GoDocker

A batch scheduling system with Docker containers

Web - http://www.genouest.org/godocker/
Code - https://bitbucket.org/osallou/go-docker
Twitter - #godocker

Olivier Sallou – IRISA - 2016

What

- Execute batch jobs/commands in containers
- For multi-user system (ldap based for example)
- With personal and/or common shared directories (home, central data, …)
- In a scalable architecture to handle massive job submission.

Why?

- Need for an open source scheduling job submission tool (like Sun Grid Engine)
  - with isolation of resources
  - availability of tools without cluster specific OS/version issues (with containers)
  - with remote and authenticated access
  - with access to job resource monitoring

How?

- Using proven technologies and software
- Using scalable components
- With plugin support to modify easily default behavior and adapt it to YOUR system.

Technologies

- Docker: for containers
- Docker Swarm, Apache Mesos, Kubernetes for job execution and dispatch, as well as for node monitoring.
- Google cAdvisor: for job monitoring
- Language: Python
- Databases backend: MongoDB, Redis, InfluxDB (optional).
- Monitoring: Prometheus

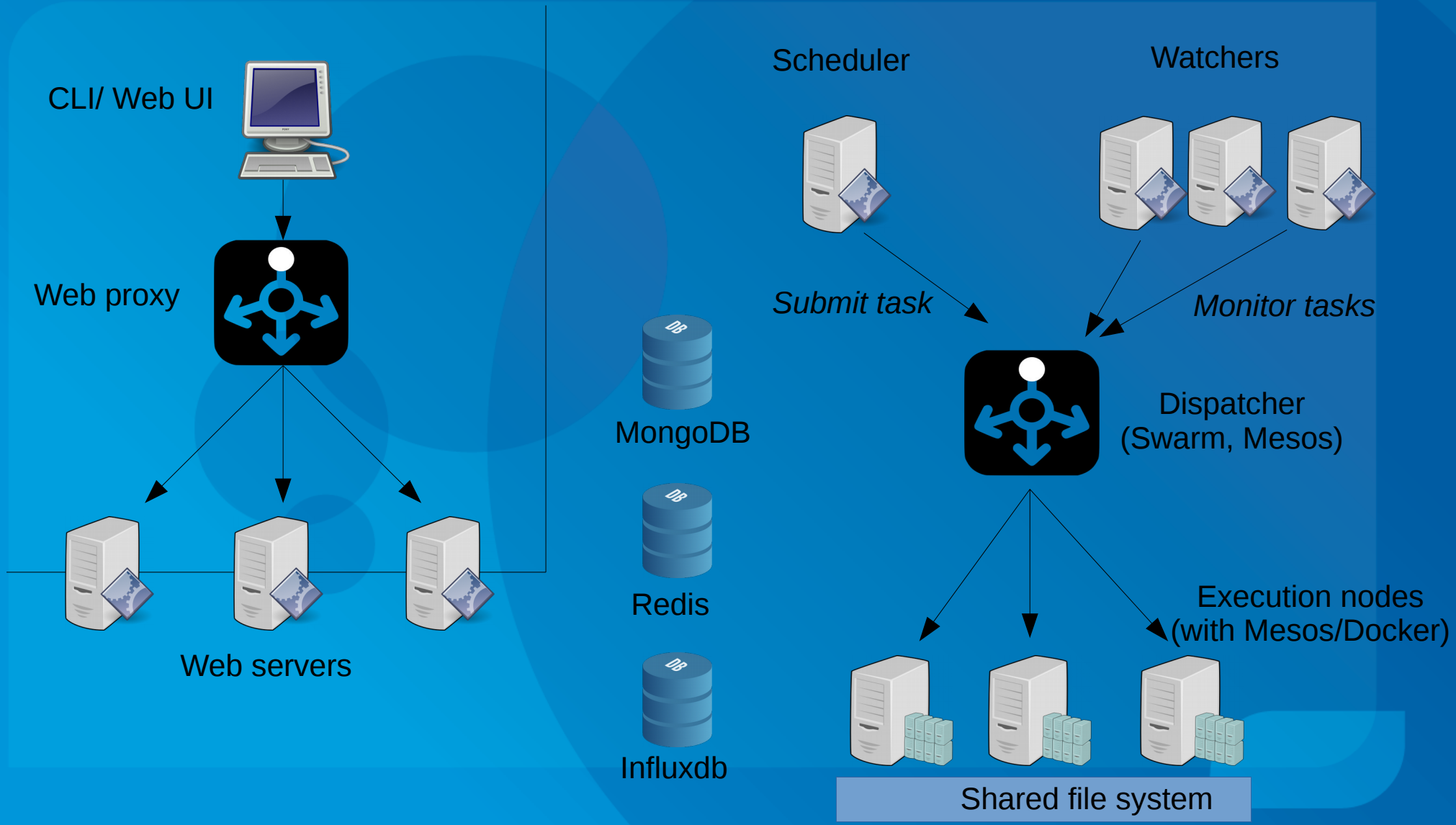
Features

- Remote execution of a job (command line)
  - in a Docker container
  - with requested resources (cpu, memory)
  - with requested directories mounted in container (according to ACL)
- Allowed container images can be limited to a list (config)
- User can specify the container image (config)
- Optional root access to container (config)

Features

- Interactive sessions (ssh) in a container
- User/Group priority and quotas.
- Jobs scheduling according to multiple properties (priority, waiting time, previous usage, …). Fair share algorithm available.
- Plugins to modify or add features.
- Global and per job monitoring (past and live).
- Partial DRMAA v1 support

Architecture



Databases

- Mongodb:
  - used to store jobs data
- Redis:
  - use lists to dispatch requests between executors to monitor jobs
- Influxdb:
  - optional db to store time based data (cpu/memory usage, number of jobs, etc.)

Components

- CLI : Command Line Interface
- Web interface / REST API
- Authentication / ACLs => plugins
- Scheduler => plugins
- Executor => plugins
- Watchers => plugins

Command Line Interface

- Execute commands using the REST API of the web server:
  - submit and control running jobs
  - download output files from jobs
  - etc.
- Some commands are dedicated to administrators:
  - project and user quota manager
  - etc.

Web server

- Submit and manage tasks via web UI
- REST interface for remote control
- Partial DRMAA v1 integration

- Register new tasks for scheduler.

Authentication / ACL

- A plugin is available to authenticate users with an LDAP but it may be adapted to your needs
  - manage authentication for web site
  - define which volumes/directories can be mounted in container (user home directory etc.), and their mode (ro, rw).
- Other plugins can be developed for specific authentication/acl

Scheduler

- Only one instance of the process
- The scheduler reorder job requests:
  - per priority (user and/or group)
  - reject if quota reached
  - different algorithms are available:
    - fifo
    - fair(share)
    - others can be added with plugins

Scheduler

- It executes the job command using the executor plugin:
  - Docker Swarm
  - Apache Mesos
  - others can be developed
  - manage port mapping for interactive jobs

Executor

- Multiple instances can be run to scale with the number of jobs to monitor.

- Manage kill or reschedule requests
- Checks the status of the job (running, over)
- Trigger watchers (see next slide)

- When job is over, it updates job status.

Watcher

- Watchers are plugins called by executors during job execution to act upon job life cycle:
  - ex: kill job
  - ex: update some meta-data
- New plugins can be added
- Available:
  - Maxlifespanwatcher: kill a job after X days.

Monitoring

- Cadvisor
  - helps to monitor “live” job cpu/memory usage.
  - data can be saved in InfluxDB for later analysis.
- Previous jobs data are kept in MongoDB for statistics/analysis.
- Prometheus metrics endpoint for global statistics
- Prometheus can connect to cAdvisor for container metrics
- Python logging: can export to Graylog etc.

Supervision

- Can register to etcd or consul to monitor processes
- With consul, possibility to use service discovery with his DNS for automatic load balancing to web servers.

Networking

- Support CNI (container network interface) for IP per container to avoid port mapping
- Via Docker plugins
- Via Mesos Unified Containerizer + CNI plugins

Deployment

- On premise
- Vagrant box for testing
- Docker images
- Chef and Amazon recipes

Job execution

- Script is executed in a container with his uid/gid
- User can ask for home directory and/or shared directories (read only if root)
- User can write in a per-job directory accessible from a gateway or HTTP
- Stdout/stderr are saved in job directory
- Live log of stdout
- Live resource usage monitoring of job
- When job is archived (config or manual), per-job directory is deleted

About

- Authors:
  - Olivier Sallou (IRISA / Univ. Rennes 1)
  - Cyril Monjeaud (IRISA/ Univ. Rennes 1)

- License: Apache-2.0

Web: http://www.genouest.org/godocker/

Code: https://bitbucket.org/osallou/go-docker

Documentation: https://godocker.atlassian.net/wiki/display/GOD/GODOCKER