# DESY Implementation and Usage of the HTCondor Batch System:

The talk provides an overview of the DESY configurations for HTCondor. It focuses on features we need for user registry integration, node maintenance operations and fair share / quota handling. We are working on Docker, Jupyter and GPU integration into our smooth and transparent operating model setup.

**DESY/IT-Systems:**
Thomas Finnern
Christoph Beyer
Martin Flemming
Yves Kemp

HELMHOLTZ ASSOCIATION

HEPiX BROOKHAVEN NATIONAL LABORATORY

Nikhef

HTCondor High Throughput Computing

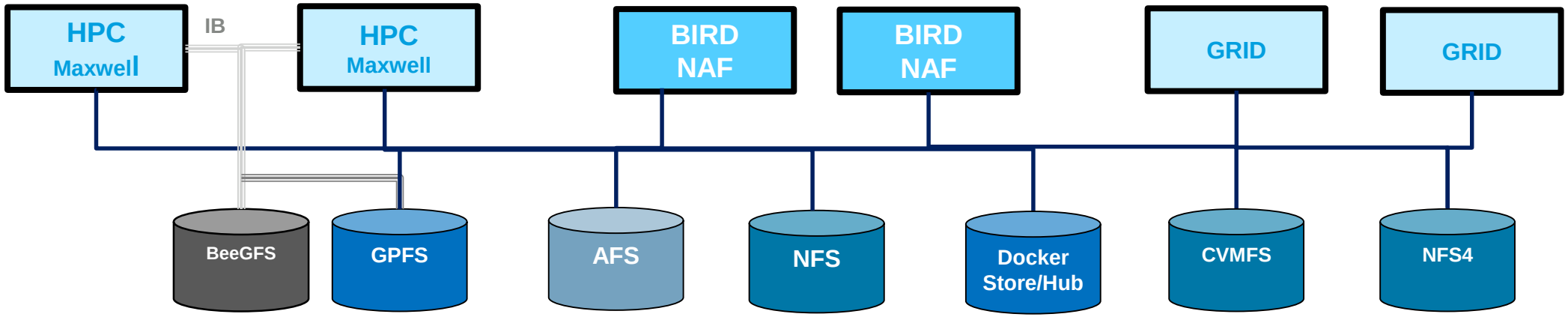
DESY

# Overview DESY Batch Infrastructure !

## Basic Blocks

| High Performance Computing | BIRD Computing | Test Cluster | Grid Computing | CE |
|---|---|---|---|---|
| | | | | ARC CE |

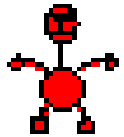




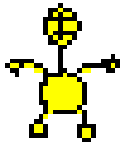
# Outline of Talk

## HTCondor DESY Environment

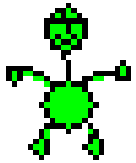
- Main Focus on BIRD Facility
- BIRD/NAF Overview



- User Perspective

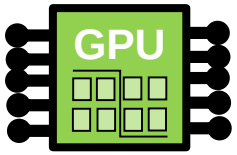


- Administrator aspects

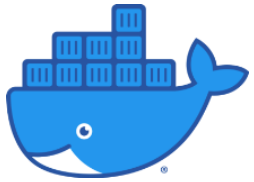


- Operator Support

- Auth Topics







- New Features
  - GPU Support
  - Jupyter Notebook Integration
  - Docker Service
  - Function as a Service

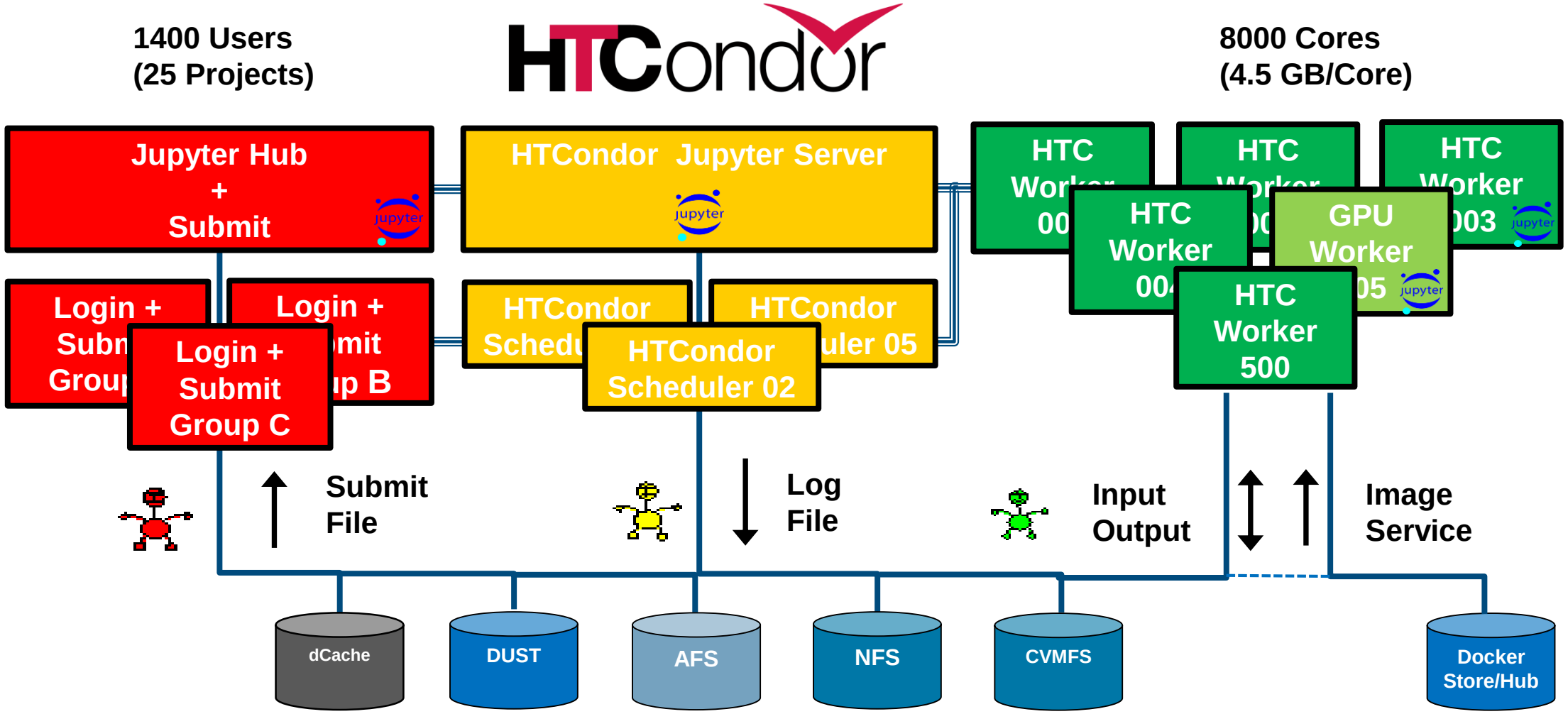
- Outlook and Conclusions

**BIRD, NAF, HTC and HPC:**
**Batch Infrastructure Resource at DESY**
**National Analysis Facility**
**High Throughput Computing**
**High Performance Computing**

Colors
Red: Interactive User
Orange: HTC Server
Green: WorkerNodes
Blue: Authentication

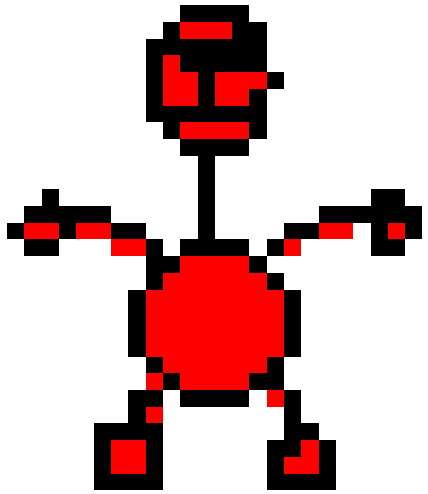
# BIRD/NAF Simple Block View

## Block Numbers

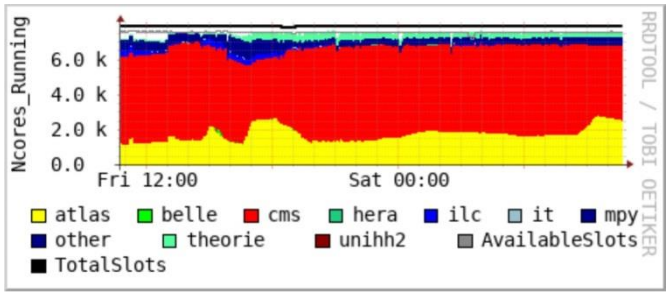


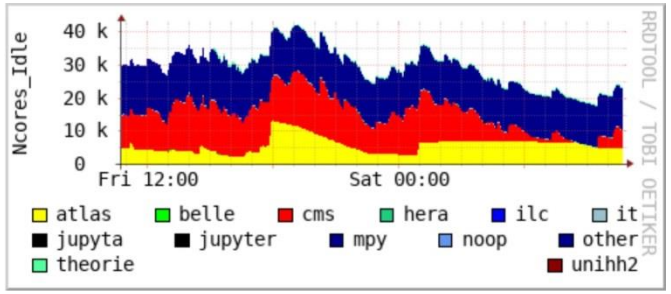
# User Perspective

**„The start time of a job should be less or in the same order as the requested runtime“**

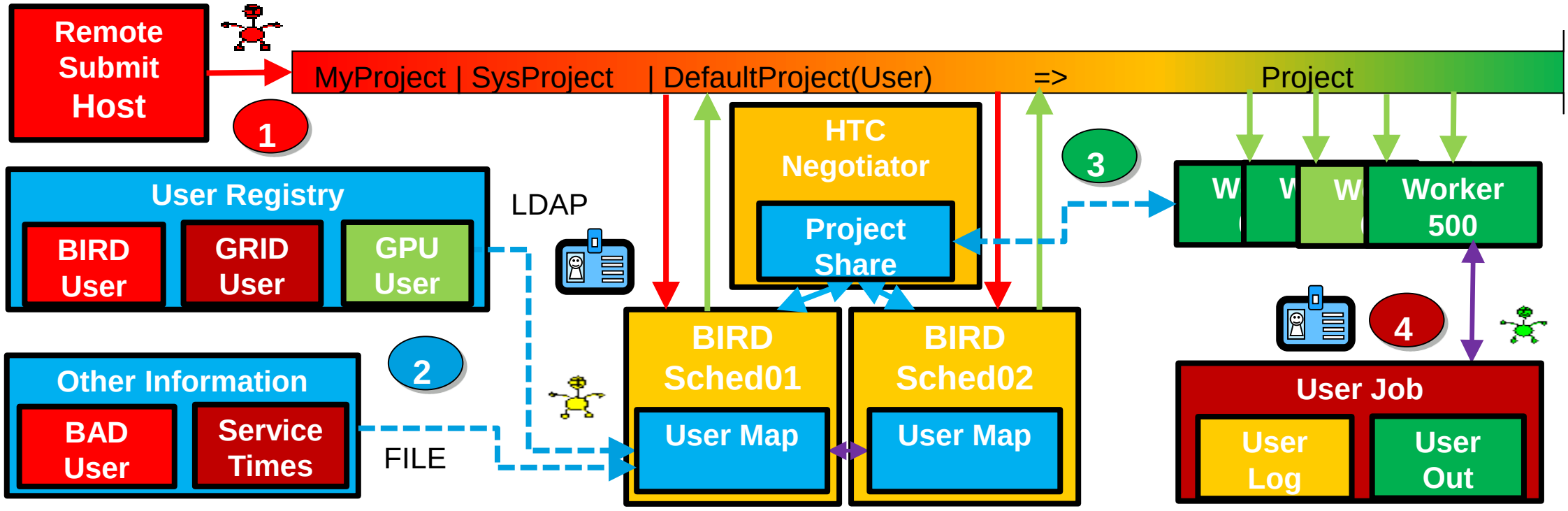


- User ClassAds
  - RequestRuntime
  - RequestMemory
  - RequestDisk
  - MyProject
- Job
  - Default Runtime 3 hours
  - Runtimes requestable from 1 Week to a few Minutes
  - Kerberos and AFS credentials
- Share
  - Proportional to the amount of ownership of the hardware
  - One week history
  - Job Classification lite and bide



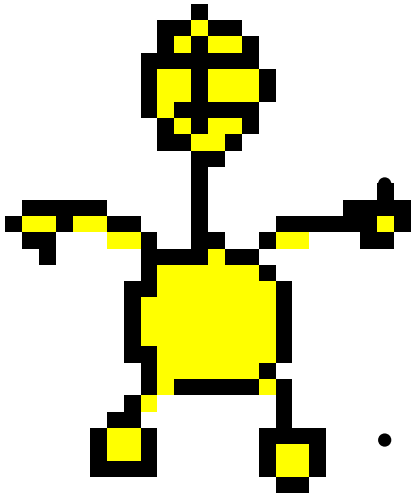
# User Registry + User Blacklisting + Project Share




| 1 | 2 | 3 | 4 |
|---|---|---|---|
| | Generate_UserMap.sh | | Job_Wrapper.sh |
| Condor_submit | cron, ldap, Transforms.htc | Quota.htc | |

# Administrator aspects

## „Most Policies are set centrally on the Schedulers“

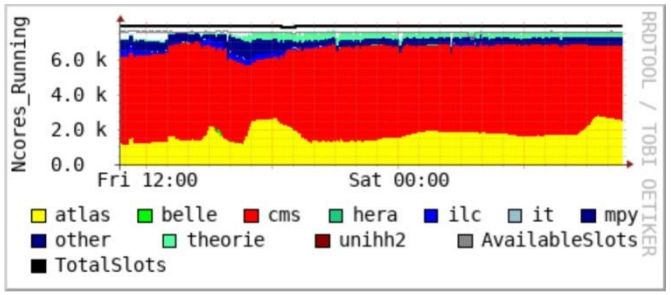


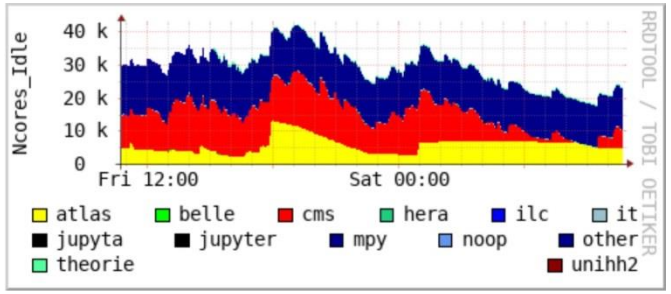
- Policies set on Scheduler
  - Maximum, requested and default Job Runtimes
  - Common Scheduler configs using Transforms
  - Project Verification

Runtime

- Default 3 hours
- Requestable from 1 Week to a few Minutes

- Implementation
  - HTC Feature Periodic Remove
  - Runtime Calculation within HTC Interface
  - Node Runtime essential in Process Management
  - Quota/Fairshare Overcommittment

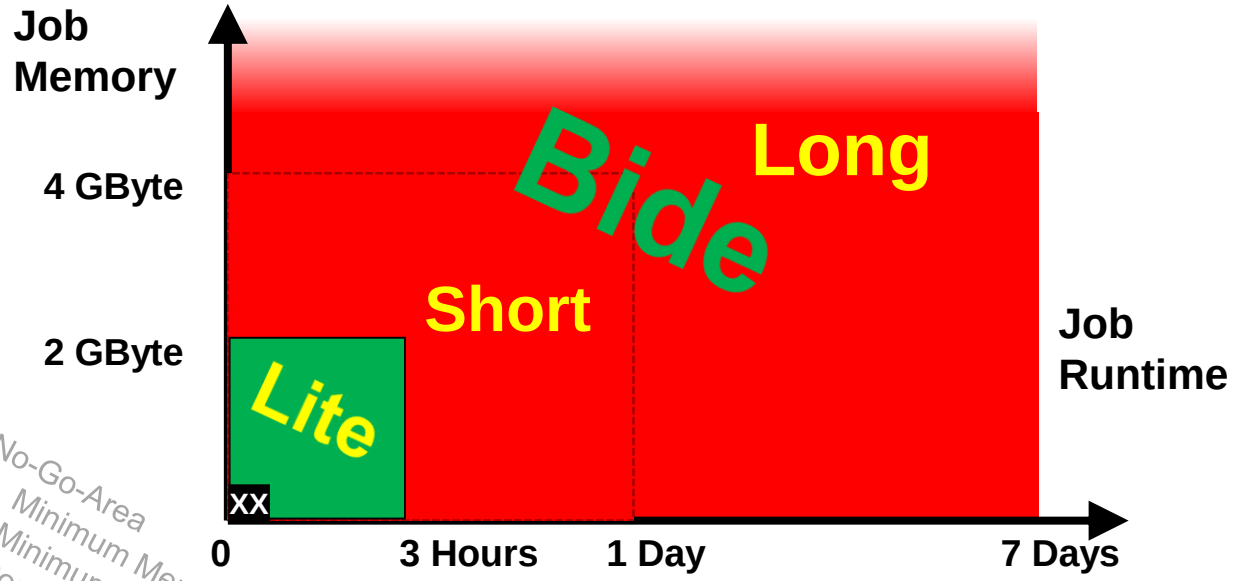


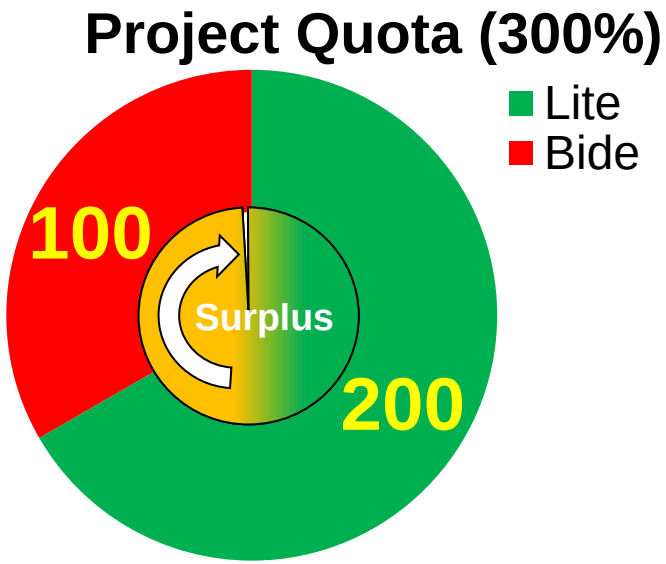
# The Base: Job Classes

## Defines metrics for Quota/Fairshare and Node Management

- Job Classes
  - Lite, Short and Long defined
  - Lite and Bide for Quotas and Shares
- Job Types
  - Single, Array, Multicore, Multiarray
  - For Informational Purpose
- Shared Quota
  - 300 % Oversubscription for Lite Jobs
  - 33 % Oversubscription for all Jobs (~ Entropy)

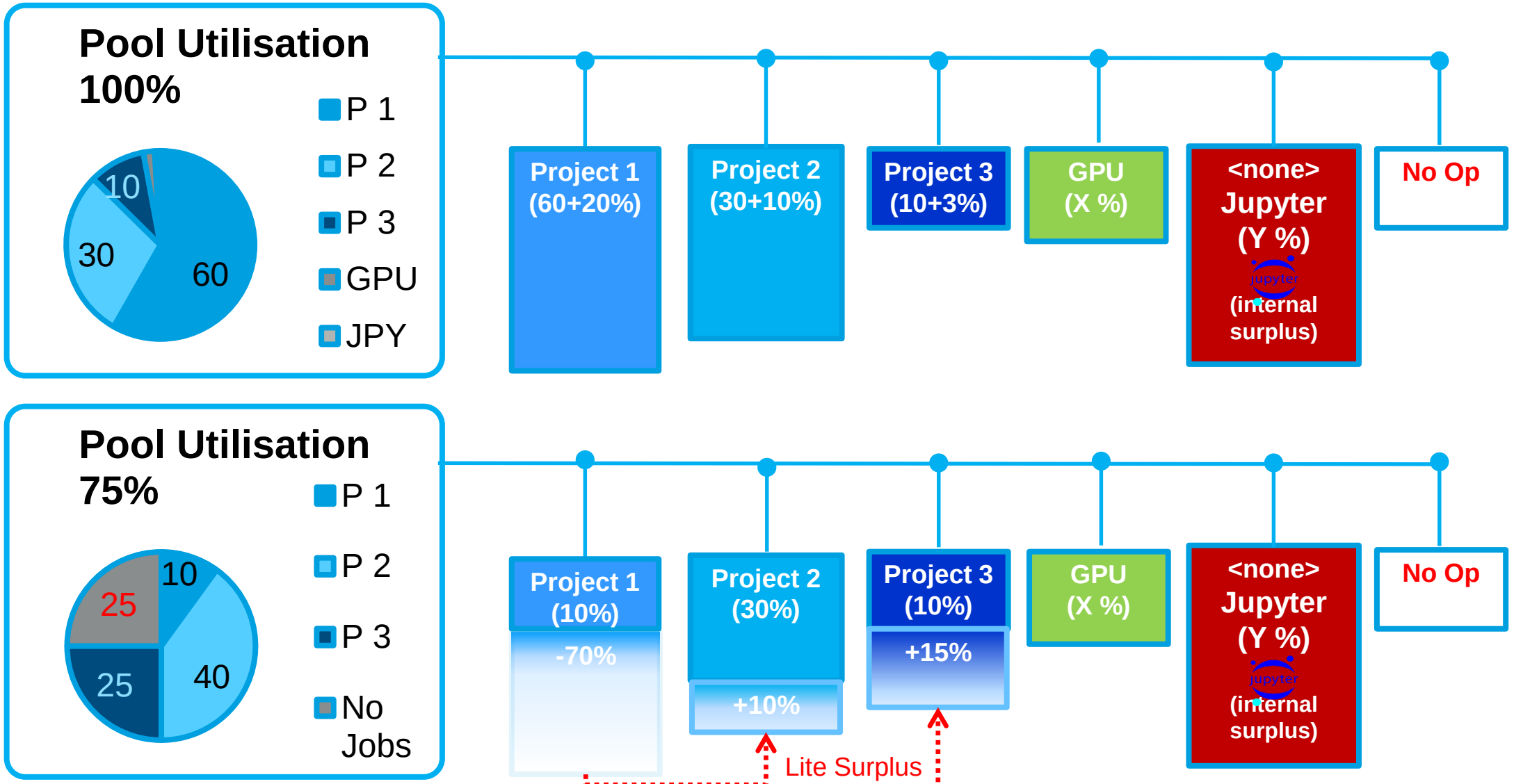



- System ClassAds
  - SysProject
  - DefProject
  - Accounting Group
  - Quota/Fairshare
- User ClassAds
  - RequestRuntime
  - RequestMemory
  - RequestDisk
  - MyProject



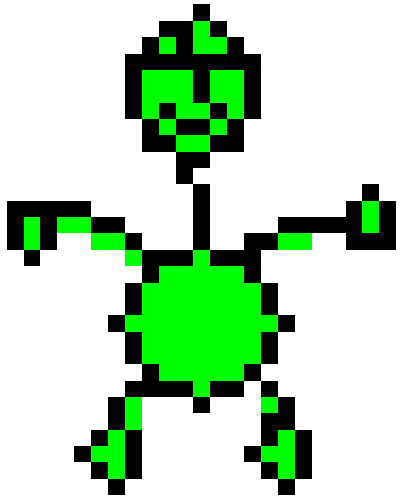
# Implementing Dynamic Fair Share (133 %)

## Two different utilisations

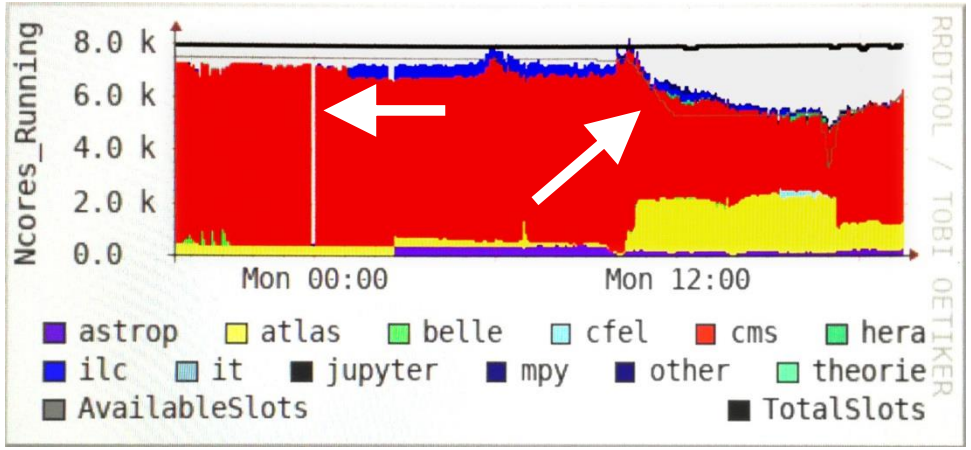


# Operator Support

## Node Management simple and without job interference

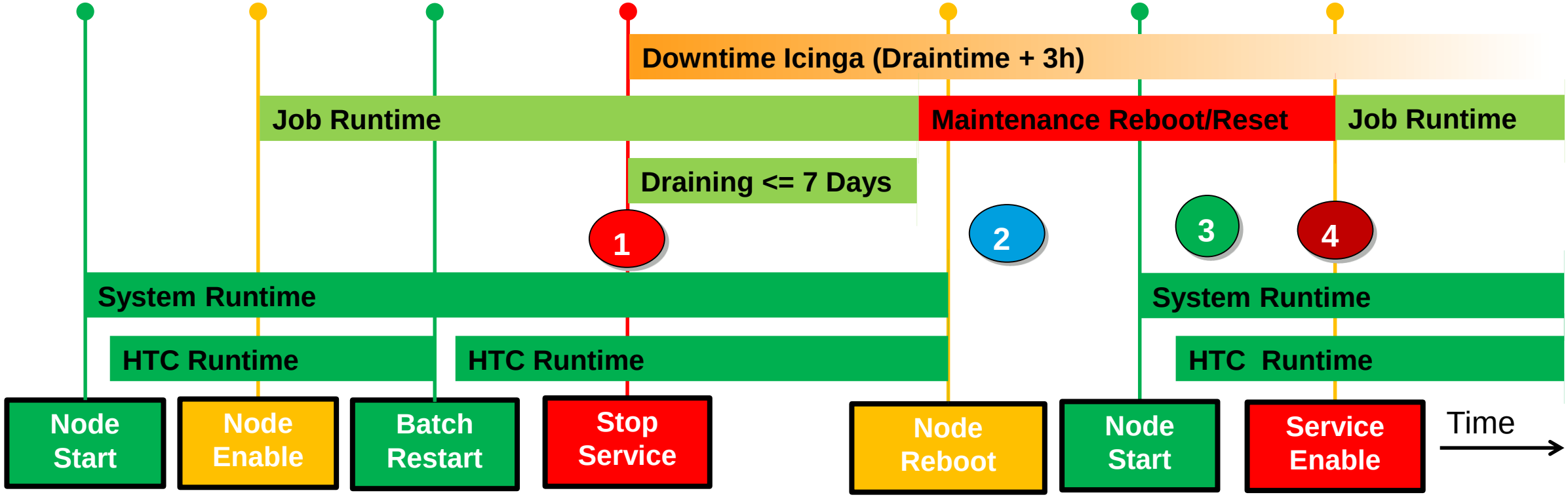


- Automated Operation of Nodes
  - For Problems (e.g. node failures)
  - For Service (e.g cluster kernel update)
  - Manually/CLI or by scripting
- Disable, drain, reboot and reset Nodes
  - No preemption or job killing
  - No specific operator knowledge needed
  - All states in one view
  - Hourly status update
  - Sets/resets exact icinga downtimes
  - Works for all pools
    - GRID, BIRD, TEST, …
- CLI
  - Batchnode.sh used on interactive nodes
  - Testversion with „Draining without Drain“



# Node and Job Timing

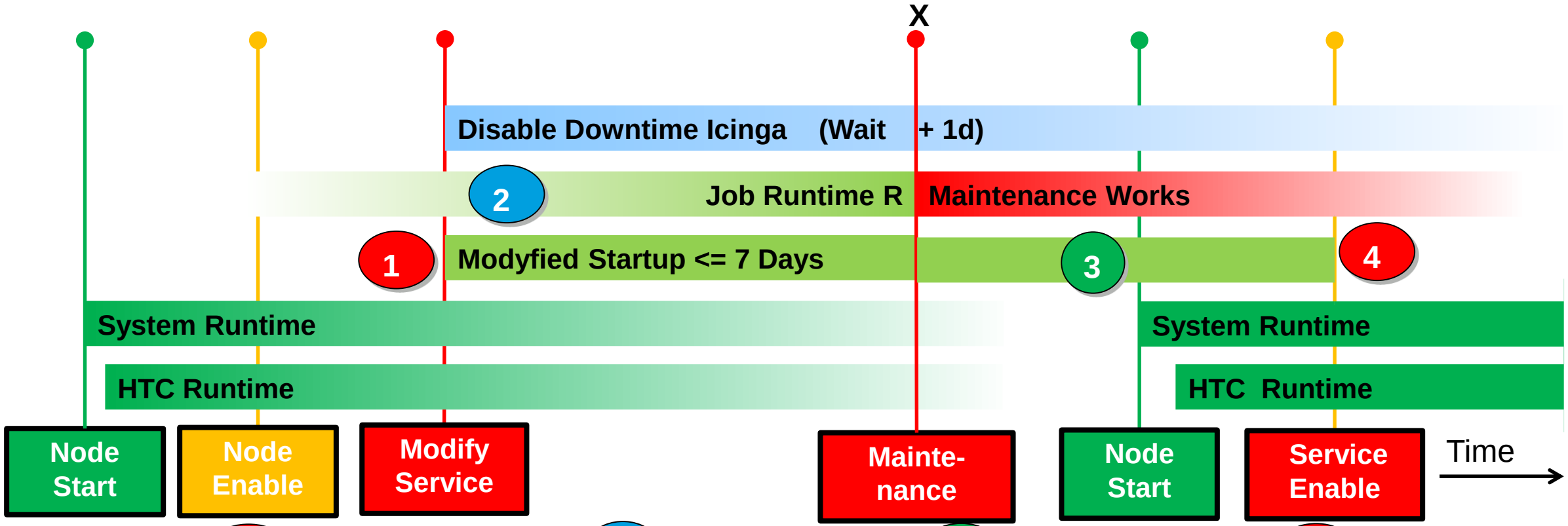
## Version A: Automatic Node Draining (Disable/)Drain/Reboot/Reset




| 1 | 2 | 3 | 4 |
|---|---|---|---|
| Batchnode.sh | Node.cron (Sys reboot) | Node.cron (node status) | Node.cron (node enable) |
| StartJobs = False<br>Condor_drain -graceful | Cron.hourly | Cron.hourly | StartJobs = True<br>Condor_drain –cancel |

# CC and Job Timing

## Version B: Draining a node w/o Drain for CC Maintenance

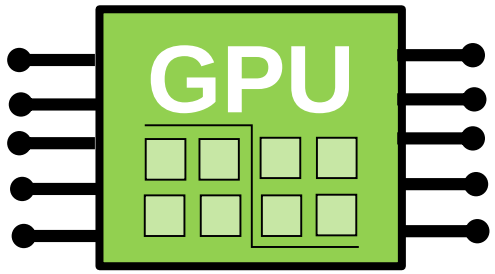


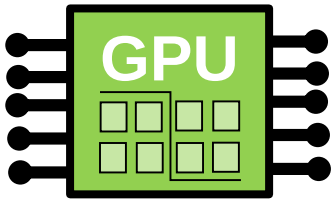

| 1 | 2 | 3 | 4 |
|---|---|---|---|
| Batchnode.sh | Jobstart if (Now + R < X) | Nodes free for Maintenance | Batchnode.sh |
| ShutdownTime = X - Min<br>InStageDrain = True | | | ShutdownTime = 0<br>InStageDrain = False |

# GPU Support

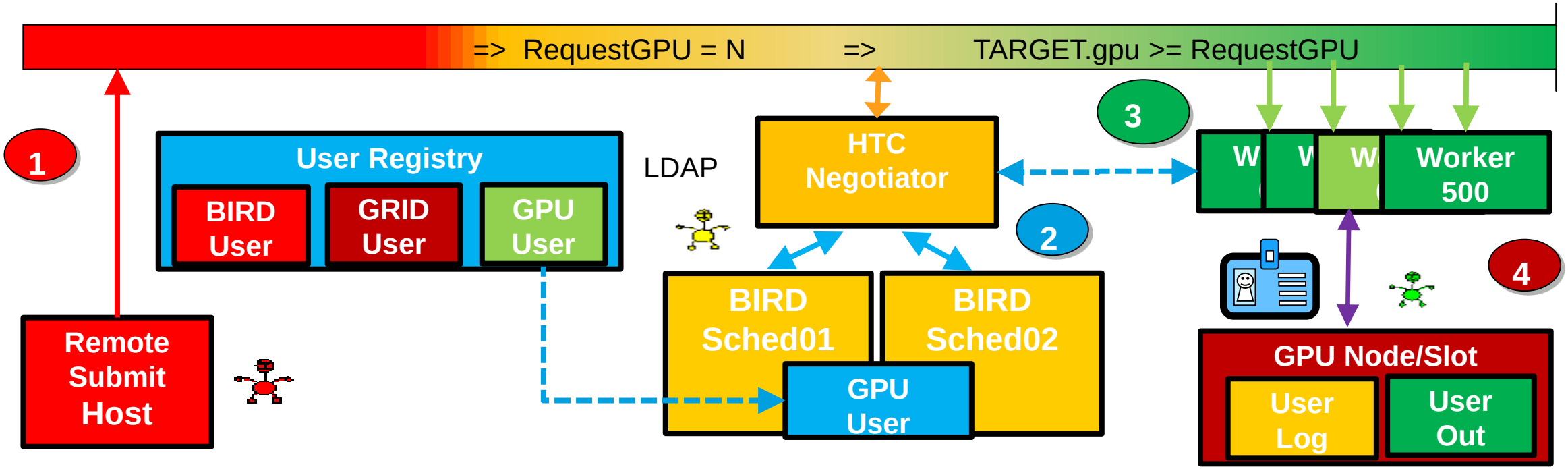
## Starting with a full node slot for a GPU

- Uses Enterprise Linux >= Version 7
- Full Node Scheduling
  - 1 GPU-Slot per GPU-Node
  - In Combination with Jupyter Slot(s)
- User needs BIRD-GPU-Resource in Registry





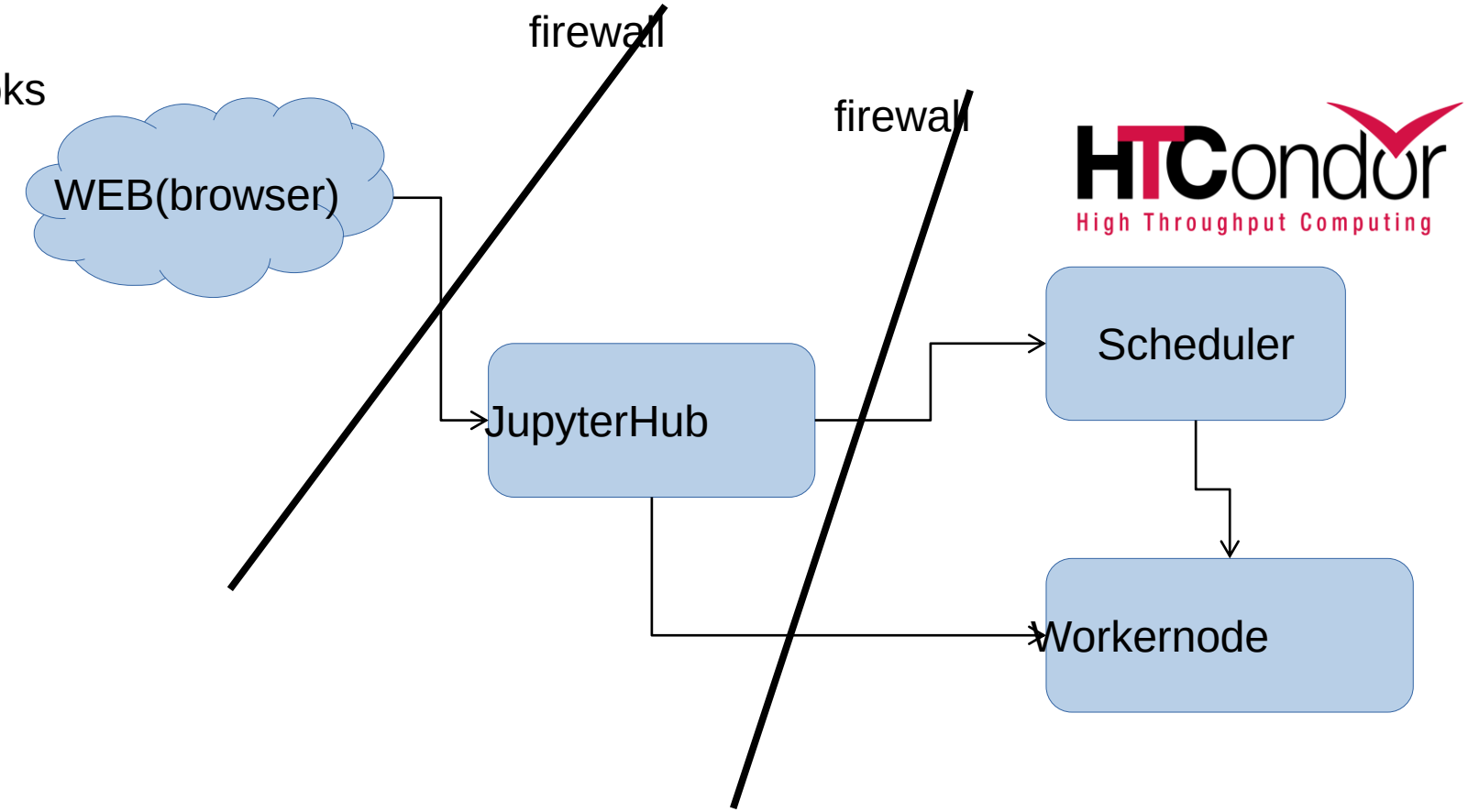

# Configuration




| 1 | 2 | 3 | 4 |
|---|---|---|---|
| | Check auth | Find GPU node | Job_Wrapper.sh |
| Condor_submit | Transforms.htc | HTC GPU Feature | Mount, configure |

# jupyter on the NAF

- Webinterface Jupyter Notebook
  - Does the Proxying to Notebooks
  - https://naf-jhub.desy.de/
  - Runs the DESY DMZ
  - Runs the Python Kernel
- Functionality Notebook Server
  - Login of Users
  - Database of logged in user
  - Access to Data, Mounts etc.
  - 2bg Ram soft limit
  - 20h runtime “start your notebook once a day”

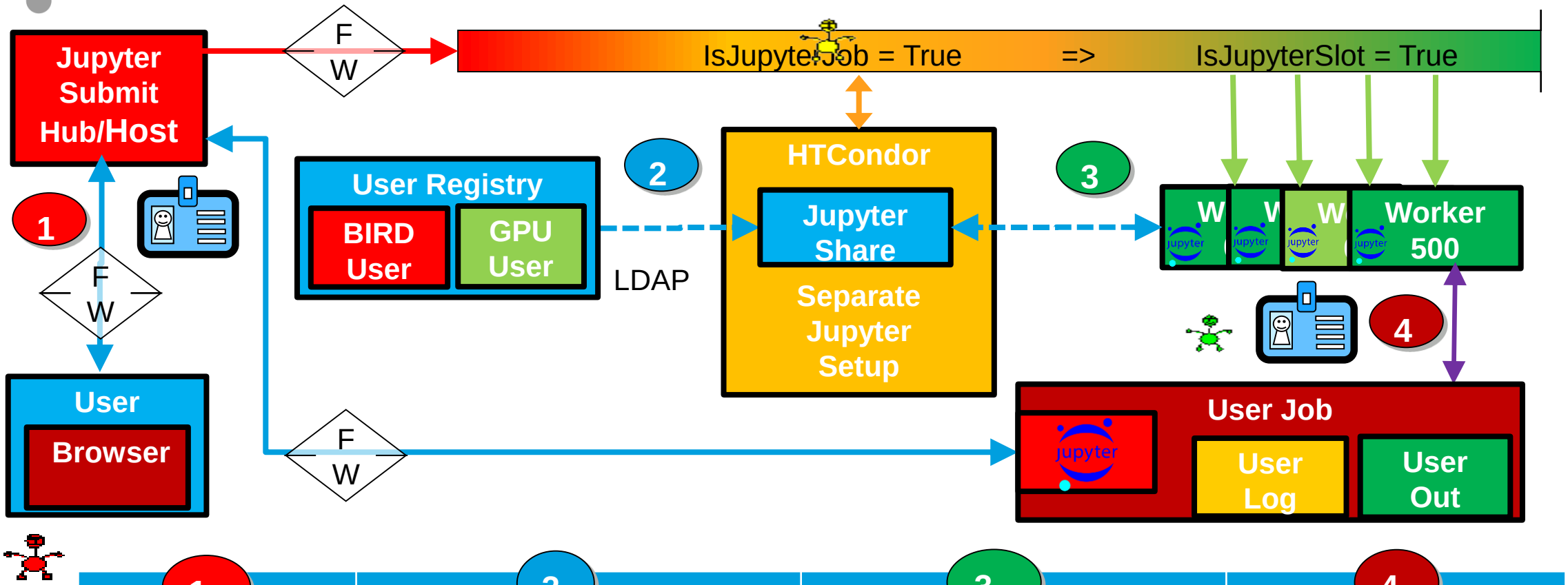


# Integration of jupyter Notebooks


- BIRD/NAF
  - Jupyter-Hub for User Access
    - User needs BIRD-Resource in Registry
      - User don’t needs BIRD-GPU-Resource
    - Jupyter Software and Configuration
    - HTC-Interface Configuration
      - Project, Runtime, …
    - External Access
    - Open Port Ranges to HTCondor Schedds and Workers

- HTCondor-Backend Configuration
  - Needs Enterprise Linux >= Version 7
  - Slots either shared on GPU nodes or oversubscripted on worker nodes
  - Scheduler Transforms for Automatic Setup
  - Special Jupyter Slots
  - Workernode Software Add On
  - Fast Startup for Interactive Usage

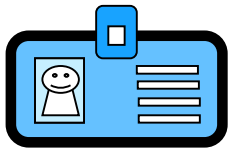
# Communication Blocks



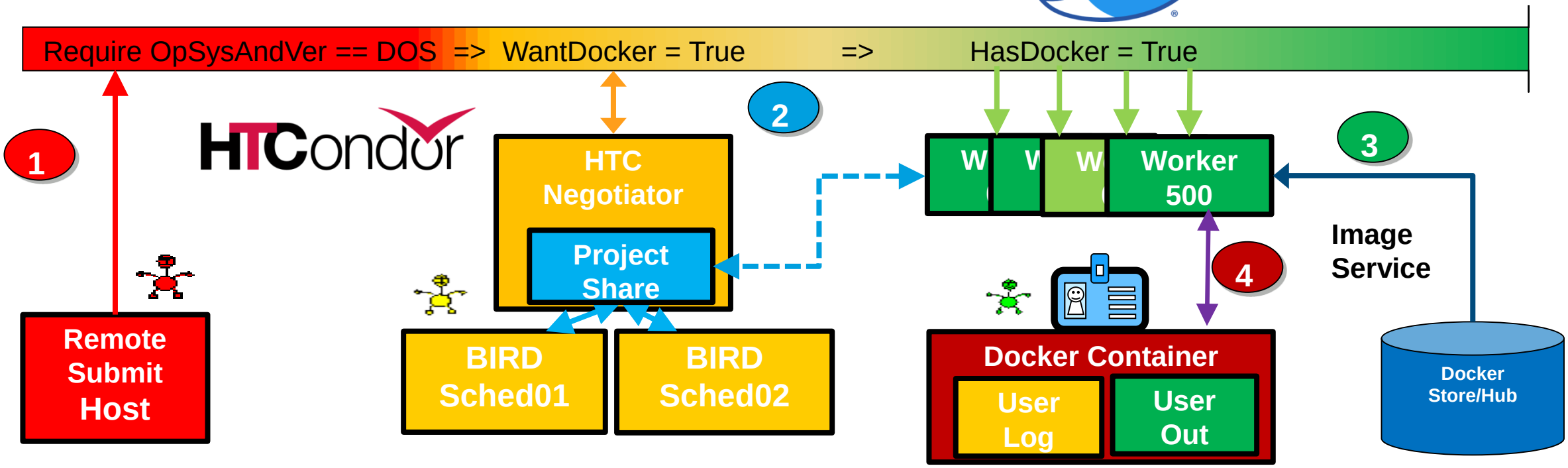

| 1 | 2 | 3 | 4 |
|---|---|---|---|
| | Prepare Jupyter Settings | Fast Quota Bypass | Job_Wrapper.sh |
| Condor_submit | ldap, Transforms.htc | Quota.htc | |

# Docker Service


- BIRD/NAF
  - „Proof of Concept“ for planned feature done
  - Docker without Docker Universe
  - External mounts
  - Project settings
  - KRB/AFS
    - Additional Key Ring
  - Image Support
    - System Images vs. User Images
    - Docker Hub with secure Images
    - Image Loading and Network Bandwidth
    - Storage Optimisation (Image Layer)



# Docker Communication Blocks




| 1 | 2 | 3 | 4 |
|---|---|---|---|
| | Prepare Docker Settings for DOS | Prepare DOS-Image | Job_Wrapper.sh |
| Condor_submit | Transforms.htc | Docker | Mount, configure |

# Function-as-a-Service

## Idea: “anonymous” job workflows with authorization tokens

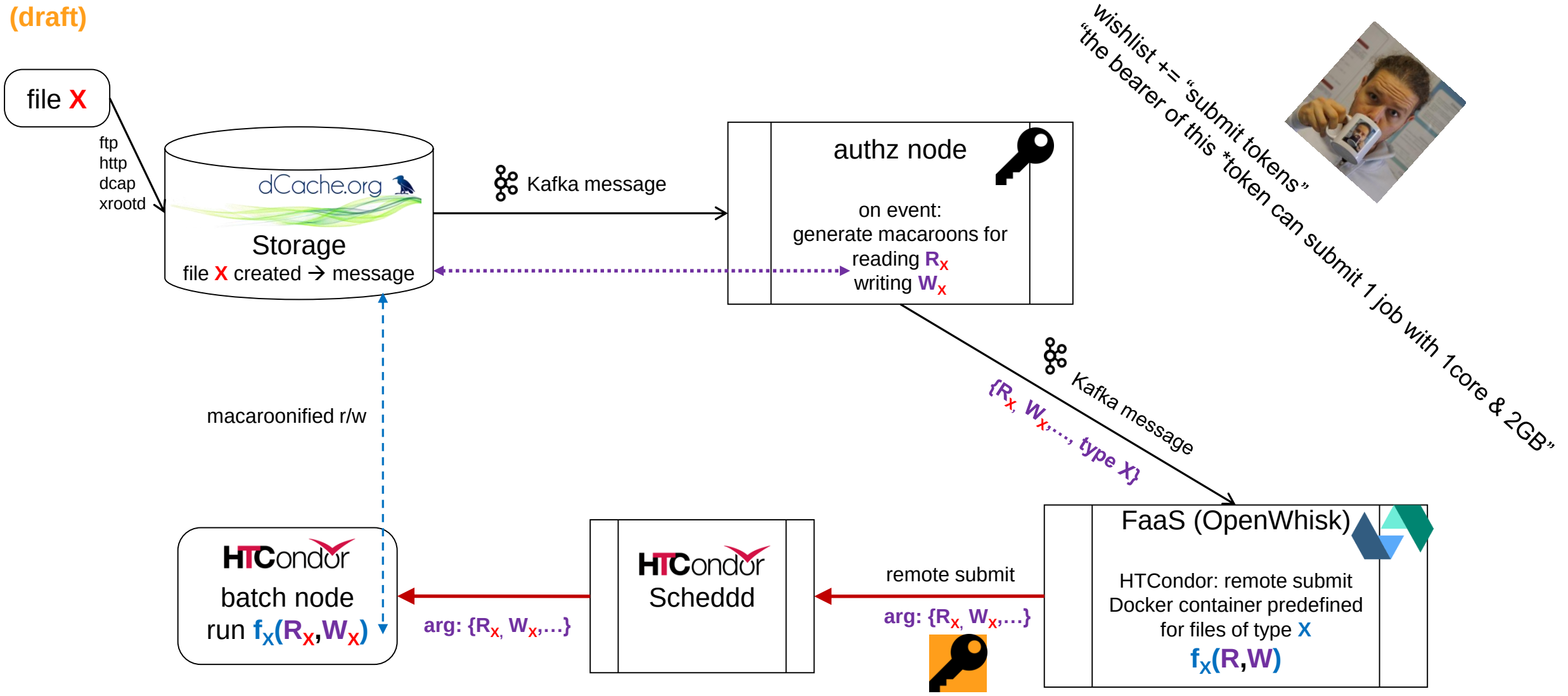
- “*personal job*”: submit  as user ‘alice’ a job running under ‘alice’ to read file X in ‘alice’ resources
- “*anonymous job*”: job is run under “nobody”, reads/writes  authorized through a transient token

- automated workflows: data events initiate jobs:
  - a new file **X** is written to the storage
  - files of type **X** are always to be processed with the same application **f**
    - let new files **X** trigger *themselves* their own processing with *function* **f(X)**

- use tokens for authorization
  - Storage Access with “*Macaroons*” as tokens
    - “the bearer of this token is allowed to read file **X** within the next 2 hours”
    - not need to carry any user authentication through the whole workflow

# Event based FaaS with Condor as Backend

(draft)




submission still needs some form of authz…
(… and have to do accounting right)

# Outlook and Conclusions

- BIRD/NAF
  - „Proof of Concept“ for planned Feature done
  - Waiting for HTCondor new Auth Features
    - Full Kerberos and AFS support as builtin
- Next Steps may be …
  - Final Jupyter and GPU Configurations
  - No SL6 and NFS
  - Docker as SL6 Replacements
    - GRID uses Singularity for legacy jobs
  - Docker for different operating system flavours
  - Function-As-A-Service
  - HTCondor Update: RequestRuntime, Idleing Jobs, Central node Status
- More Details
  - https://indico.cern.ch/event/817927/contributions/3570422/