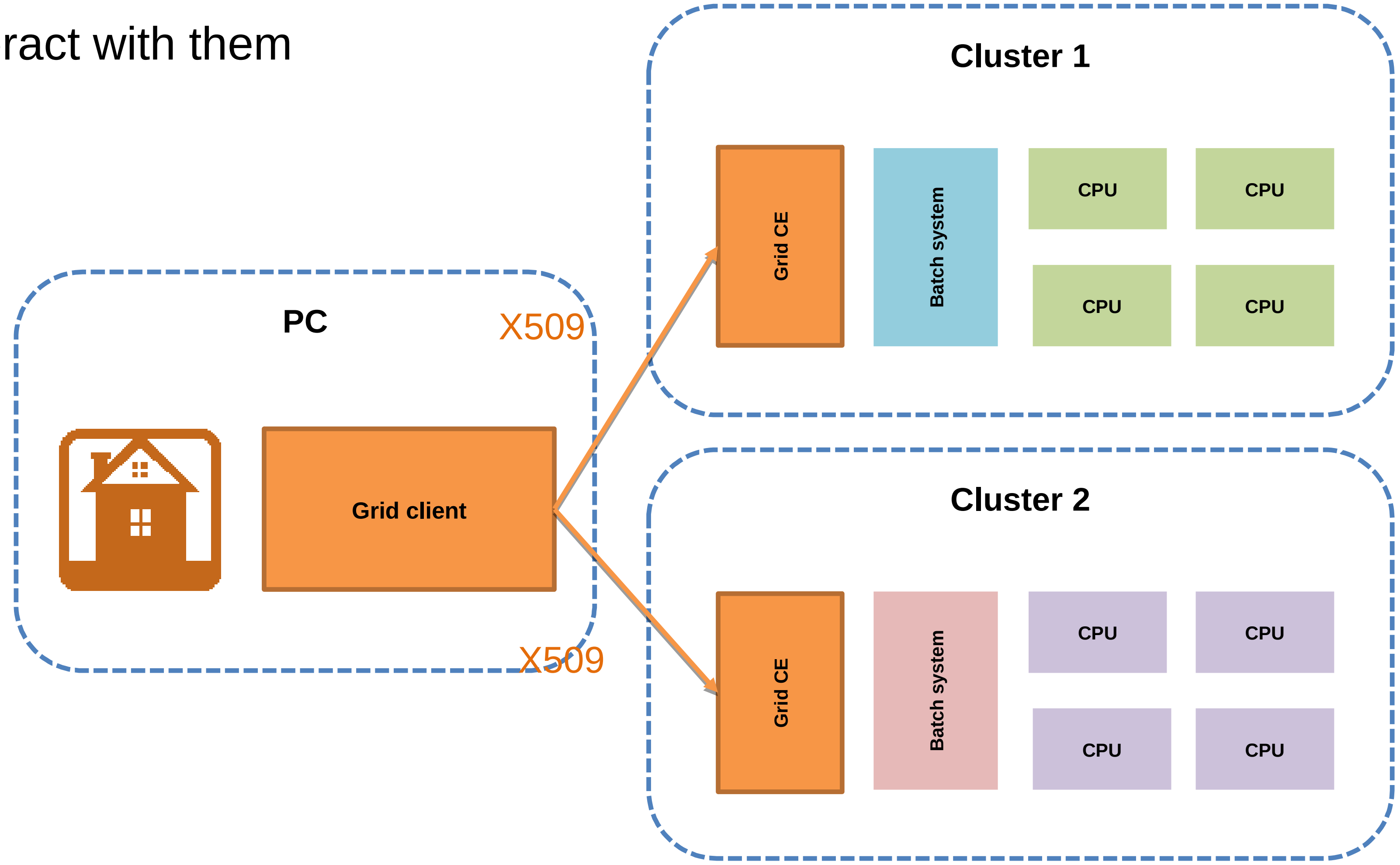
ARC CE Tutorial

- NTF004F2019
  Florido Paganelli
  florido.paganelli@hep.lu.se

The Grid

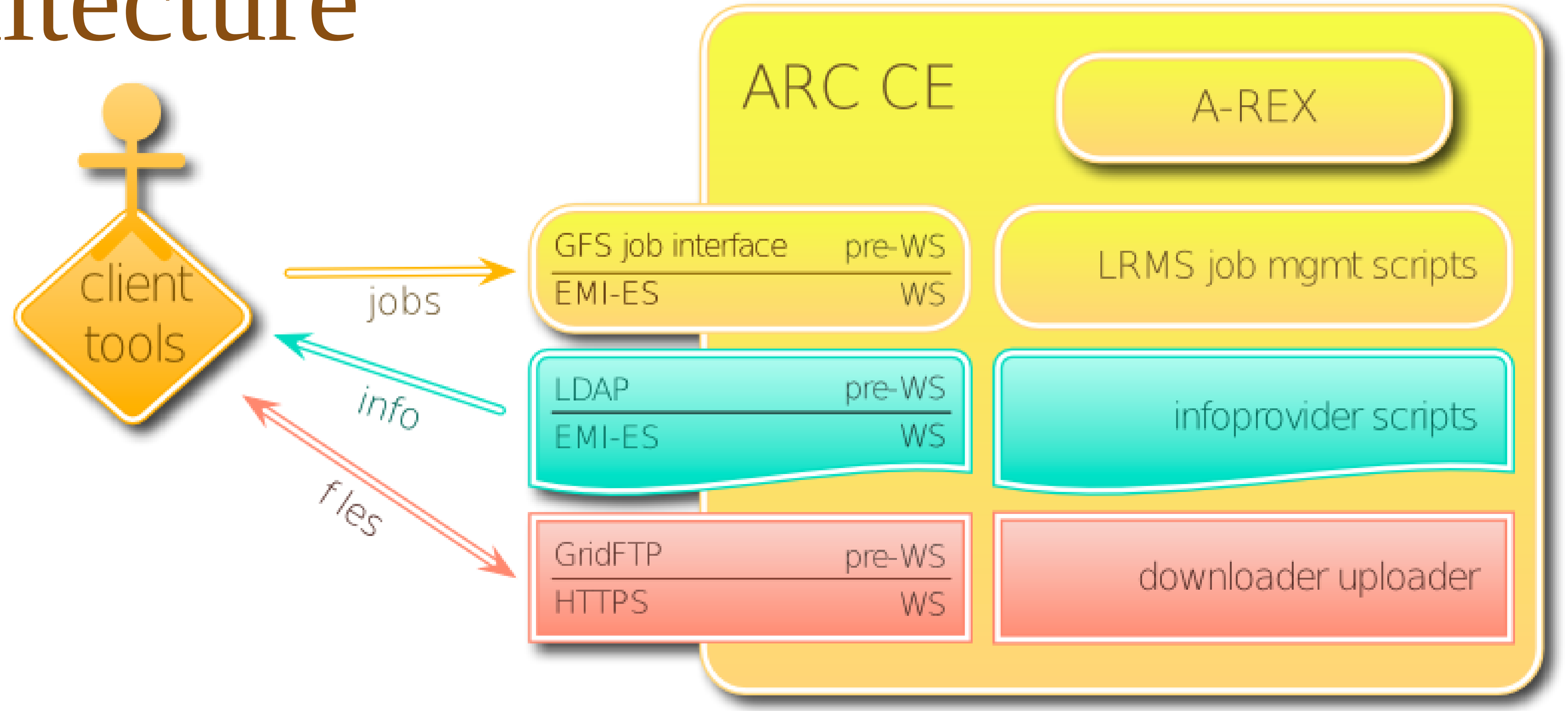
- Built when computing hardware was expensive to federate multiple computing centers. Focused on a single computing task, a “job”.
- Mainly three components:
  - HPC/HTC clusters of computers
  - Middleware sitting in front of HPC/HTC batch system software, usually called Grid Computing Element (CE)
  - A client to interact with them



Why ARC

- **Advanced Resource Connector**
- Invented in the Nordics, developed in the Nordics and eastern Europe area
- Today the only survivor of the grid era. Other project slowly disappearing. New project like HTCCondor have different features and purposes.
- It’s open source
- Intensively developed
- HEP community/CERN the main users, but also used in biology and other areas in countries with not enough funds for top-notch datacenters
- Version 6 complete review to be more cloud-friendly

Architecture

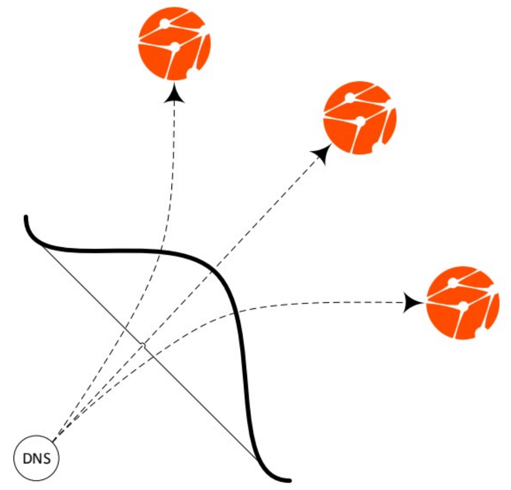


- Written in C++, Python, Perl, Bash
- Web services container called HED (Hosting Environment Daemon), runs A-REX, the ARC Remote EXecution service
- A-REX starts/controls the various services (job management, information system, data management)
- External software: LDAP server + bdb, sqlite, openssl
- One single configuration file for everything
- Deployment: available in most popular linux distributions package managers
- Code: https://github.com/nordugrid/arc

What does ARC do

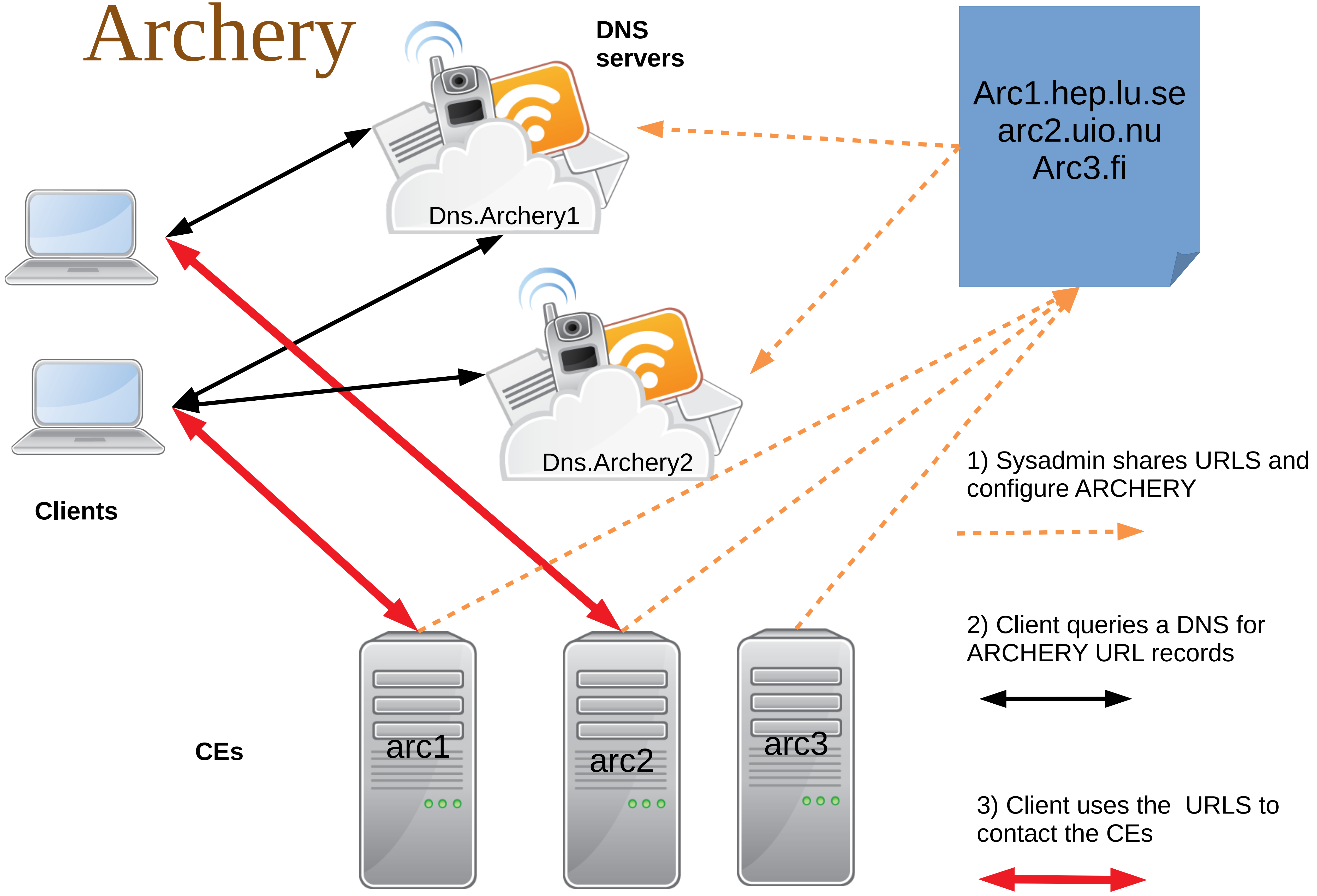
- **A-REX Computing Element (CE).** Middleware server that provides services/tools for federated
  - Authentication and authorization
  - Accounting
  - Information on job status and resource discovery
  - Data movement/staging
  - Job state manipulation
- **ARC** Clients
  - Provide a unified way of specifying a job, regardless of the batch system
  - Abstract ways of checking/controlling the job state
  - Job results retrieval
  - Both **CLI** and **API** to create custom clients
- **ARCHERY** Index
  - Gathers together various CEs

Archery



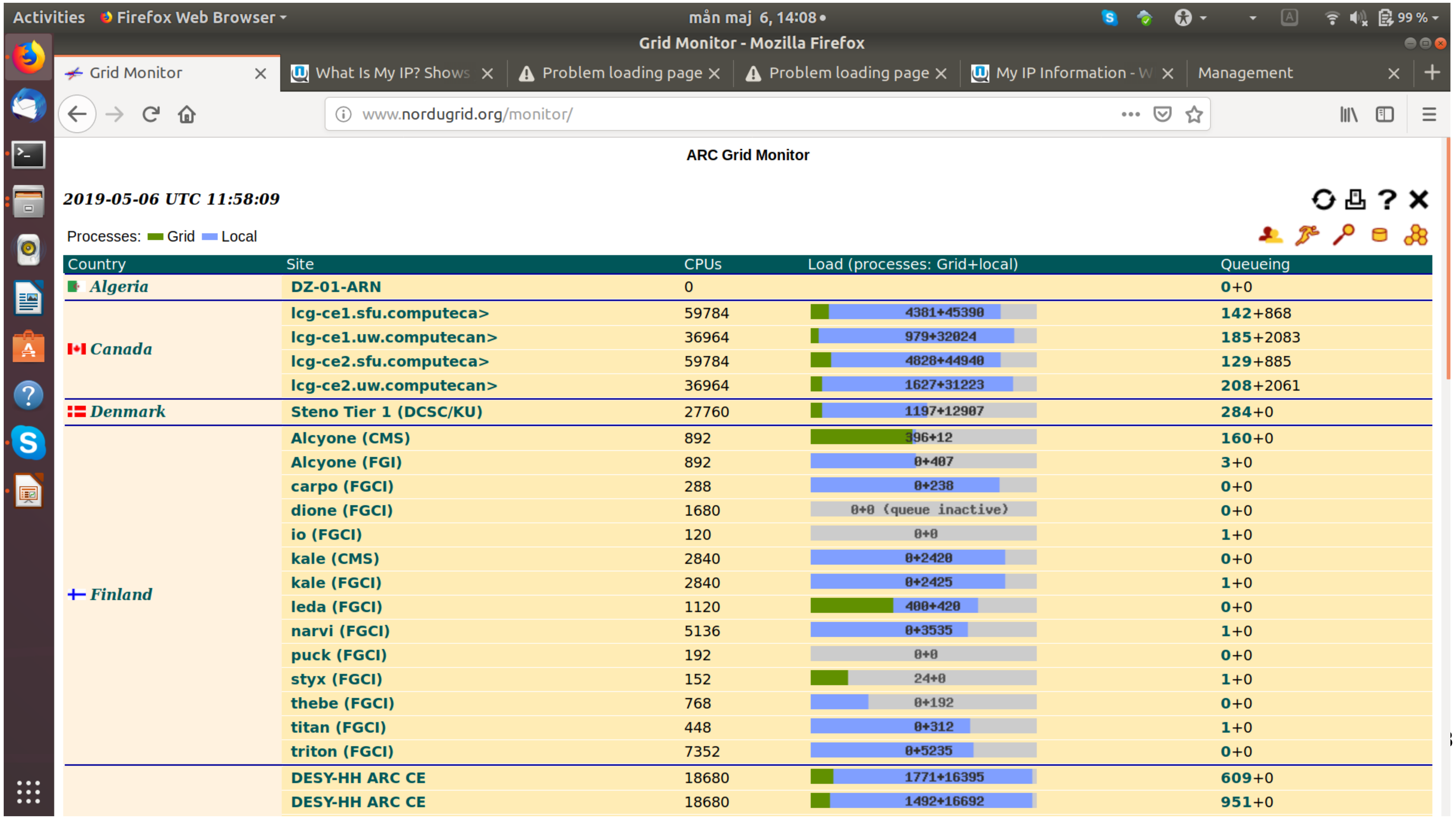
- Archery is ARC’s novel index service
- It connect various CEs in a distributed fashion, but can be centralized as well.
- It’s based on DNS (Domain Name Server), the system used on the internet to associate a human-readable machine name (hostname and domain name) to a machine-readable network addres (IP address number)
  example: check https://www.whatismyip.com/
- Configuration requires no intervention by the sysadmin on a grid CE but sending the information about their hostname/ip address to the DNS manager
- An index admin is responsible to create the index on DNS server.

Archery


DNS servers
Dns.Archery1
Dns.Archery2
Arc1.hep.lu.se
arc2.uio.nu
Arc3.fi
Clients
CEs
arc1
arc2
arc3
1) Sysadmin shares URLS and configure ARCHERY
2) Client queries a DNS for ARCHERY URL records
3) Client uses the URLS to contact the CEs

The infosys at work

- You can see a client for the whole infosys deployed at nordugrid.org:
  http://www.nordugrid.org/monitor



Grid Monitor - Mozilla Firefox

www.nordugrid.org/monitor/

ARC Grid Monitor

2019-05-06 UTC 11:58:09

Processes: 🟩 Grid 🟦 Local

Country	Site	CPUs	Load (processes: Grid+local)	Queueing
Algeria	DZ-01-ARN	0		0+0
Canada	lcg-ce1.sfu.computeca>	59784	4381+45390	142+868
	lcg-ce1.uw.computecan>	36964	979+32024	185+2083
	lcg-ce2.sfu.computeca>	59784	4828+44940	129+885
	lcg-ce2.uw.computecan>	36964	1627+31223	208+2061
Denmark	Steno Tier 1 (DCSC/KU)	27760	1197+12907	284+0
Finland	Alcyone (CMS)	892	396+12	160+0
	Alcyone (FGI)	892	0+407	3+0
	carpo (FGCI)	288	0+238	0+0
	dione (FGCI)	1680	0+0 (queue inactive)	0+0
	io (FGCI)	120	0+0	1+0
	kale (CMS)	2840	0+2420	0+0
	kale (FGCI)	2840	0+2425	1+0
	leda (FGCI)	1120	400+420	0+0
	narvi (FGCI)	5136	0+3535	1+0
	puck (FGCI)	192	0+0	0+0
	styx (FGCI)	152	24+0	1+0
	thebe (FGCI)	768	0+192	0+0
	titan (FGCI)	448	0+312	1+0
	triton (FGCI)	7352	0+5235	0+0
	DESY-HH ARC CE	18680	1771+16395	609+0
	DESY-HH ARC CE	18680	1492+16692	951+0

Today’s exercise TODO

- Fetch a minimalist docker container with all the needed to run arc.
- Install ARC in your machines, in the docker container
- Test your setup by submitting locally
- Understand arc configuration
- Understand network requirements
- If there is time, create a grid with your installations:
  - Configure security
  - Configure client
  - I will manage the index
  - You will try to submit to each other’s machine

Step one: prepare the container 1/2

- Create a folder where you will store the exercise file. For example:

```
mkdir ~/L8
cd L8
```

- Create a folder that will be used to share data among the container and the host.

```
mkdir tutorial8
```

- For practical reasons, I prepared a Centos7 image with systemd so that the environment is similar to a virtual machine.
  - Get the container from docker hub:

    ```
    docker pull floridop/c7-systemd:L8
    ```

- Start the container with the following:

```
docker run -d --privileged -it --name arc6tut \
-v /sys/fs/cgroup:/sys/fs/cgroup:ro \
-v /home/pflorido/L8/tutorial8:turorial8 -p 443:443 \
floridop/c7-systemd:L8
```

  - You can check that is running with `docker ps`

- Enter the command line in the container with

```
docker exec -it arc6tut /bin/bash
```

Step one: prepare the container 2/2

- If you need to restart the container during the exercise (unlikely), always use the container name *arc6tut*.
- Anything you will do in the container will not be updated in the original image. To save your progress as image you need to use the `docker commit` command. I'll show you an example later
  - For now: don't delete the container during the tutorial!

Step two: ARC zeroconf install

- Follow instructions in http://www.nordugrid.org/arc/arc6/admins/try_arc6.html
- To install the packages refer to the page:
  http://www.nordugrid.org/arc/arc6/common/repos/repository.html
  where:
  - We will use the **release-candidate** version
  - We will **NOT** use the latest nightly build.
  - CentOS7 is a Red-Hat based distribution. To install the repositories, follow the instructions for **red-had based distributions**.
  - Use **yum** with the **EL7** link.
  - You do NOT need to configure the repository manually.
  - We will need packages in the *nordugrid-testing* repository
- The only package we will use for the moment is **nordugrid-arc-arex**
- Follow the instructions at the link at the top (From Step 3)

Step two: ARC zeroconf install cont.

- Whenever you see:
  - `[root ~]#` means you must be user "root"
  - `[user ~]$` means you must be user "user01"
    - You can become **user01** by issuing
      `su - user01`
      - Go back to root by issuing `exit`
    - You can check which user you are with
      `whoami`
- You can discover your hostname by writing
  `hostname`

Step three: understanding configuration

- Examine the /etc/arc.conf file
- All members of the group zero are contained in the testCA.allowed-subjects list, which contains the generated identity
- All these members are mapped to the user nobody:nobody when executing jobs
- For each user there is a reference CA in the certificates folder
- Start a-rex again:
  systemctl start a-rex

Step four: networking

- Configure docker for external networking (already done, the -p 443:443 option)
- Open ports in system firewall
  in ubuntu
  ```
  sudo ufw allow in from any to any port 443 proto tcp
  ```
- Discover your current ip address/hostname
  - check www.whatismyip.com
  - Command lines:
    - Linux:
      ```
      ip addr show
      hostname
      ```
    - Windows:
      ```
      ipconfig /all
      ```
- Test connectivity from outside.
  ping <someone’s IP/hostname>
  Eduroam a blocker, we will stop here.

Docker commit

- Login to the container
- Create a file in the container and exit

```
touch TESTFILE
ls testfile
exit
docker stop arc6tut
```

- Start another container based on the same image:

```
docker run -d --privileged -it --name arc6tut2 \
-v /sys/fs/cgroup:/sys/fs/cgroup:ro c7-systemd:L8
```

- Login to the container and list files

```
docker exec -it arc6tut2 /bin/bash
ls
```

  - The file is gone!

- Now repeat the file creation and exit.
- Run `docker container ls`

```
CONTAINER ID    IMAGE           COMMAND           CREATED          STATUS          PORTS          NAMES

7ead4da6225b    c7-systemd:L8   "/usr/sbin/init"  2 minutes ago    Up 2 minutes    443/tcp        arc6tut2
```

  - The container with the changed content has a new ID *hash*. One can tell docker to incorporate the changes in the image by issuing:
    `docker commit <hash>`
  - `Docker image ls` shows the image with the new hash. You can start a new container using the hash for that image

```
 docker run -d --privileged -it --name arc6tut3 -v
/sys/fs/cgroup:/sys/fs/cgroup:ro <hash>
```

References

- ARC6 documentation
  http://www.nordugrid.org/arc/arc6