CAMD Summer School 2024

Electronic structure tools: ASE and GPAW
and introduction to the computer projects

Jakob Schiøtz, DTU Physics

Overview

- The Python language
- The Atomic Simulation Environment (ASE)
  - The anatomy of an atomic-scale simulation/calculation
  - The ASE
- Examples
  - Almost the simplest possible molecular dynamics simulation.
  - Almost the simplest possible GPAW calculation.
- Using the DTU “databar” (computer lab).
- Computational projects

Part I

Brief Introduction to Python

Why Python?

- For the programmer: Python is object-oriented
  - Object oriented and modular: Facilitates writing and maintaining complex problems.
  - Dynamically typed: Flexibility, facilitates code reuse.
  - Easy to write readable code: Code is maintainable.
  - Large libraries available (numerics, plotting, …)
- For the user: Python is a scripting language.
  - Great for scripting a calculation
  - Great for small programs and prototypes.
  - Great for interactive experimenting.
  - Easy to learn.
  - Objects are powerful in scripts!
- Python can be extended in C/C++/Fortran
  - Solves performance problems of non-compiled languages.

Learning Python

- Don’t waste money on Python books!
  - It’s not that hard, and online docs are good.

- Python tutorial: http://docs.python.org/3/tutorial/
  - More documentation at docs.python.org

- Learn Python + ASE + GPAW by example
  - Get a simple script, and modify it.
  - Simple scripts are almost like old-fashioned input files!

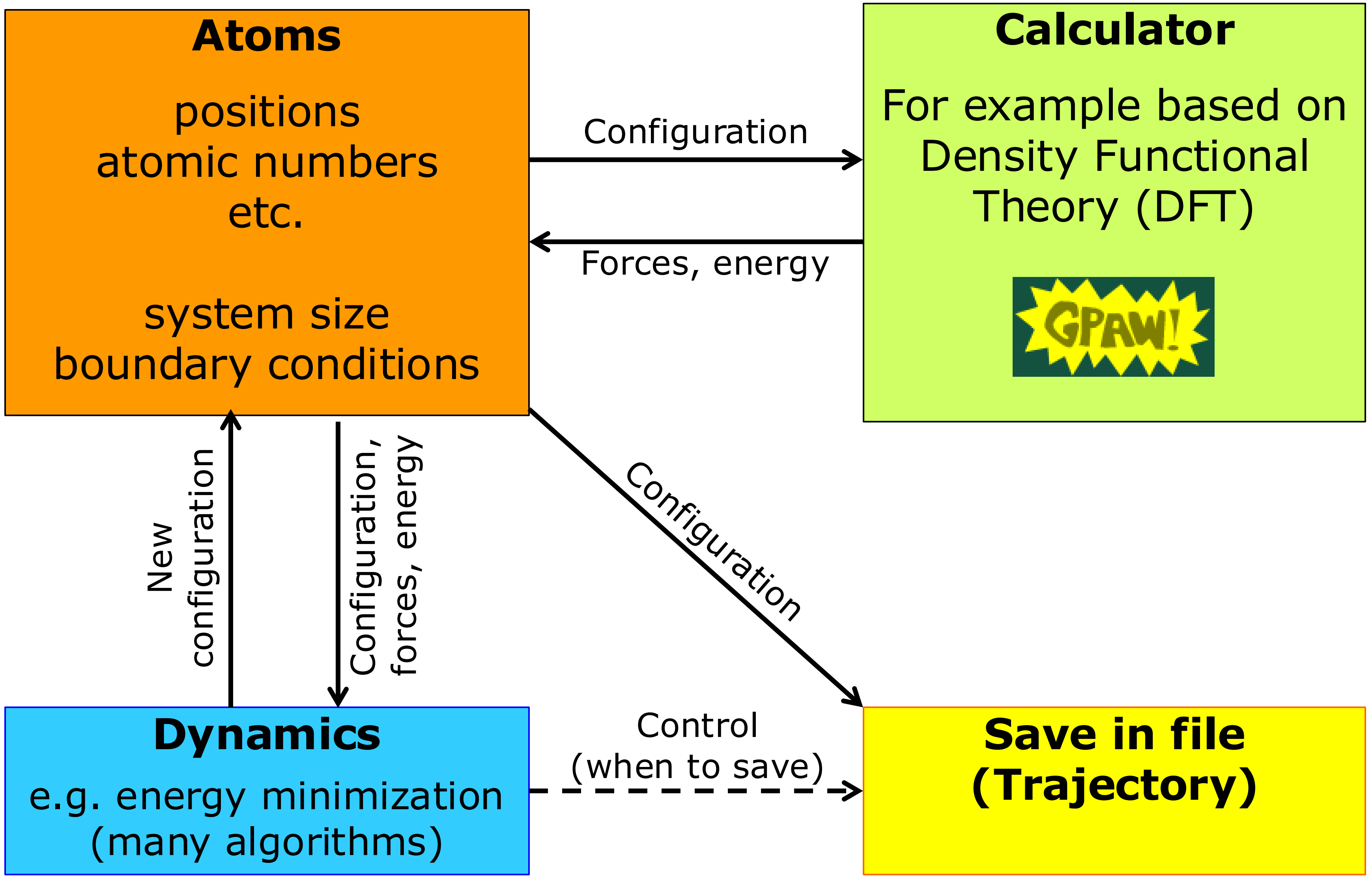
A few places where Python is different

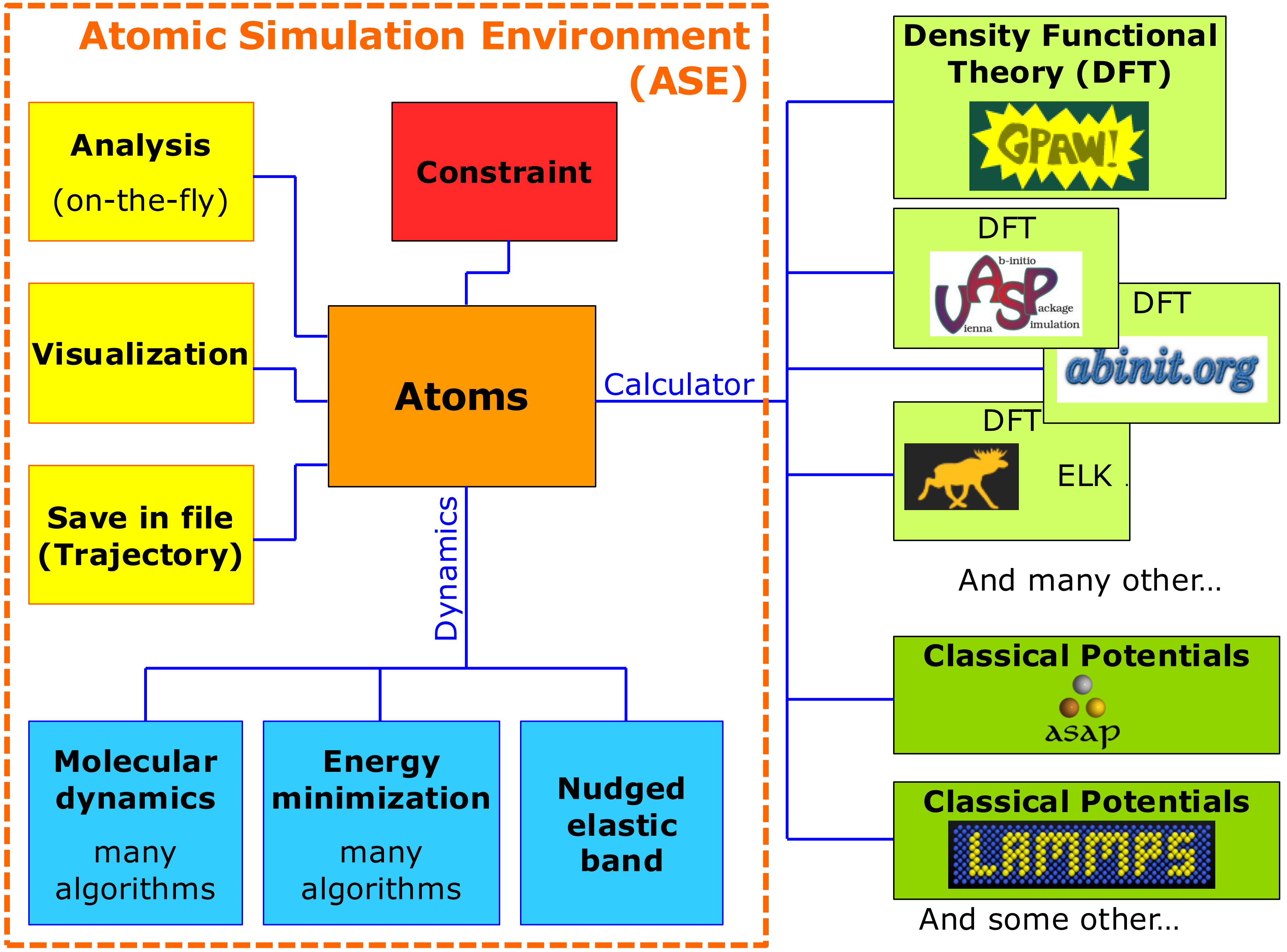
- See notebook!

Part II

INTRODUCTION TO ASE AND GPAW

The "anatomy" of an atomic-scale computer simulation




Atomic Simulation Environment (ASE)
Analysis
(on-the-fly)
Constraint
Visualization
Atoms
Calculator
Save in file
(Trajectory)
Dynamics
Molecular dynamics
many algorithms
Energy minimization
many algorithms
Nudged elastic band
Density Functional Theory (DFT)
GPAW!
DFT
b-initio
ackage
ienna
imulation
DFT
abinit.org
DFT
ELK
And many other...
Classical Potentials
asap
Classical Potentials
LAMMPS
And some other...

Example: Molecular dynamics

```python
import numpy as np
from ase import Atoms, units
from ase.io.trajectory import Trajectory
from ase.build import bulk
from ase.md.verlet import VelocityVerlet
from asap3 import EMT

# Create the atoms
atoms = bulk("Cu", "fcc", cubic=1).repeat([3,3,3])
atoms.set_pbc(False)
atoms.center(vacuum=5.0)
# Give the first atom a non-zero momentum
atoms[0].momentum = np.array([0, -11.3, 0])

# Prepare to do molecular dynamics, forces described by EMT
atoms.calc = EMT()
dyn = VelocityVerlet(atoms, 5.0*units.fs)

# Make a trajectory writing output every fifth timestep.
trajectory = Trajectory("MD-output.traj", "w", atoms)
dyn.attach(trajectory, interval=5)

# Now do 1000 timesteps.
dyn.run(1000)
```

GPAW Example: Atomization energy of Hydrogen (1/2)

```
from ase import Atoms, Atom
from gpaw import GPAW

a = 4.  # Size of unit cell (Angstrom)
c = a / 2
# Hydrogen atom:
atom = Atoms('H',
             positions=[(c, c, c)],
             magmoms=[1],
             cell=(a, a, a))

# gpaw calculator:
calc = GPAW(h=0.18, nbands=1, xc='PBE', txt='H.out')
atom.calc = calc

e1 = atom.get_potential_energy()
calc.write('H.gpw')
```

Continued…

GPAW Example: Atomization energy of Hydrogen (2/2)

Continued …

```
# Hydrogen molecule:
d = 0.74  # Experimental bond length
molecule = Atoms('H2',
                 positions=([c - d / 2, c, c],
                            [c + d / 2, c, c]),
                 cell=(a, a, a))

calc.set(txt='H2.out')
molecule.calc = calc
e2 = molecule.get_potential_energy()
calc.write('H2.gpw')

print 'hydrogen atom energy:     %5.2f eV' % e1
print 'hydrogen molecule energy: %5.2f eV' % e2
print 'atomization energy:       %5.2f eV' % (2 * e1 – e2)
```

The Atoms object

The `Atoms` object is the main simulation object. It contains:

- Per-atom data: positions, velocities, charges, magnetic moments, tags.
- Global data: Unit cell, boundary conditions.
- Refs to helper objects: Calculator, constraints, …

Accessing the data:

`r = atoms.get_positions()` **or** `r = atoms.positions`

`atoms.set_positions(r)` **or** `atoms.positions = r`

`f = atoms.get_forces()` Will trigger a calculation, if needed.

The Atoms object

Manipulating the `Atoms` object:

`atoms.center(vacuum=5.0)`	Centers in unit cell, possibly adjusting the amount of vacuum.
`atoms.rotate(45, ’z’)`	Rotate the atoms
`atoms.repeat([2,2,1])`	Replicate atoms along x,y,z axes.

The `Atoms` object is a Python sequence:

`atoms[i]`	The ith atom (starting at zero).
`atoms += atoms2`	Add more atoms
`atoms1 + atoms2`	Merge two atoms objects
`for a in atoms: print(a.position)`	Loop over atoms

```
for a in atoms:
    print(a.position)
```

Making systems

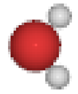
Manually specify the atoms:

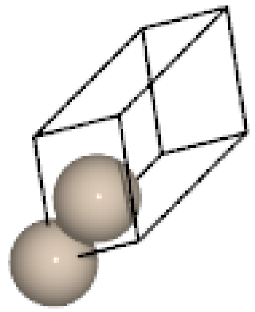
```
from ase import Atoms
h2 = Atoms('H2', positions=[(0, 0, 0), (0, 0, 0.74)])
```

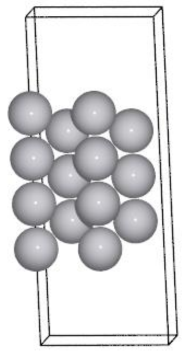
Known molecules and bulk structures:

```
from ase.build import molecule, bulk

water = molecule('H2O')
si2 = bulk('Si', 'diamond', a=5.4)
```







Simple surfaces:

```
from ase.build import fcc110
slab = fcc110('Pt', (2, 1, 7), a=4.0, vacuum=6.0)
```

GPAW essentials

- **Grid mode** – wavefunctions on a real-space grid

  ```
  calc = GPAW(h=0.18, xc='PBE', …)
  ```

  Parallelizes very well for large systems. High accuracy

- **Plane wave mode** – wavefunctions in k-space

  ```
  calc = GPAW(mode=PW(400), xc=…,   …)
  ```

  Faster than grid mode for small/medium systems. High accuracy

- **LCAO mode** – wavefunctions in real space; with basis fct.

  ```
  calc = GPAW(mode='lcao', basis='dzp', h=0.18, …)
  ```

  Fast but less accurate.

Part III

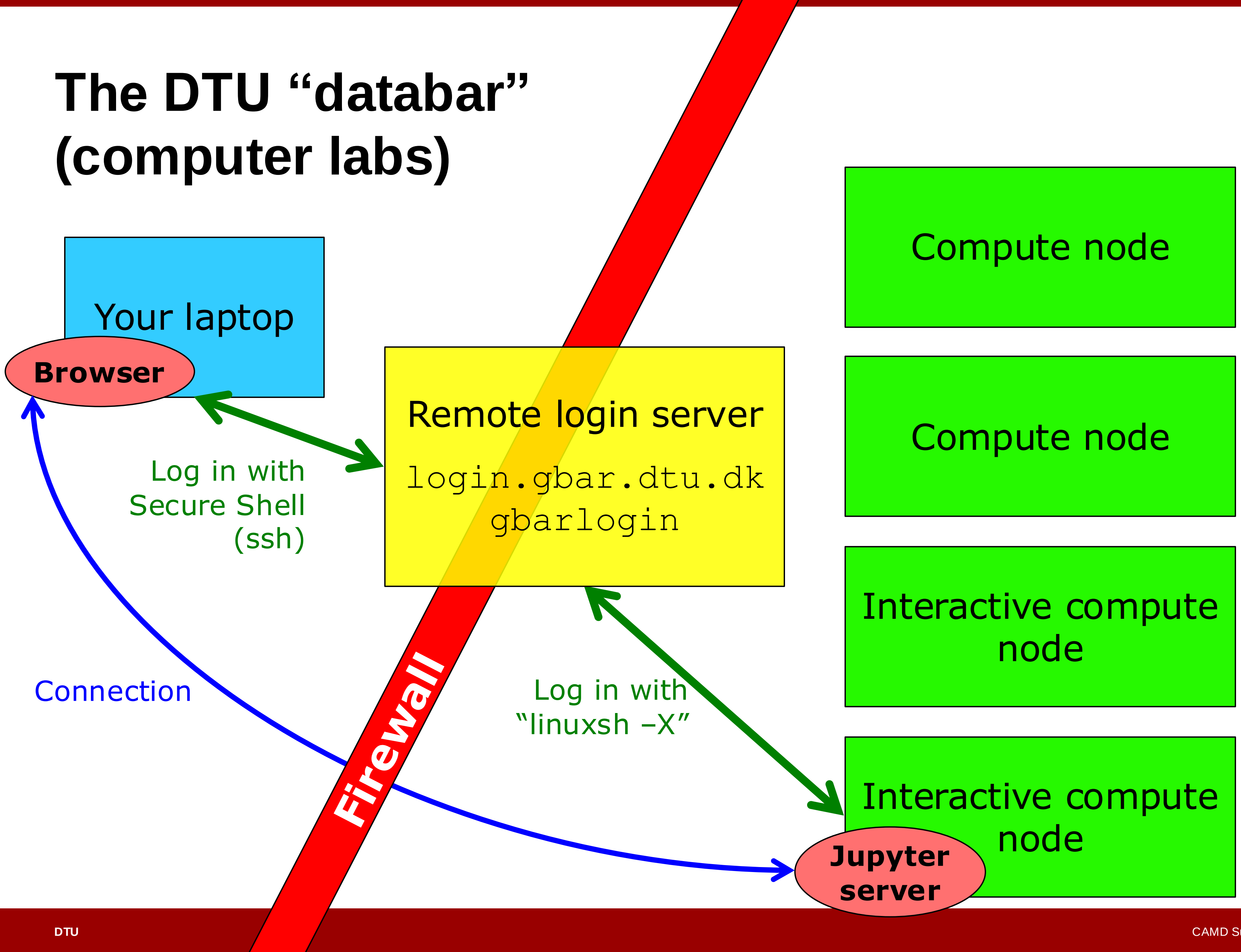
Computer Exercises and the DTU “Databar”

Increased IT security at DTU

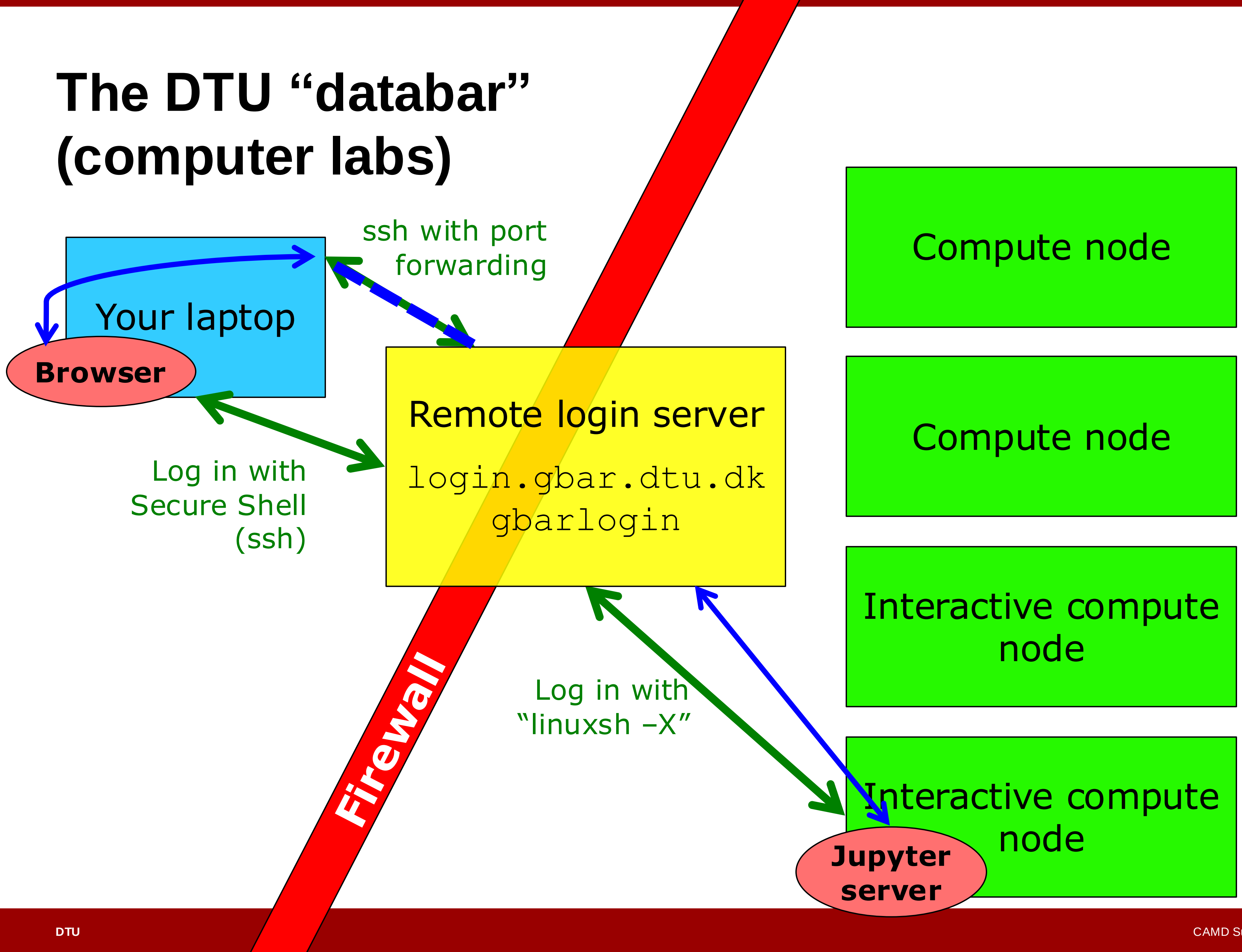
- All external log in to DTU systems require two-factor login
- In our case the two factors are
  - Your password
  - A cryptographic key on your laptop

Your laptop
Secret key
DTU server
Public key

The DTU "databar" (computer labs)



The DTU "databar" (computer labs)



The “databar” and the projects

- Information about the Summerschool projects:

  wiki.fysik.dtu.dk/gpaw/

  Includes detailed instructions on how to log in.

- Changing password in the HPC system (“databar”):
  - Change at password.dtu.dk (takes up to 1 hour to sync)
  - Download your SSH Secret key with original password!

- **SLIDES** available from the GPAW summerschool pages.

Projects

- Excited states
  - Jakob Svaneborg and Jiban Kangsabanik
- Catalysis
  - Georg Kastlunger and Dipam Patel
- Magnetism
  - Martin Ovesen and Varun Rajeev Pavizhakumari
- Batteries
  - William Hansen and Lotte Kortstee
- Machine Learning
  - Jesper Rask Pedersen and Armando Morin Martinez
- Computational workflows
  - Ask Hjort Larsen and Tara Boland

Project: Excited states

- Calculate electronic band structures and bandgaps
- Calculate optical absorption

- Special methods:
  - The GW approximation
  - The Random Phase approximation
  - Optional: The Bethe-Salpeter Equation

Project: Catalysis

- DFT calculations of adsorption geometries of molecules on surfaces.
- Calculations of the reaction path and the transition energy

- Special methods:
  - Nudged Elastic Band (for transition paths)
  - Optional: Vibrational analysis for adsorption entropy

Project: Magnetism in 2D

- Calculations of the critical temperature of a CrI_3 monolayer
- Calculations of the noncollinear ground state in VI_2
- Autodiscovery of new magnetic monolayers in the Computational 2D Materials Database (C2DB)

- Special methods:
  - Energy mapping analysis
  - Noncollinear DFT calculations

Project: Batteries

- Study the anode and cathode materials of a Li-ion battery with DFT
- Calculate the intercalation energy of Li in graphite, establish the equilibrium potential of a $LiFePO_4$/C battery and determine important battery characteristics such as Li transport barriers and the voltage profile.

- Special methods:
  - Structure creation and modification with ASE
  - Unit cell relaxation
  - Bayesian error estimation
  - Nudged Elastic Band (NEB) calculations for estimating Li migration barriers

Project: Machine Learning

- Machine Learning methods used on an example database
- Predicting band gaps and heat of formation using a structure “fingerprint”
- Predictions with several methods (e.g., ridge regression, decision tree, and gaussian process)
- Validation of predictions using DFT

Project: Computational Workflows

- Learn to define tasks and workflows with TaskBlaster
- Use command-line tools to create and manage tasks in a directory tree
- Run a materials workflow on multiple materials