

GO-Docker: Batch scheduling with containers

Olivier Sallou, Cyril Monjeaud

► **To cite this version:**

Olivier Sallou, Cyril Monjeaud. GO-Docker: Batch scheduling with containers. IEEE Cluster 2015, Sep 2015, CHICAGO, United States. <<http://www.mcs.anl.gov/ieeeccluster2015/>>. <10.1109/CLUSTER.2015.89>. <hal-01213323>

HAL Id: hal-01213323

<https://hal.inria.fr/hal-01213323>

Submitted on 26 Oct 2015

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



GO-Docker: Batch scheduling with containers

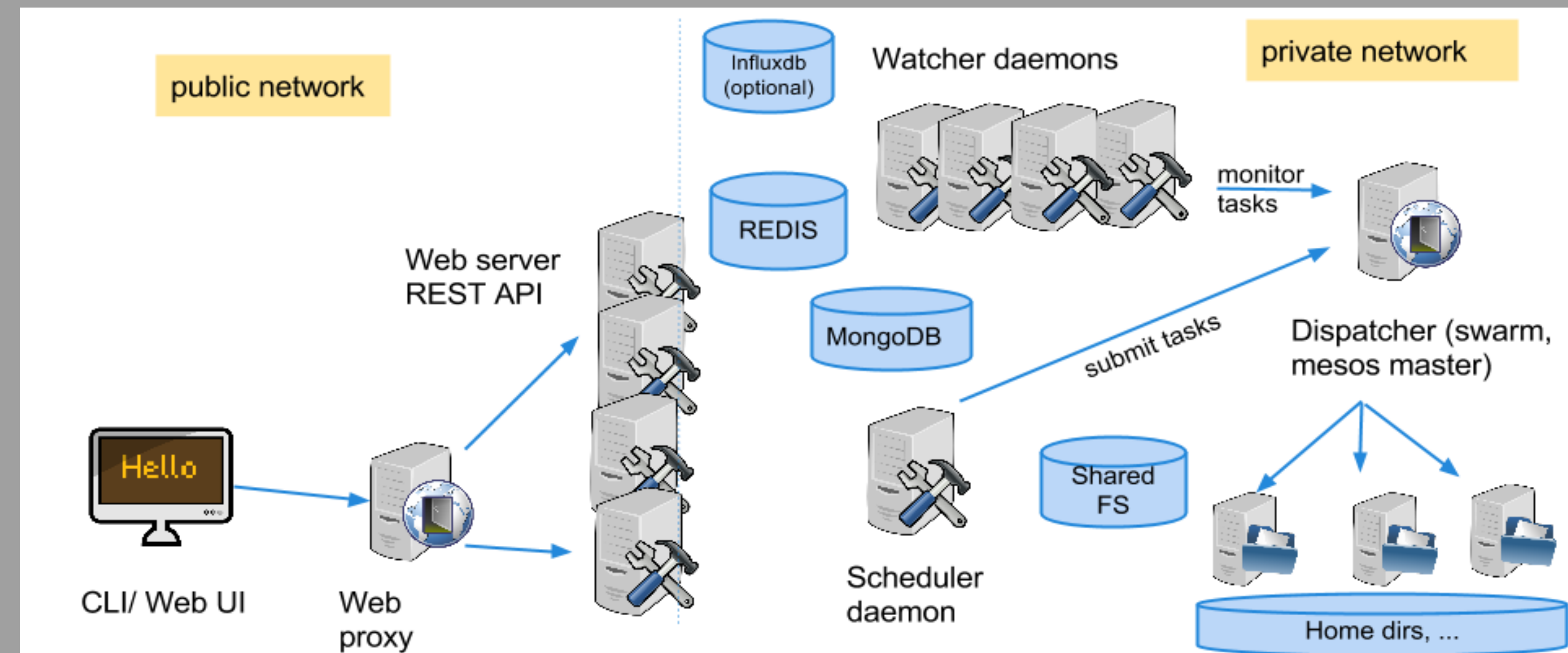
Olivier Sallou, Cyril Monjeaud

<https://bitbucket.org/osallou/go-docker/>
<http://www.genouest.org/godocker/index.html>



Jobs in containers

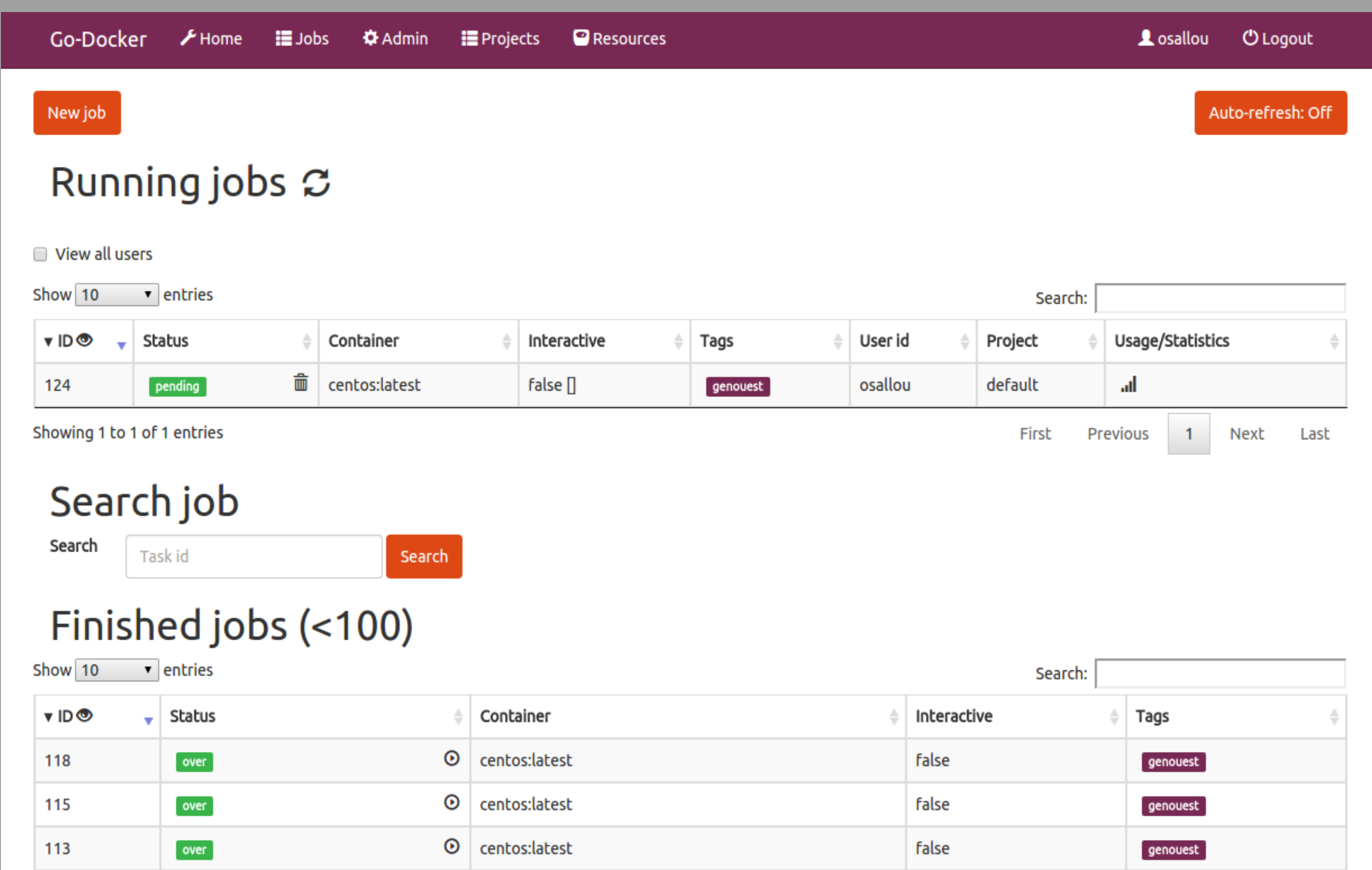
Go-Docker is a batch scheduling job manager. It executes jobs in Docker containers, providing security/isolation of resources and access to a software catalog and multiple operating systems. For multi-user systems, it mounts user directories in the container and schedules jobs according to user requirements (cpu, memory, queues). Quota and priorities are managed at user and group level. Software is in beta-test at GenOuest Bioinformatics core facility, before production, with Apache Mesos.



The Go-Docker architecture

Implementation

Go-Docker is built upon several components to orchestrate the job life cycle. Docker Swarm or Apache Mesos are in charge of the node management i.e the execution of the task on the available nodes (the scheduler daemon manages tasks priority and submission based on requirements while the watcher daemons manage tasks supervision).



Jobs can be managed via the web interface, the REST API or the CLI tool

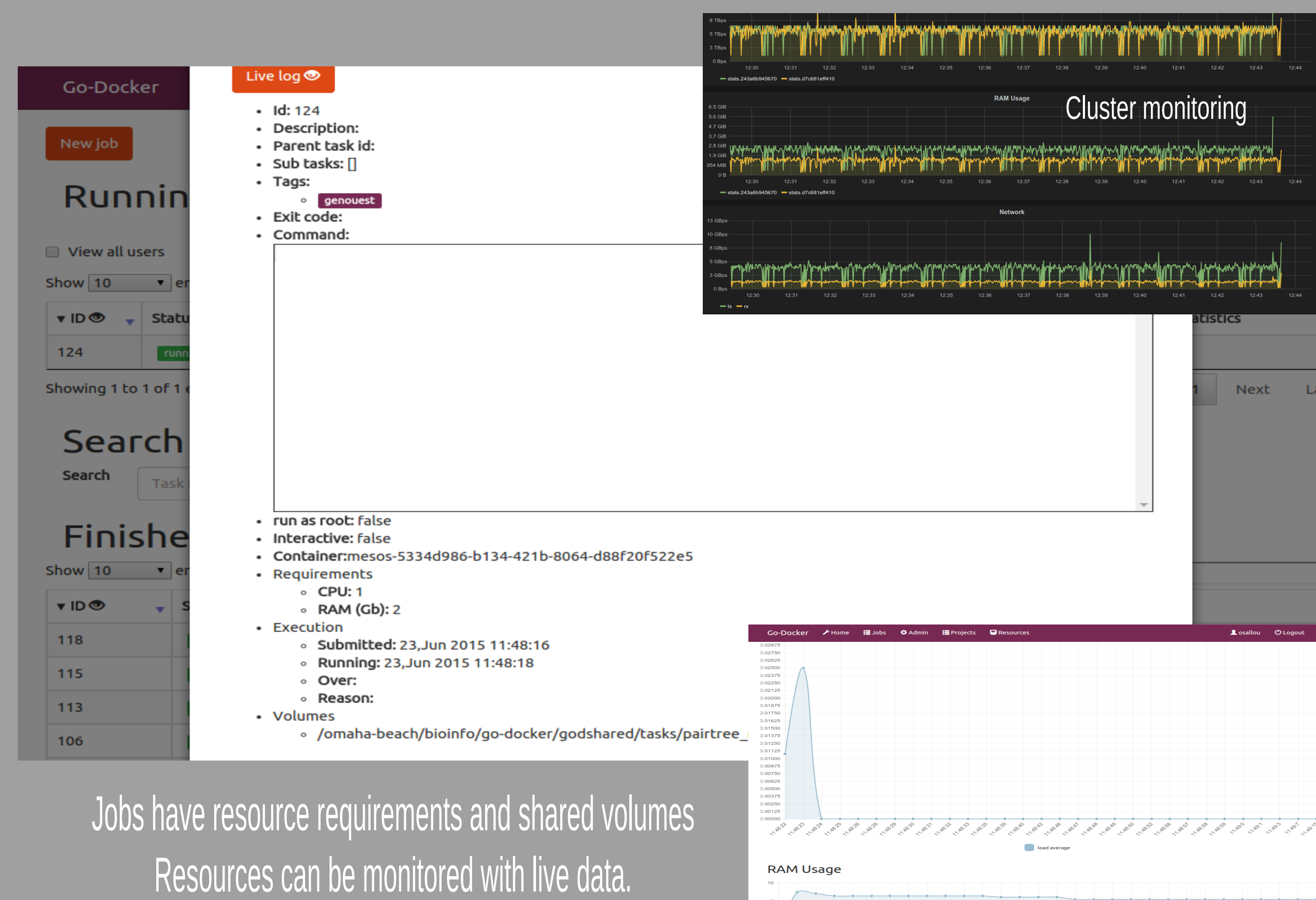
Plugins

The plugin architecture gives the possibility to add new authentication/authorization scripts, different scheduler algorithms or execution systems. It can also be extended with "watchers" to monitor job life cycle (kill after X days for example).

Scalability

Go-Docker processes and databases are horizontally scalable to scale with jobs load. The single point of failure is the scheduler, a single process to order all pending jobs according to quota and user/projects priorities.

Scalability and cloud setup will be tested soon with an AWS Research grant in Education.



Jobs have resource requirements and shared volumes
Resources can be monitored with live data.

Conclusion

- Go-Docker provides:
- secured job/interactive session control with isolation
 - jobs live monitoring and past statistics
 - optional root access to the container
 - immediate access to different operating systems and a growing catalog of software (Docker registry)
 - easy container usage with no super-user rights
 - Web UI, CLI, REST interface