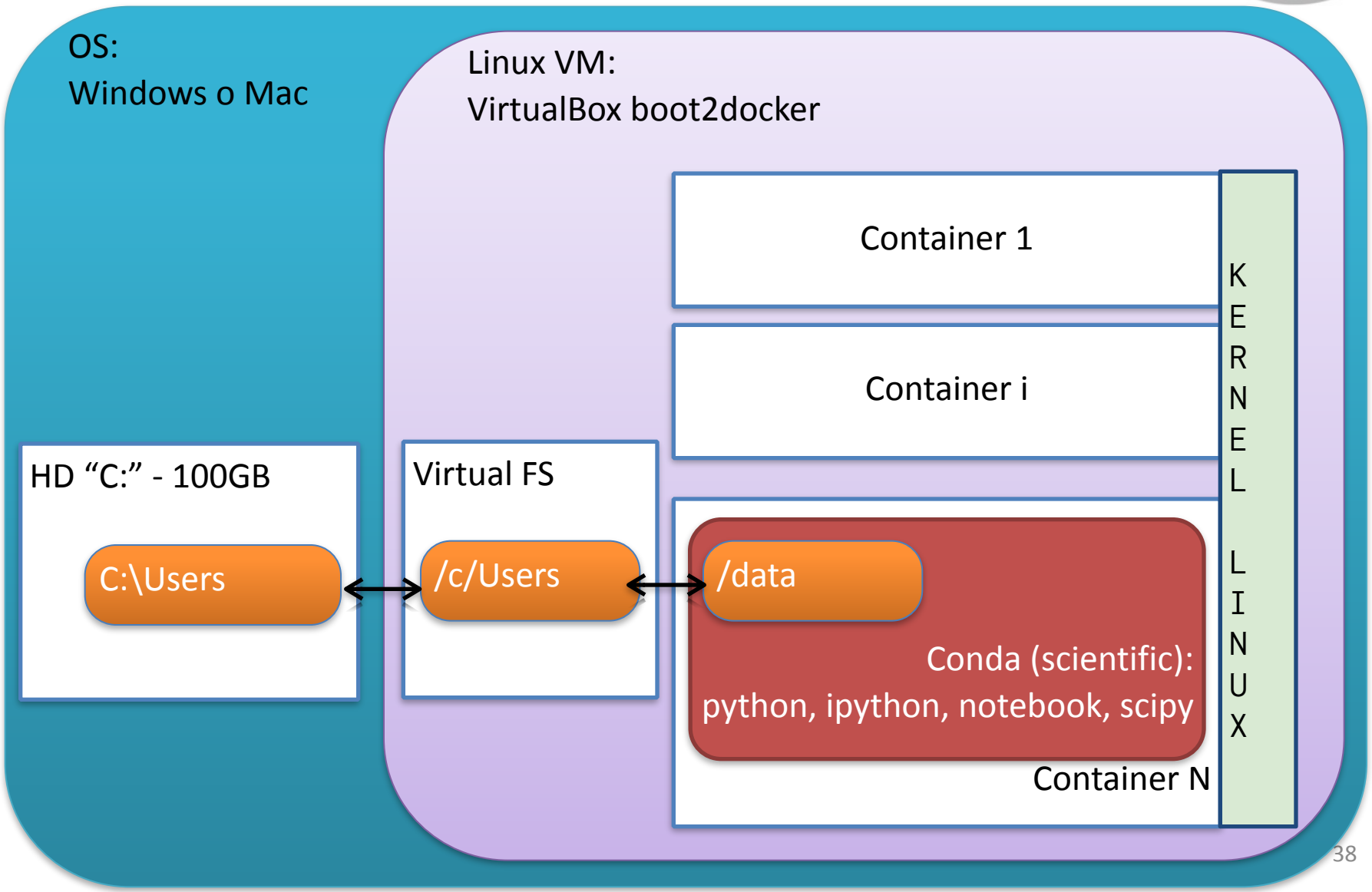
Dockerfile

```
FROM ubuntu:14.04
RUN apt-get update
RUN apt-get install -y build-essential

ADD miniconda.sh /opt/installer.sh
WORKDIR /opt
RUN sh installer.sh

RUN conda create -n scientific
       ipython-notebook numpy scipy
       matplotlib cython
```

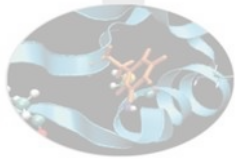
Bash

```
$ docker build -t myimage .

$ docker run -it myimage bash

myimage$ source activate scientific

(scientific)$ python

>>> import scipy
>>>
```

Sul vostro portatile



Sul vostro portatile



- Windows

windows cmd>

  - Linux vm (boot2docker)

docker@boot2docker:~$

    - Container bash

(scientific)root@<cid>:/opt#

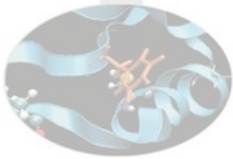
      - Python env

Python 2.7.8 >>>

      - Ipython shell

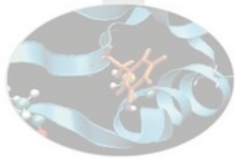
In [1]:

Docker registry



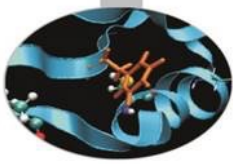
- **Commit** come GIT… ma non solo
- **Push** immagini
- **Pull** di immagini pubbliche
  - …. e.g. Ubuntu :)
  - *FROM:* pull di una immagine se non la trova in locale
- Esistono le “ricette” pronte per la maggior parte dei servizi conosciuti
  - Database
    - postgres, MySQL, mongodb
- Chiunque può creare il proprio Registry locale
  - e.g. azienda fornisce le proprie immagini pronte a qualsiasi computer interno (server o desktop)

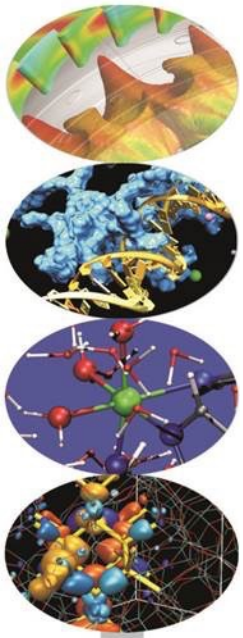
Comunicazione tra i container



- Porte
  - Comando EXPOSE per aprire le porte
- Namespace
  - Links all'avvio di un container per vedere un altro nodo
- File system condiviso
  - VOLUME per montare volumi

CINECA SCAI
SuperComputing Applications and Innovation





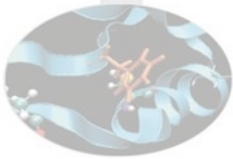
Corso python

ipython

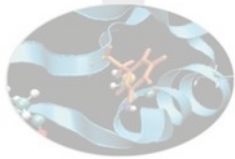
la shell "avanzata"

CINECA

Introduzione ad *IPython*



```
1. root@7c1f5a2054ba: /opt (docker)
(scientific)root@7c1f5a2054ba:/opt# ipython
Python 2.7.8 |Continuum Analytics, Inc.| (default, Aug 21 2014, 18:22:21)
Type "copyright", "credits" or "license" for more information.

IPython 2.3.0 -- An enhanced Interactive Python.
Anaconda is brought to you by Continuum Analytics.
Please check out: http://continuum.io/thanks and https://binstar.org
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

In [1]: a = 3

In [2]: a * 2
Out[2]: 6

In [3]: a
Out[3]: 3

In [4]: type(a)
Out[4]: int

In [5]: a.__str__()
Out[5]: '3'

In [6]: %magic
```

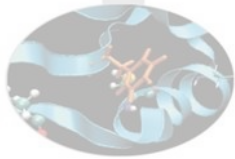


Introduzione ad *IPython*

IPython offre una ricca architettura per il calcolo interattivo che supporta:

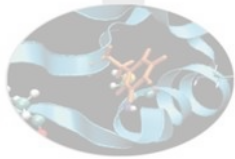
- Una potente shell interattiva (sia terminale che Qt-based)
  - più avanti una ampia dimostrazione :)
- notebook browser-based con il supporto per
  - codice,
  - testo,
  - espressioni matematiche,
  - plot in-line, …
- Visualizzazione grafica interattiva
- Tool ad alte prestazioni per il calcolo parallelo

Ipython: autocompletamento



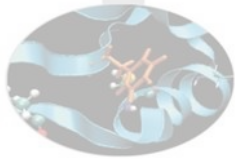
```
(scientific)$ ipython

In [1]: abracadabra = (1,2,3)

In [2]: abr [TAB]
In [2]: abracadabra
```

Ipython: silenziare il comando



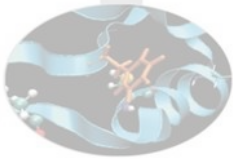
```
In [1]: abracadabra = (1,2,3)

In [2]: abracadabra
Out[2]: (1, 2, 3)

In [3]: abracadabra;

In [4]:
```

Ipython: introspezione



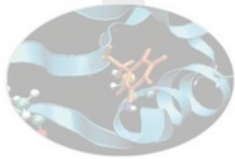
```
In [1]: abracadabra = (1,2,3)
In [2]: whos
Variable      Type      Data/Info
-------------------------------
abracadabra   tuple     n=3
In [3]: pinfo abracadabra
Type:          tuple
String form:   (1, 2, 3)
Length:        3
Docstring:
tuple() -> empty tuple
tuple(iterable) -> tuple initialized from iterable's
items
```

Ipython: introspezione



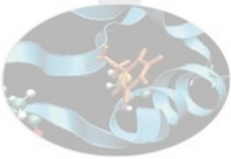
```
In [1]: def incrementa(num):
   ...:     "Aggiungi 1 se minore di 99"
   ...:     if num < 99:
   ...:         num = num + 1
   ...:     return num
   ...:

In [2]: incrementa(81)
Out[2]: 82

In [3]: incrementa(100)
Out[3]: 100
```

Ipython: introspezione



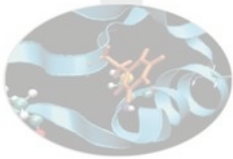
```
In [4]: incrementa?
Type:        function
String form: <function incrementa at 0x7ff0d34799b0>
File:        /opt/<ipython-input-1-03a1b7919e1b>
Definition:  incrementa(num)
Docstring:   Aggiungi 1 se minore di 99

In [5]: incrementa??
Type:        function
String form: <function incrementa at 0x7ff0d34799b0>
File:        /opt/<ipython-input-1-03a1b7919e1b>
Definition:  incrementa(num)
Source:
def incrementa(num):
    "Aggiungi 1 se minore di 99"
    if num < 99:
        num = num + 1
```

Ipython: comandi terminale Linux



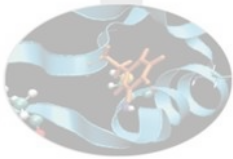
```
In [1]: ls
miniconda/ start_notebook.sh
In [2]: pwd
Out[2]: u'/opt'
In [3]: current_dir = pwd
---------------------------------------------
NameError Traceback (most recent call last)
<ipython-input-9-e6ef9e2ac581> in <module>()
----> 1 current_dir = pwd
NameError: name 'pwd' is not defined
In [4]: !pwd
/opt
```

Ipython: output comandi Linux



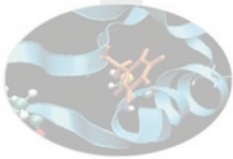
```
In [5]: current_dir = !pwd

In [6]: print "Sono nella directory", current_dir
Sono nella directory [‘/opt']

In [7]: current_dir
Out[7]: [‘/opt']

In [8]: print "Sono nella directory", current_dir[0]
Sono nella directory /opt
```

Ipython: Linux pipe



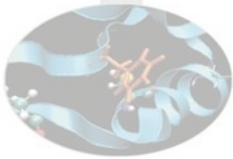
```
In [1]: ls
miniconda/  start_notebook.sh
In [2]: ls -l
total 8
drwxr-xr-x 14 root root 4096 Dec  1 16:39 miniconda/
-rw-r--r--  1 root root   70 Nov 13 14:28 start_note.sh
In [3]: ls -l | wc
      3      20     137
In [4]: ls -l | wc | sed 's/3/4/'
      4      20     137
In [5]: ls -l | wc | sed 's/3/4/g'
      4      20     147
```

Ipython



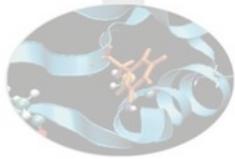
```
In [1]: xrange?
Type:           type
String form:    <type 'xrange'>
Namespace:      Python builtin
Docstring:
xrange(stop) -> xrange object
xrange(start, stop[, step]) -> xrange object

Like range(), but instead of returning a list,
returns an object that generates the numbers in the
range on demand.  For looping, this is slightly
faster than range() and more memory efficient.
```

Ipython: alias



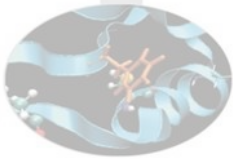
```
In [2]: %alias_magic t timeit
Created `%t` as an alias for `%timeit`.
Created `%%t` as an alias for `%%timeit`.
In [3]: %pdoc range
Class docstring:
    range(stop) -> list of integers
    range(start, stop[, step]) -> list of integers

In [4]: %t -n1 range(1,250,2)
1 loops, best of 3: 5.01 us per loop
In [5]: %t -n1 xrange(1,250,2)
1 loops, best of 3: 4.05 us per loop
```

Ipython: cronologia



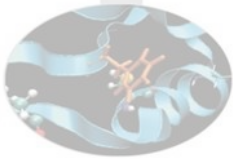
```
In [1]: a = 2
In [2]: b = 4
In [3]: a / b
Out[3]: 0
In [4]: b / a
Out[4]: 2
In [5]: %hist
a = 2
b = 4
a / b
b / a
%hist
```

Ipython: cronologia



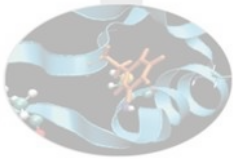
```
In [1]: print "test 1"
test 1

In [2]: print "test 2"
test 2

In [3]: _i
Out[3]: u'print "test 2"'

In [4]: _iii
Out[4]: u'print "test 1"'

In [5]: exec _iii
test 2
```

Ipython: cronologia



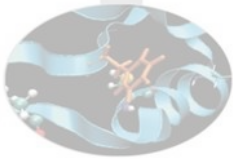
```
In [1]: print "test 1"
test 1

In [2]: print "test 2"
test 2

In [3]: _
Out[3]: u'print "test 2"'

In [4]: _ _ _
Out[4]: u'print "test 1"'
```

Ipython: cronologia



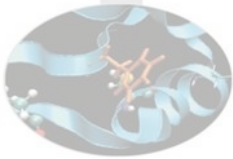
```
In [4]: a + 1
Out[4]: 3

[…]

In [9]: _i4
Out[9]: u'a + 1'

In [11]: %rep 4

In [12]: a + 1
```

Ipython: cronologia



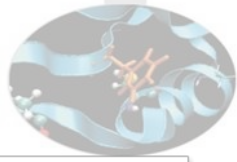
```
In [1]: a = 2
In [2]: a
Out[2]: 2
In [3]: print a
2
In [4]: a + 1
Out[4]: 3
In [5]: print a + 2
4

#output history
In [6]: _oh
Out[6]: {2: 2, 4: 3}
```

Ipython: cronologia



```
In [1]: abc = 3

In [2]: 5 + abc
Out[2]: 8

In [3]: bcd = 4

In [4]: bcd * 3
Out[4]: 12

In [5]: %hist -g abc     #Ricerca in cronologia
   1: abc = 3
   2: 5 + abc
   6: %hist -g abc
```

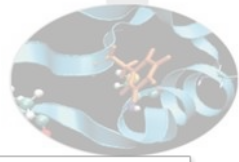
Ipython: logging



```
In [1]: a = 1

In [2]: a = 2

In [3]: a = 3

In [4]: %logstart

Activating auto-logging.
Current session state plus
future input saved.

Filename       :
ipython_log.py

Mode           : rotate

Output logging : False

Raw input log  : False

Timestamping   : False

State          : active
```

```
/opt# more ipython_log.py

# IPython log file


a = 1

a = 2

a = 3

get_ipython().magic(u'logsta
rt')
```

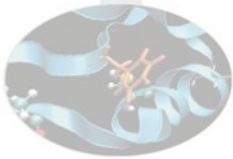
Ipython: logging



```
In [1]: %logstart

In [2]: a = 1

In [3]: a = 2

In [4]: %logoff
Switching logging OFF

In [5]: a = 3

In [6]: %logon
Switching logging ON
```

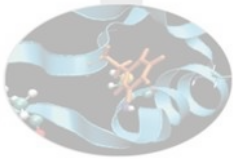
```
/opt# more ipython_log.py
# IPython log file


get_ipython().magic(u'logsta
rt')
a = 1
a = 2
get_ipython().magic(u'logoff
')
```

Ipython: autocall



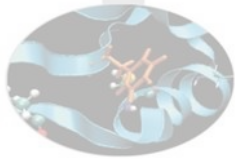
```
In [1]: def p(s):
   ...:     print s
   ...:
In [2]: p("test")
test
In [3]: p(test)
NameError: name 'test' is not defined
In [14]: p "test"
    p "test"
            ^
SyntaxError: invalid syntax
```

Ipython: autocall



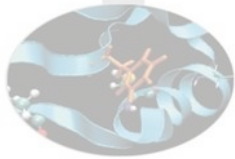
```
In [11]: %autocall
Automatic calling is: Smart

In [12]: p "test"
--------> p("test")
test

In [13]: ,p test
test

In [14]: p
Out[14]: <function __main__.p>
```

Ipython: autocall

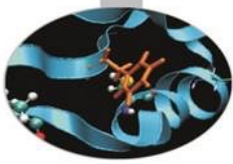


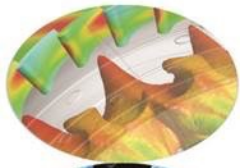
```
In [15]: %autocall 2
Automatic calling is: Full

In [16]: p
-------> p()
TypeError: p() takes exactly 1 argument (0 given)

In [17]: def p(s="vuoto"):
   ....:     print s
   ....:

In [18]: p
-------> p()
vuoto
```

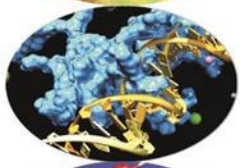
Do the *magic*

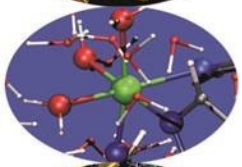


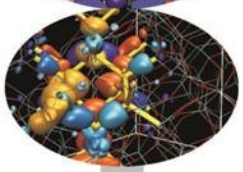
- Su shell ipython:
  - %quickref
- In rete:
  - http://ipython.org/ipython-doc/dev/interactive/magics.html





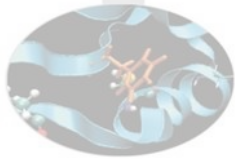






Corso python

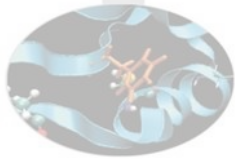
notebook



Avviare il server notebook

```
$ ipython notebook

$ ipython notebook --ip=0.0.0.0 --port=8888 --pylab=inline –no-browser
```

- --*ip* A quale sottorete fornire accesso
- --*port* Su quale porta restare in ascolto per il servizio web
- --*pylab* Insieme da librerie da caricare
  - *“Pre-load matplotlib and numpy for interactive use”*
- --*no-browser* …non aprire il browser all’avvio…



Accesso al notebook

http://192.168.59.103:8888/

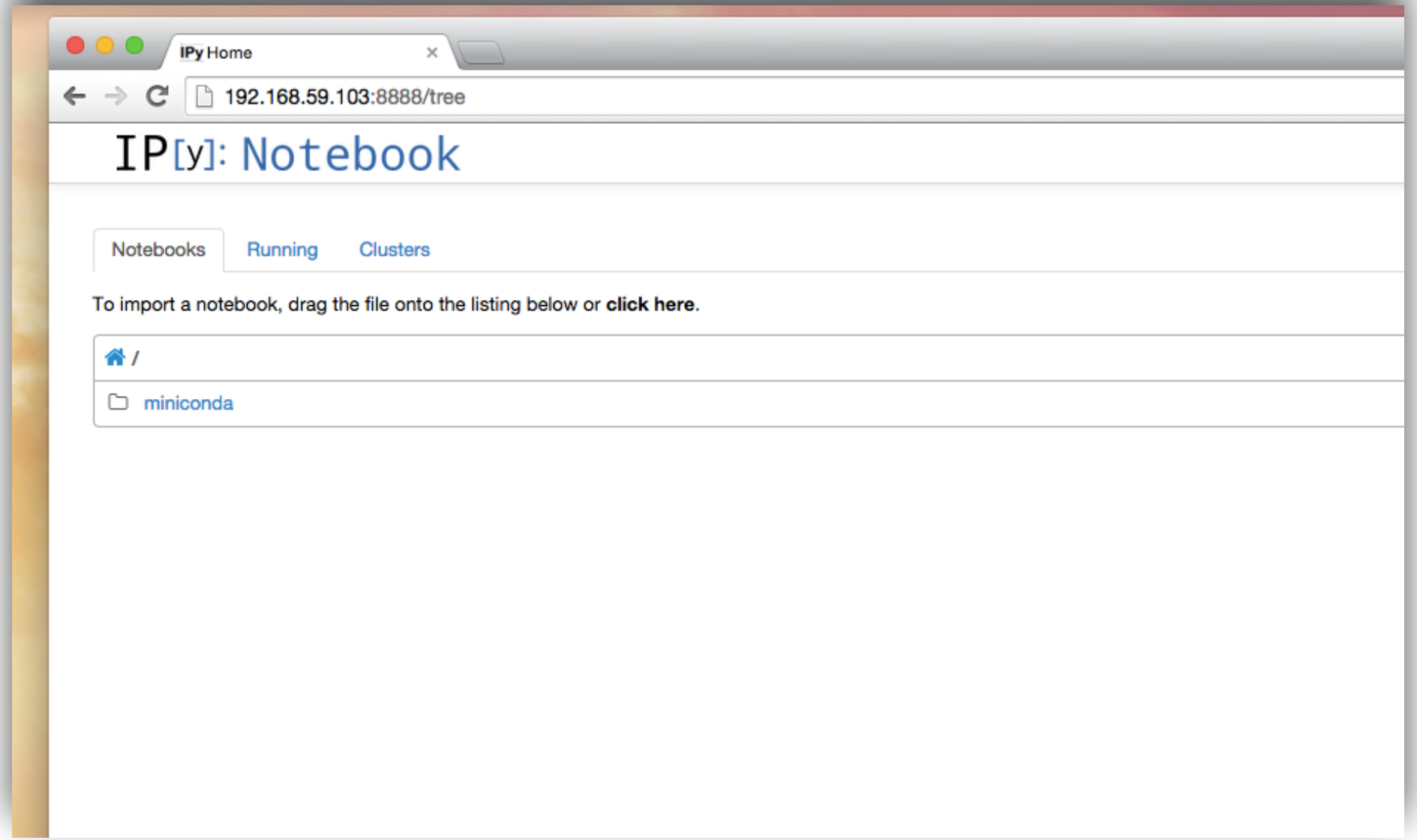
Hint: 192.168.59.103 è l'IP

assegnato alla nostra macchina virtuale "boot2docker"

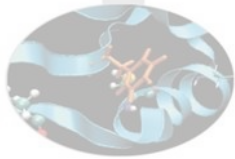
Eseguendo docker da Linux l'accesso sarebbe:

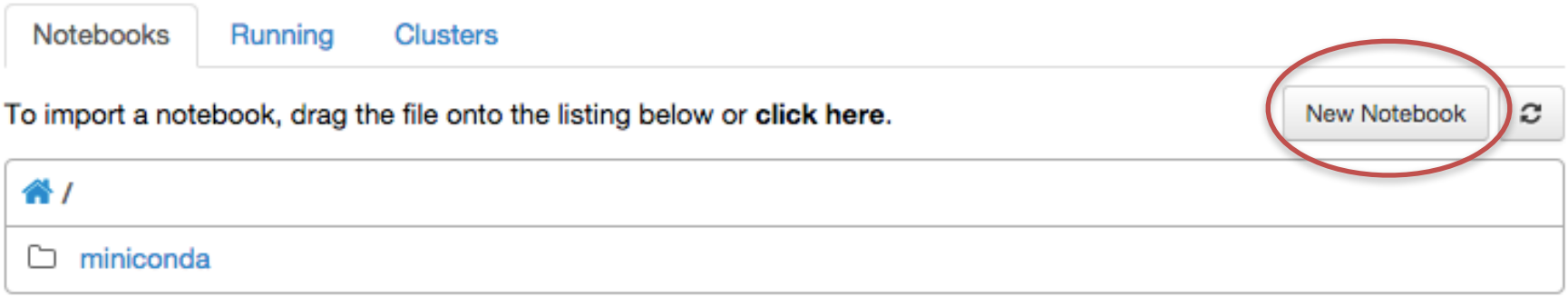
http://localhost:8888

Accesso al notebook

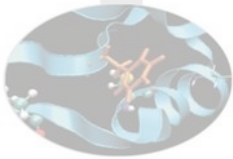


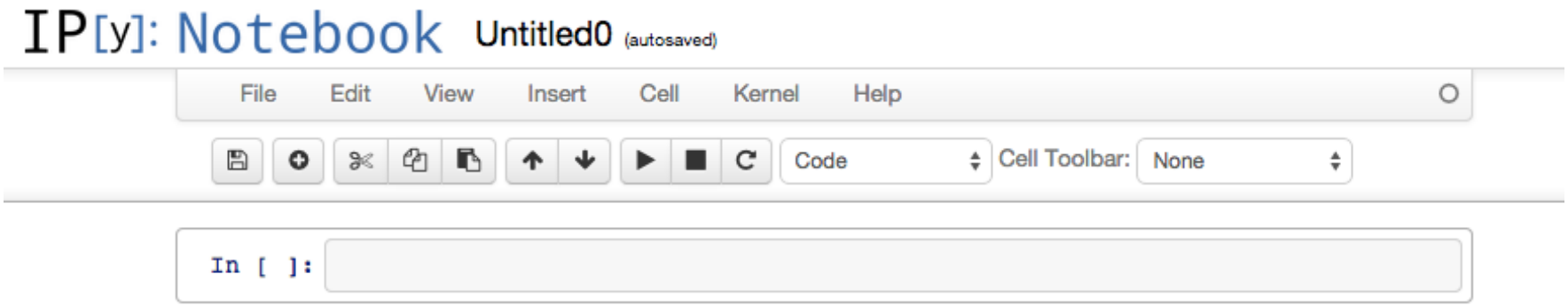
Ipython notebook



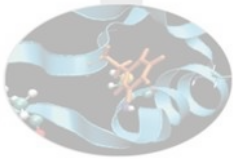


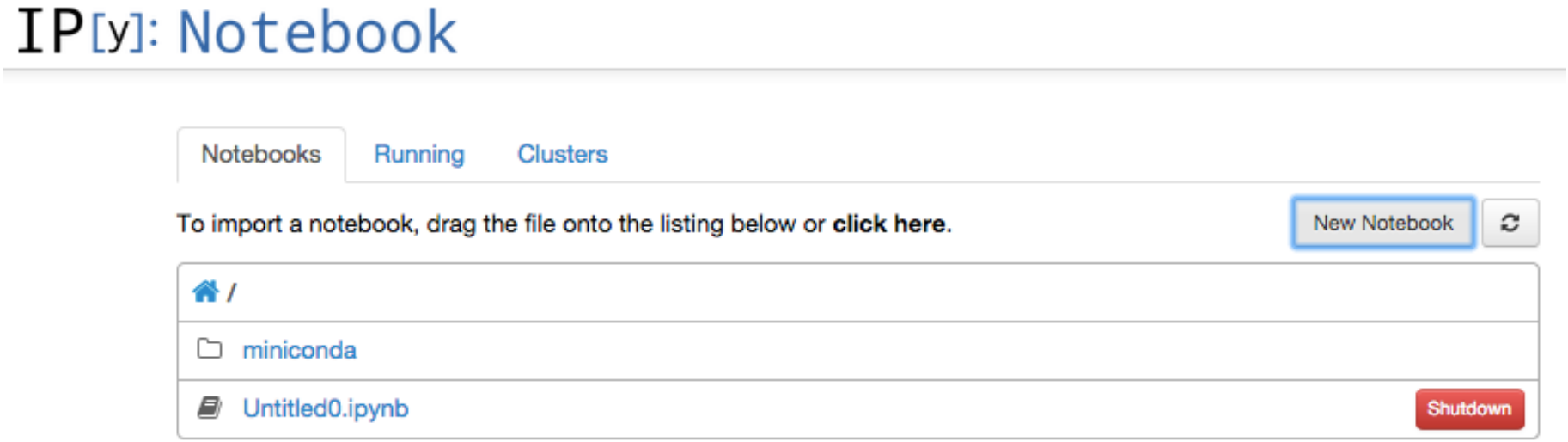
Ipython notebook: new

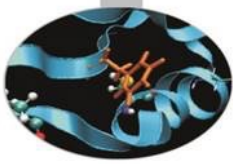




Ipython notebook: list







Prossimo passo:

approfondimento
sugli aspetti scientifici del calcolo