Workflow automation using Docker Swarm and GitLab CI

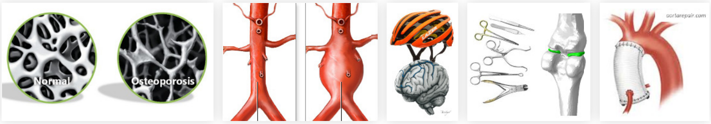
@ Flanders Institute for Biomechanical Experimentation

Johan Philips - NL-RSE - November 20th, 2019

Find slides at https://u0052546.pages.mech.kuleuven.be/presentations/rse/ (non-IE browser)

A bit of context…

What is FIBEr?



Mechanical properties characterization of biological tissues and biomaterials

FIBEr team

F I B E R
Leuven MTC Medical Technology Centre
HERCULES STICHTING
fwo Research Foundation Flanders Opening new horizons
KU LEUVEN

FIBEr statistics (May 2019)

54 researchers gained access to FIBEr Cloud Services

1565 labels printed with FIBEr Labeler

1233 samples registered in FIBEr Database

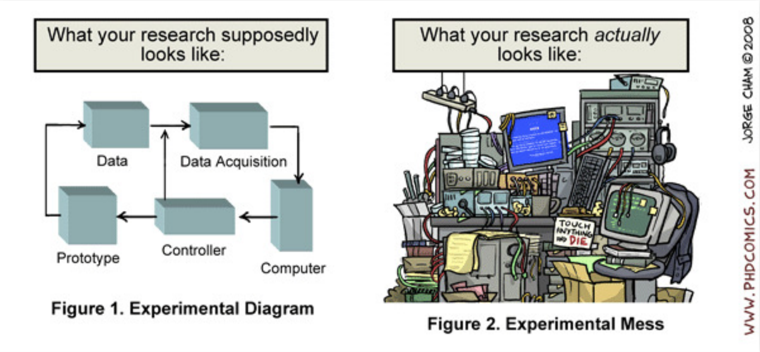
202 experiments registered via FIBEr Dashboard or FIBEr Uploader

368 datasets packed and shipped to Data Center

328.27 GB safely stored at ICTS data center

21.4 TB temporary kept on FIBEr Buffer

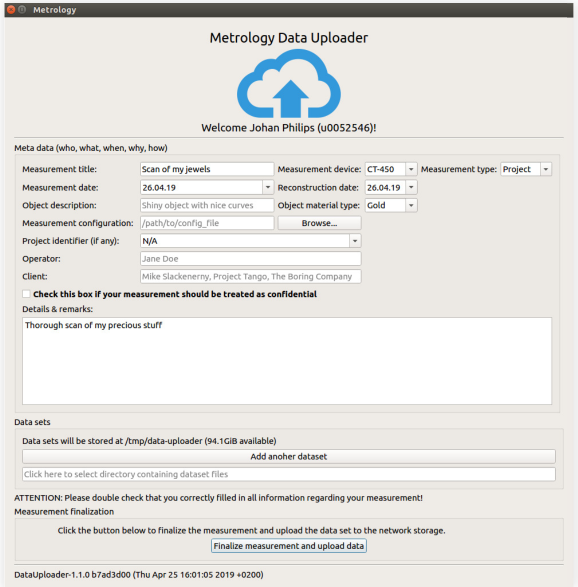
The need for workflow automation


What your research supposedly looks like:
Data
Data Acquisition
Prototype
Controller
Computer
Figure 1. Experimental Diagram
What your research *actually* looks like:
TOUCH ANYTHING AND DIE
Figure 2. Experimental Mess
JORGE CHAM ©2008
WWW.PHDCOMICS.COM

Traceability - Activities and manipulation with samples are logged.
Safe Storage - Experimental data is automatically uploaded.
Error Resilience - Automated data collection and validation reduces human error.
Ease of Use - Intuitive guided workflows help researchers during experimentation.

What do we offer?

FIBEr frontends for everyday lab workflows



Metrology Data Uploader

Welcome Johan Philips (u0052546)!

Meta data (who, what, when, why, how)

Measurement title: Scan of my jewels
Measurement device: CT-450
Measurement type: Project
Measurement date: 26.04.19
Reconstruction date: 26.04.19
Object description: Shiny object with nice curves
Object material type: Gold
Measurement configuration: /path/to/config_file
Project identifier (if any): N/A
Operator: Jane Doe
Client: Mike Slackenerny, Project Tango, The Boring Company

□ **Check this box if your measurement should be treated as confidential**

Details & remarks:
Thorough scan of my precious stuff

Data sets

Data sets will be stored at /tmp/data-uploader (94.1GiB available)

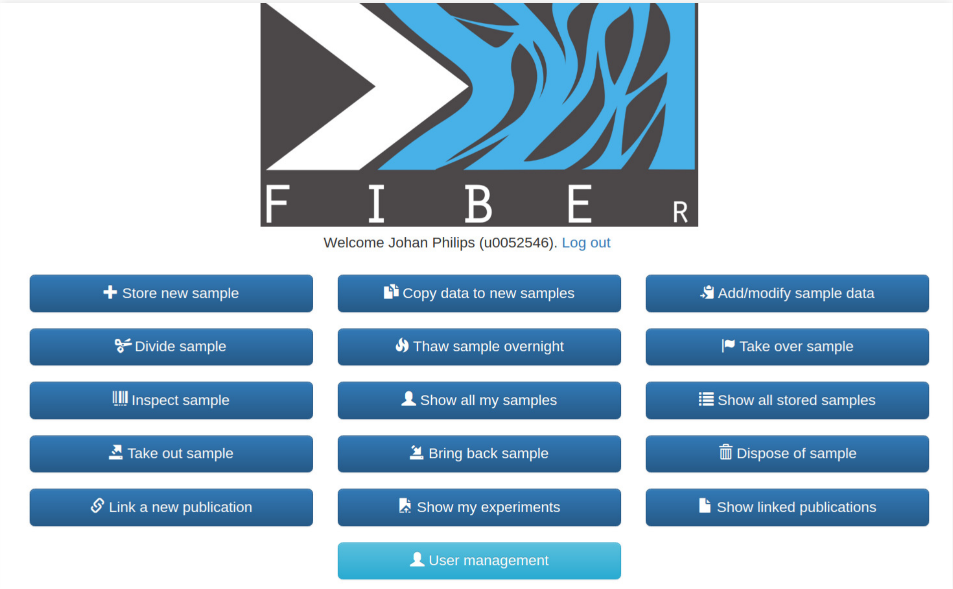
Click here to select directory containing dataset files

ATTENTION: Please double check that you correctly filled in all information regarding your measurement!

Measurement finalization

Click the button below to finalize the measurement and upload the data set to the network storage.

DataUploader-1.1.0 b7ad3d00 (Thu Apr 25 16:01:05 2019 +0200)

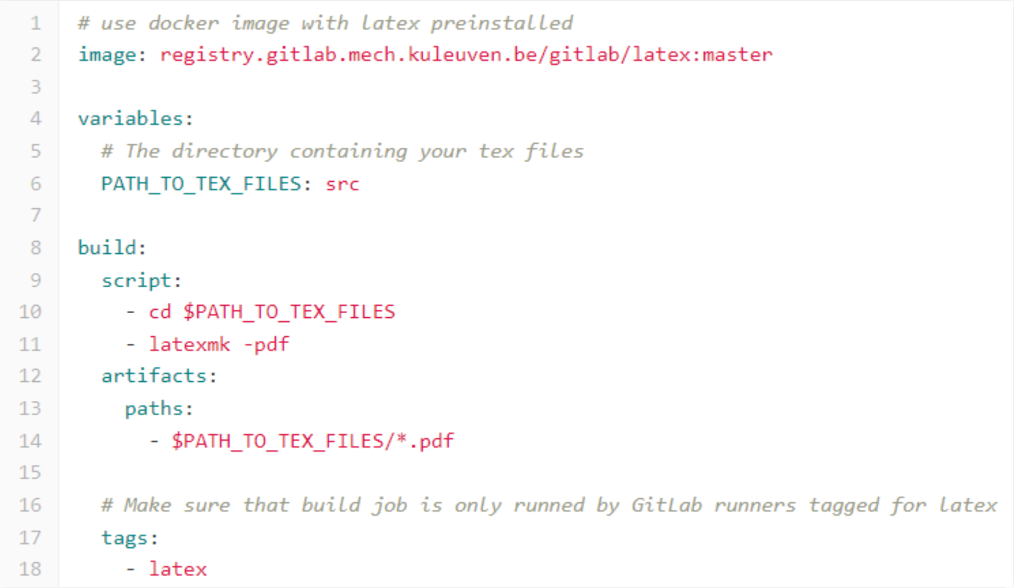


Welcome Johan Philips (u0052546). Log out

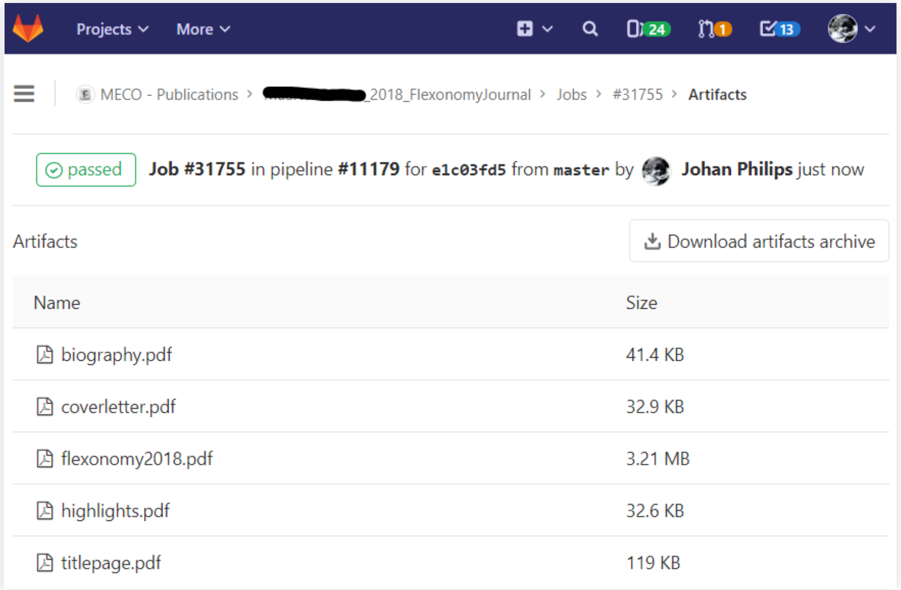
Software development to support FIBEr researchers
Reused in already five other labs!

Automation support for other research workflows

GitLab CI for version control and auto-build of LaTeX publications



```
# use docker image with latex preinstalled
image: registry.gitlab.mech.kuleuven.be/gitlab/latex:master

variables:
  # The directory containing your tex files
  PATH_TO_TEX_FILES: src

build:
  script:
    - cd $PATH_TO_TEX_FILES
    - latexmk -pdf
  artifacts:
    paths:
      - $PATH_TO_TEX_FILES/*.pdf

  # Make sure that build job is only runned by GitLab runners tagged for latex
  tags:
    - latex
```



MECO - Publications › [redacted]2018_FlexonomyJournal › Jobs › #31755 › **Artifacts**

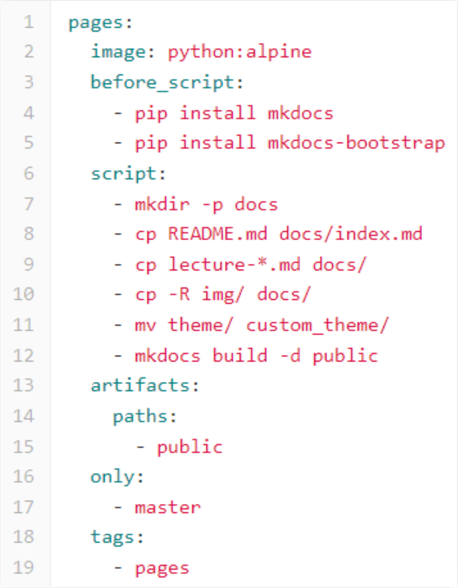
passed **Job #31755** in pipeline **#11179** for `e1c03fd5` from `master` by **Johan Philips** just now

Artifacts

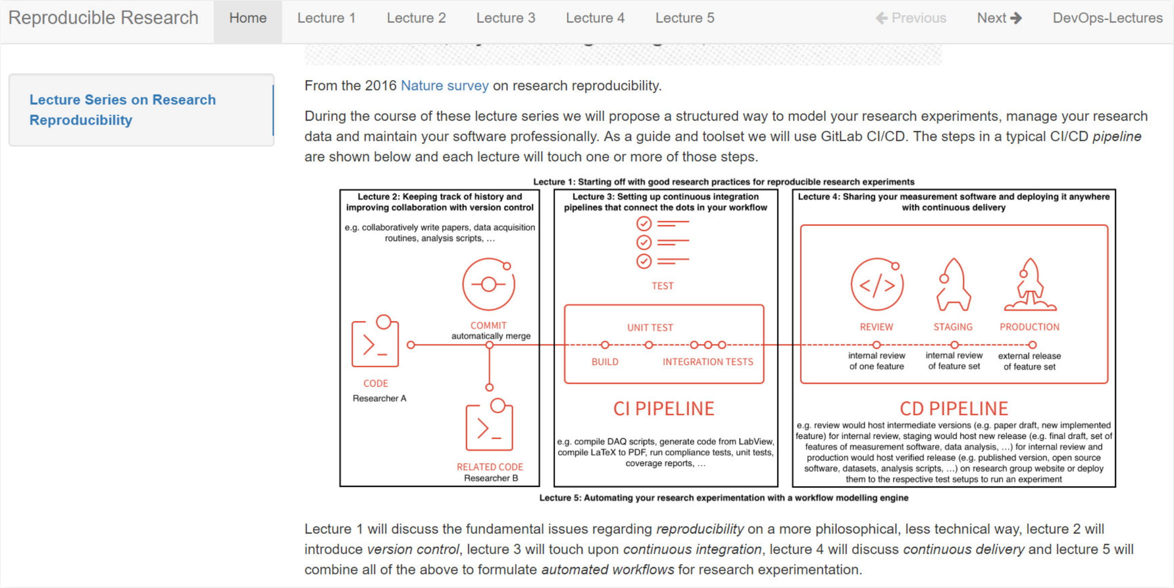
Name	Size
biography.pdf	41.4 KB
coverletter.pdf	32.9 KB
flexonomy2018.pdf	3.21 MB
highlights.pdf	32.6 KB
titlepage.pdf	119 KB

Automation support for other research workflows (2)

GitLab Pages for automated web pages for lectures, research, staff info



```
pages:
  image: python:alpine
  before_script:
    - pip install mkdocs
    - pip install mkdocs-bootstrap
  script:
    - mkdir -p docs
    - cp README.md docs/index.md
    - cp lecture-*.md docs/
    - cp -R img/ docs/
    - mv theme/ custom_theme/
    - mkdocs build -d public
  artifacts:
    paths:
      - public
  only:
    - master
  tags:
    - pages
```



From the 2016 Nature survey on research reproducibility.

During the course of these lecture series we will propose a structured way to model your research experiments, manage your research data and maintain your software professionally. As a guide and toolset we will use GitLab CI/CD. The steps in a typical CI/CD *pipeline* are shown below and each lecture will touch one or more of those steps.

Lecture 1: Starting off with good research practices for reproducible research experiments

Lecture 2: Keeping track of history and improving collaboration with version control

e.g. collaboratively write papers, data acquisition routines, analysis scripts, ...

Lecture 3: Setting up continuous integration pipelines that connect the dots in your workflow

e.g. compile DAQ scripts, generate code from LabView, compile LaTeX to PDF, run compliance tests, unit tests, coverage reports, ...

Lecture 4: Sharing your measurement software and deploying it anywhere with continuous delivery

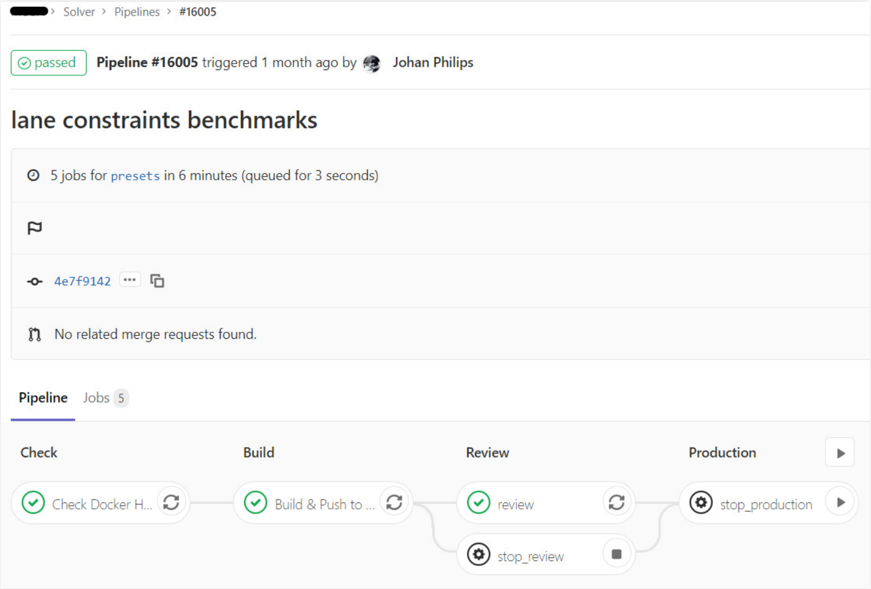
e.g. review would host intermediate versions (e.g. paper draft, new implemented feature) for internal review, staging would host new release (e.g. final draft, set of features of measurement software, data analysis, ...) for internal review and production would host verified release (e.g. published version, open source software, datasets, analysis scripts, ...) on research group website or deploy them to the respective test setups to run an experiment

Lecture 5: Automating your research experimentation with a workflow modelling engine

Lecture 1 will discuss the fundamental issues regarding *reproducibility* on a more philosophical, less technical way, lecture 2 will introduce *version control*, lecture 3 will touch upon *continuous integration*, lecture 4 will discuss *continuous delivery* and lecture 5 will combine all of the above to formulate *automated workflows* for research experimentation.

Automation support for other research workflows (3)

Custom GitLab CI pipelines to improve reproducibility



[redacted] › Solver › Pipelines › #16005

passed **Pipeline #16005** triggered 1 month ago by **Johan Philips**

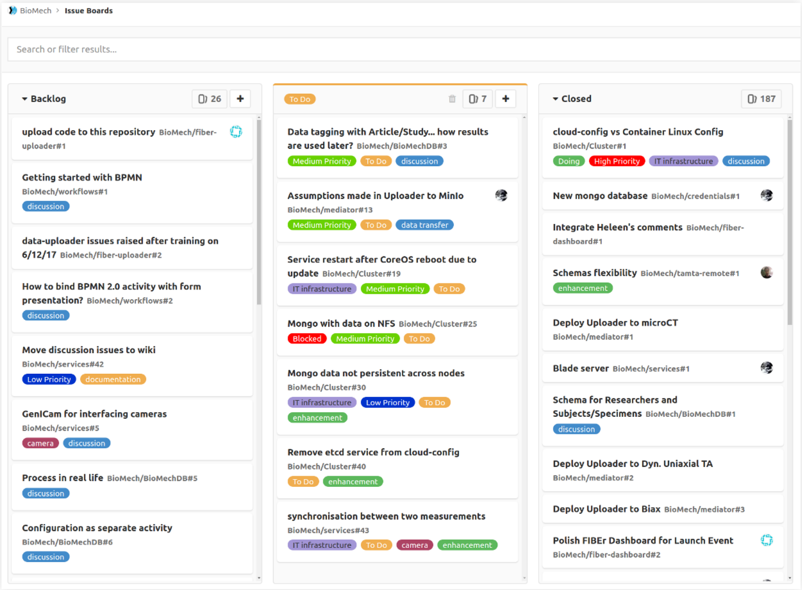
lane constraints benchmarks

5 jobs for `presets` in 6 minutes (queued for 3 seconds)

`4e7f9142`

No related merge requests found.

GitLab Issue board for ’support tickets’ and software project management



Backlog 26

- **upload code to this repository** BioMech/fiber-uploader#1
- **Getting started with BPMN** BioMech/workflows#1 — discussion
- **data-uploader issues raised after training on 6/12/17** BioMech/fiber-uploader#2
- **How to bind BPMN 2.0 activity with form presentation?** BioMech/workflows#2 — discussion
- **Move discussion issues to wiki** BioMech/services#42 — Low Priority, documentation
- **GenICam for interfacing cameras** BioMech/services#5 — camera, discussion
- **Process in real life** BioMech/BioMechDB#5 — discussion
- **Configuration as separate activity** BioMech/BioMechDB#6 — discussion

To Do 7

- **Data tagging with Article/Study... how results are used later?** BioMech/BioMechDB#3 — Medium Priority, To Do, discussion
- **Assumptions made in Uploader to MinIo** BioMech/mediator#13 — Medium Priority, To Do, data transfer
- **Service restart after CoreOS reboot due to update** BioMech/Cluster#19 — IT infrastructure, Medium Priority, To Do
- **Mongo with data on NFS** BioMech/Cluster#25 — Blocked, Medium Priority, To Do
- **Mongo data not persistent across nodes** BioMech/Cluster#30 — IT infrastructure, Low Priority, To Do, enhancement
- **Remove etcd service from cloud-config** BioMech/Cluster#40 — To Do, enhancement
- **synchronisation between two measurements** BioMech/services#43 — IT infrastructure, To Do, camera, enhancement

Closed 187

- **cloud-config vs Container Linux Config** BioMech/Cluster#1 — Doing, High Priority, IT infrastructure, discussion
- **New mongo database** BioMech/credentials#1
- **Integrate Heleen's comments** BioMech/fiber-dashboard#1
- **Schemas flexibility** BioMech/tamta-remote#1 — enhancement
- **Deploy Uploader to microCT** BioMech/mediator#1
- **Blade server** BioMech/services#1
- **Schema for Researchers and Subjects/Specimens** BioMech/BioMechDB#1 — discussion
- **Deploy Uploader to Dyn. Uniaxial TA** BioMech/mediator#2
- **Deploy Uploader to Biax** BioMech/mediator#3
- **Polish FIBEr Dashboard for Launch Event** BioMech/fiber-dashboard#2

So what is behind the scenes...?

DevOps@MECH & MECH Cloud

In-house cloud infrastructure to support research labs @MECH - KU Leuven

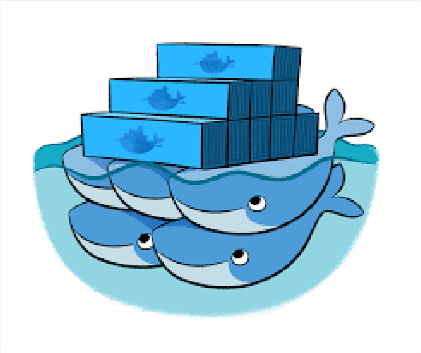
Enabling secure data management, application deployment, data processing, simulations

Set up and support by 1-2 RSEs (yes, that includes me :-))

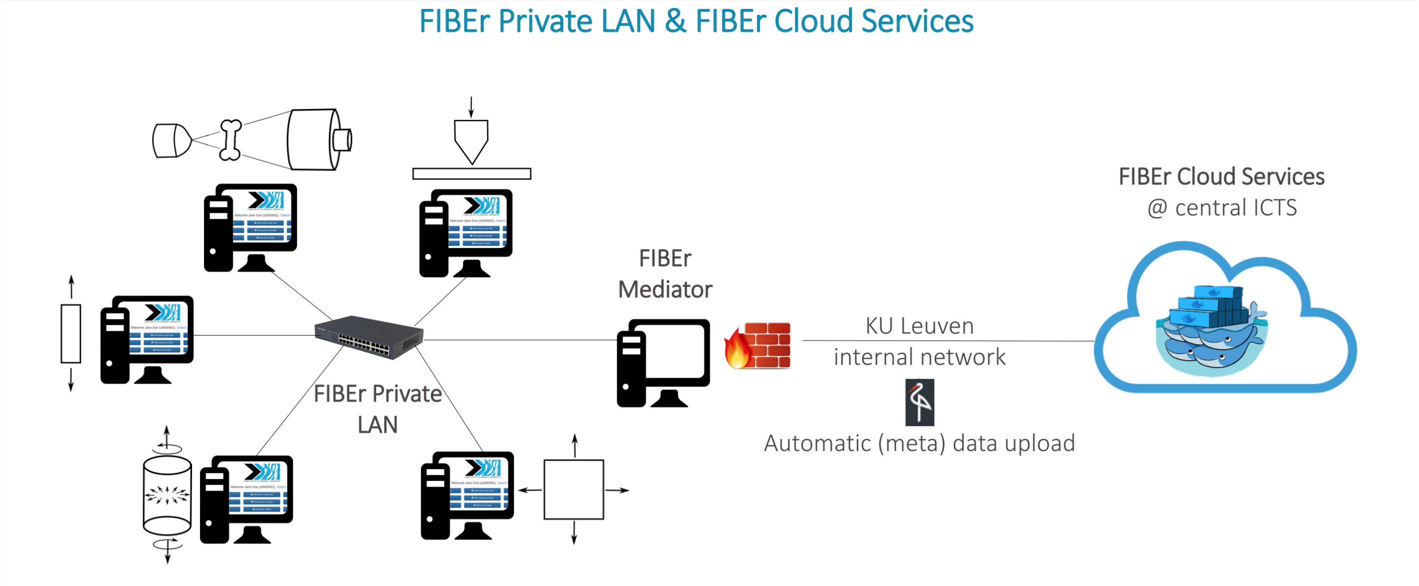
network
robotics
educating
project-driven
Sport.Vlaanderen
modelling
GDPR
support
DevOps
BMe
management
continuity
CIB
DMP
FaBeR
tools
ICT
data
ResOps
research
tailored
PMA
software
legal
FIBEr
guidance
templates
workflows
lectures
CI/CD
TME
privacy
customised
framework
security
ethics
DPO

Servicing already > 10 research groups

Backed by Docker Swarm and GitLab CI/CD!

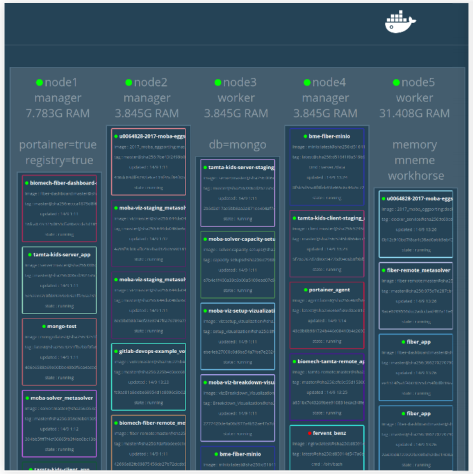


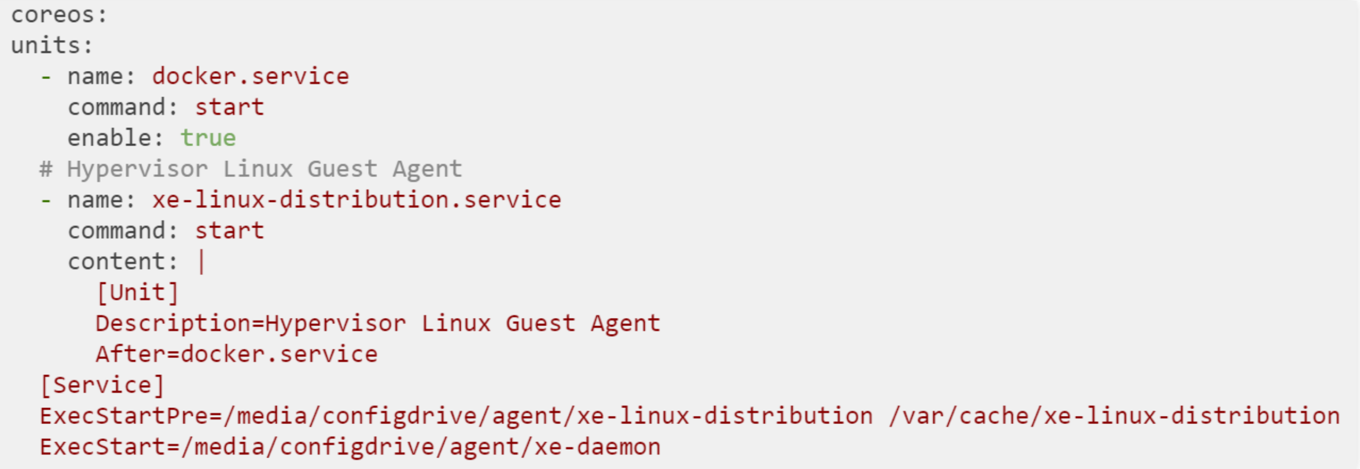
FIBEr setup


FIBEr Private LAN & FIBEr Cloud Services
FIBEr Cloud Services
@ central ICTS
FIBEr
Mediator
KU Leuven
internal network
Automatic (meta) data upload
FIBEr Private
LAN

Hardware backend

5 CoreOS nodes on Intel Xeon E5-2640 v4, 25M Cache, 2.40 GHz
480 GB SSD, 192 GB RAM, 1 TB NFS
Provisioned with XenCenter & Cloud Config





```
coreos:
units:
 - name: docker.service
   command: start
   enable: true
 # Hypervisor Linux Guest Agent
 - name: xe-linux-distribution.service
   command: start
   content: |
    [Unit]
    Description=Hypervisor Linux Guest Agent
    After=docker.service
 [Service]
 ExecStartPre=/media/configdrive/agent/xe-linux-distribution /var/cache/xe-linux-distribution
 ExecStart=/media/configdrive/agent/xe-daemon
```

Docker Swarm configuration

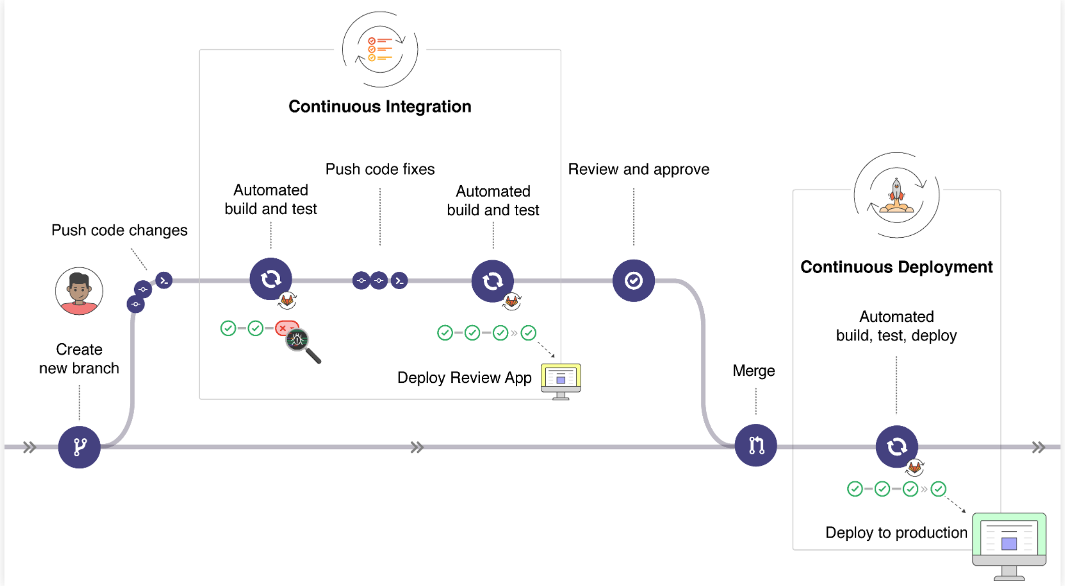
```
$ docker swarm init
Swarm initialized: current node (ip9w0ds01ius3eryxuj3mluus) is now a manager.

To add a worker to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-3e0hh0jd5t4yjg209f4g5qpowbsczfahv2dea9a1ay2l8787cf-2h4ly330d0j917ocvzw30j5x9 10.112.72.1

To add a manager to this swarm, run 'docker swarm join-token manager' and follow the instructions.
```

```
$ docker node ls
ID                            HOSTNAME            STATUS              AVAILABILITY        MANAGER STATUS      ENGINE VERSION
ip9w0ds01ius3eryxuj3mluus *   node1               Ready               Active              Leader              18.06.3-ce
wceftxg05cac28fsa1x28752r     node2               Ready               Active              Reachable           18.06.3-ce
760rkmwbhq4yqydqztxurrlbv     node3               Ready               Active                                  18.06.3-ce
tnmxgdbkexqaecckzzvew1bee     node4               Ready               Active                                  18.06.3-ce
7cfpcywjc5v3wybjexwgj3qfk     node5               Ready               Active              Reachable           18.06.3-ce
```

Docker daemon socket TLS protection: https://docs.docker.com/engine/security/https/

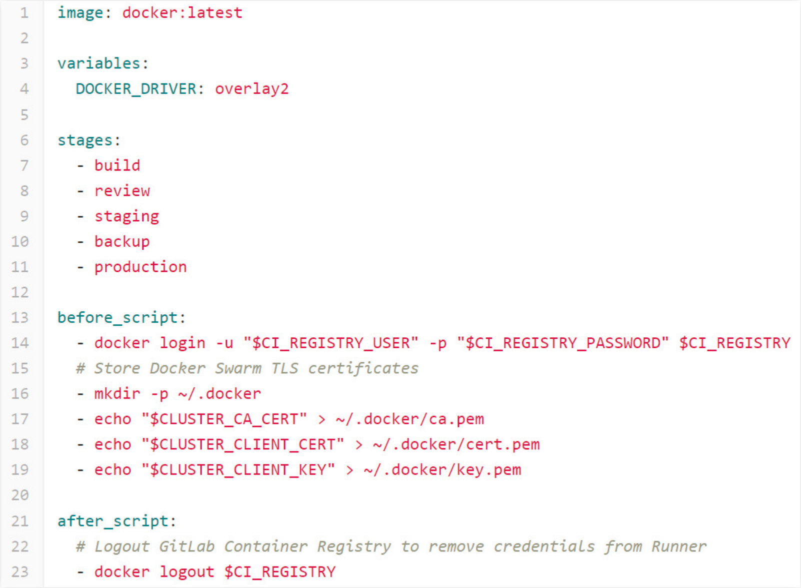
Docker Swarm Integration with GitLab CI/CD workflow

Declarative specification of GitLab CI pipeline



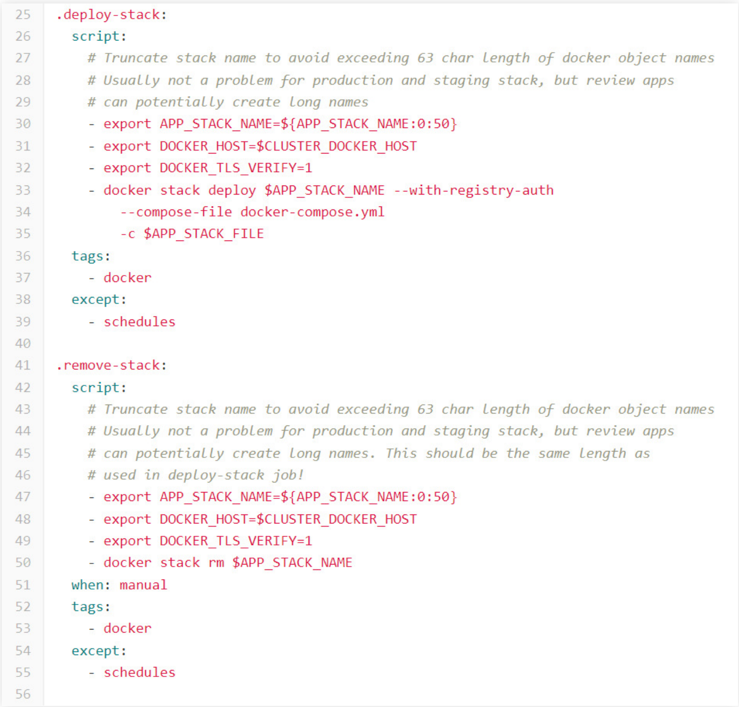
Source: https://about.gitlab.com/product/continuous-integration/abay

GitLab CI: the basics for Docker Swarm integration



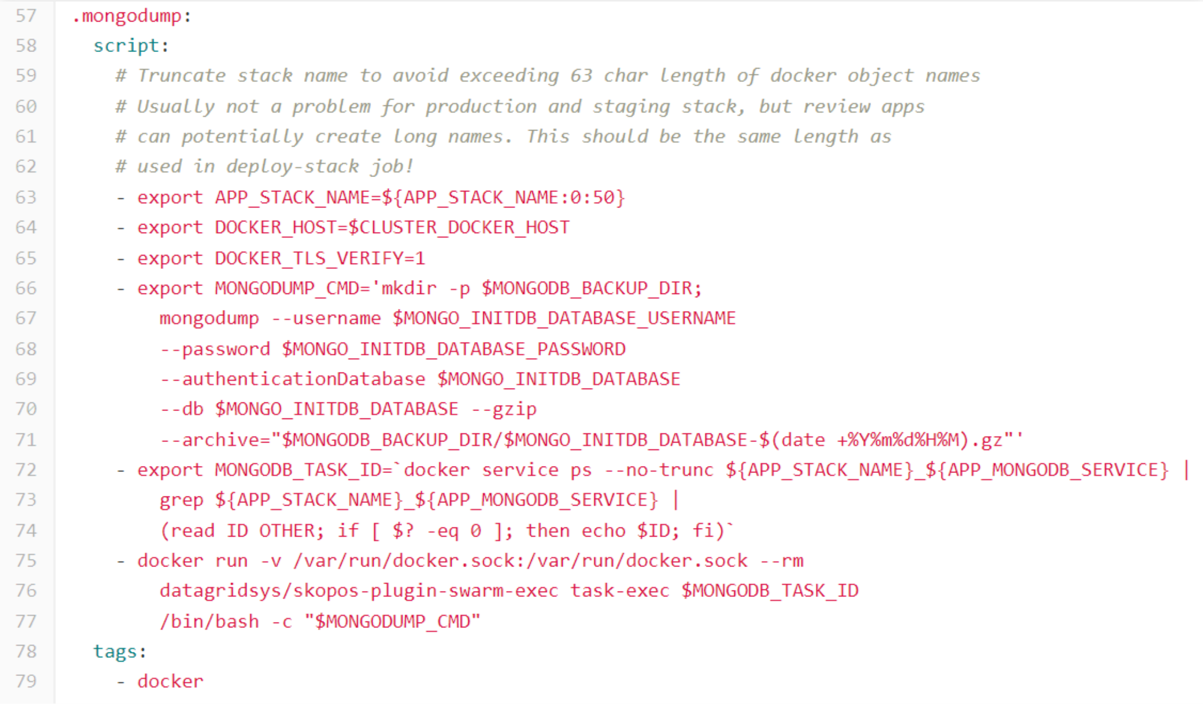
```yaml
image: docker:latest

variables:
  DOCKER_DRIVER: overlay2

stages:
  - build
  - review
  - staging
  - backup
  - production

before_script:
  - docker login -u "$CI_REGISTRY_USER" -p "$CI_REGISTRY_PASSWORD" $CI_REGISTRY
  # Store Docker Swarm TLS certificates
  - mkdir -p ~/.docker
  - echo "$CLUSTER_CA_CERT" > ~/.docker/ca.pem
  - echo "$CLUSTER_CLIENT_CERT" > ~/.docker/cert.pem
  - echo "$CLUSTER_CLIENT_KEY" > ~/.docker/key.pem

after_script:
  # Logout GitLab Container Registry to remove credentials from Runner
  - docker logout $CI_REGISTRY
```

GitLab CI: templates for Docker Stack deployment



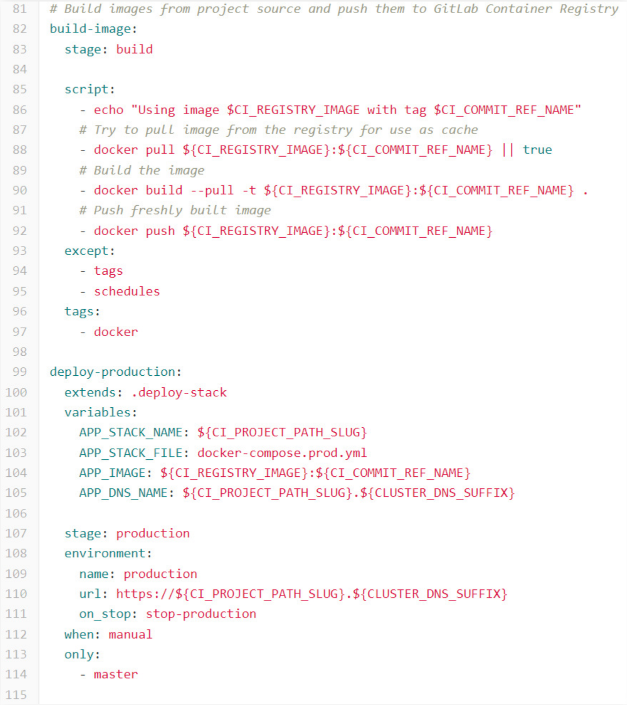
```
.deploy-stack:
  script:
    # Truncate stack name to avoid exceeding 63 char length of docker object names
    # Usually not a problem for production and staging stack, but review apps
    # can potentially create long names
    - export APP_STACK_NAME=${APP_STACK_NAME:0:50}
    - export DOCKER_HOST=$CLUSTER_DOCKER_HOST
    - export DOCKER_TLS_VERIFY=1
    - docker stack deploy $APP_STACK_NAME --with-registry-auth
        --compose-file docker-compose.yml
        -c $APP_STACK_FILE
  tags:
    - docker
  except:
    - schedules

.remove-stack:
  script:
    # Truncate stack name to avoid exceeding 63 char length of docker object names
    # Usually not a problem for production and staging stack, but review apps
    # can potentially create long names. This should be the same length as
    # used in deploy-stack job!
    - export APP_STACK_NAME=${APP_STACK_NAME:0:50}
    - export DOCKER_HOST=$CLUSTER_DOCKER_HOST
    - export DOCKER_TLS_VERIFY=1
    - docker stack rm $APP_STACK_NAME
  when: manual
  tags:
    - docker
  except:
    - schedules
```

GitLab CI: template for MongoDB backup

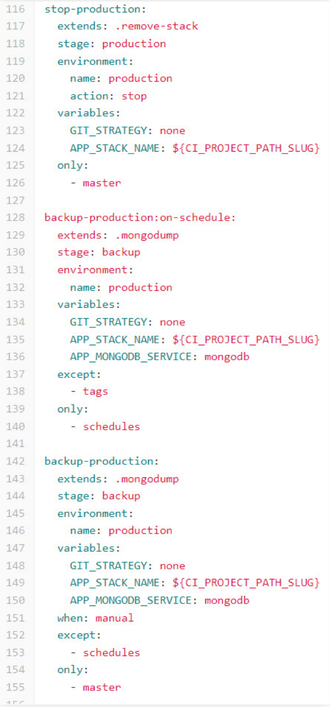


```
.mongodump:
  script:
    # Truncate stack name to avoid exceeding 63 char length of docker object names
    # Usually not a problem for production and staging stack, but review apps
    # can potentially create long names. This should be the same length as
    # used in deploy-stack job!
    - export APP_STACK_NAME=${APP_STACK_NAME:0:50}
    - export DOCKER_HOST=$CLUSTER_DOCKER_HOST
    - export DOCKER_TLS_VERIFY=1
    - export MONGODUMP_CMD='mkdir -p $MONGODB_BACKUP_DIR;
        mongodump --username $MONGO_INITDB_DATABASE_USERNAME
        --password $MONGO_INITDB_DATABASE_PASSWORD
        --authenticationDatabase $MONGO_INITDB_DATABASE
        --db $MONGO_INITDB_DATABASE --gzip
        --archive="$MONGODB_BACKUP_DIR/$MONGO_INITDB_DATABASE-$(date +%Y%m%d%H%M).gz"'
    - export MONGODB_TASK_ID=`docker service ps --no-trunc ${APP_STACK_NAME}_${APP_MONGODB_SERVICE} |
        grep ${APP_STACK_NAME}_${APP_MONGODB_SERVICE} |
        (read ID OTHER; if [ $? -eq 0 ]; then echo $ID; fi)`
    - docker run -v /var/run/docker.sock:/var/run/docker.sock --rm
        datagridsys/skopos-plugin-swarm-exec task-exec $MONGODB_TASK_ID
        /bin/bash -c "$MONGODUMP_CMD"
  tags:
    - docker
```

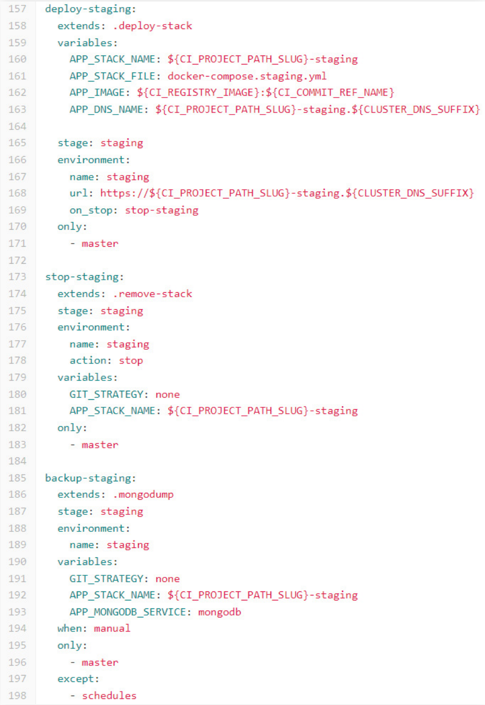
GitLab CI: Docker Image integration via Container Registry and deployment environments



```yaml
# Build images from project source and push them to GitLab Container Registry
build-image:
  stage: build

  script:
    - echo "Using image $CI_REGISTRY_IMAGE with tag $CI_COMMIT_REF_NAME"
    # Try to pull image from the registry for use as cache
    - docker pull ${CI_REGISTRY_IMAGE}:${CI_COMMIT_REF_NAME} || true
    # Build the image
    - docker build --pull -t ${CI_REGISTRY_IMAGE}:${CI_COMMIT_REF_NAME} .
    # Push freshly built image
    - docker push ${CI_REGISTRY_IMAGE}:${CI_COMMIT_REF_NAME}
  except:
    - tags
    - schedules
  tags:
    - docker

deploy-production:
  extends: .deploy-stack
  variables:
    APP_STACK_NAME: ${CI_PROJECT_PATH_SLUG}
    APP_STACK_FILE: docker-compose.prod.yml
    APP_IMAGE: ${CI_REGISTRY_IMAGE}:${CI_COMMIT_REF_NAME}
    APP_DNS_NAME: ${CI_PROJECT_PATH_SLUG}.${CLUSTER_DNS_SUFFIX}

  stage: production
  environment:
    name: production
    url: https://${CI_PROJECT_PATH_SLUG}.${CLUSTER_DNS_SUFFIX}
    on_stop: stop-production
  when: manual
  only:
    - master

```



```yaml
stop-production:
  extends: .remove-stack
  stage: production
  environment:
    name: production
    action: stop
  variables:
    GIT_STRATEGY: none
    APP_STACK_NAME: ${CI_PROJECT_PATH_SLUG}
  only:
    - master

backup-production:on-schedule:
  extends: .mongodump
  stage: backup
  environment:
    name: production
  variables:
    GIT_STRATEGY: none
    APP_STACK_NAME: ${CI_PROJECT_PATH_SLUG}
    APP_MONGODB_SERVICE: mongodb
  except:
    - tags
  only:
    - schedules

backup-production:
  extends: .mongodump
  stage: backup
  environment:
    name: production
  variables:
    GIT_STRATEGY: none
    APP_STACK_NAME: ${CI_PROJECT_PATH_SLUG}
    APP_MONGODB_SERVICE: mongodb
  when: manual
  except:
    - schedules
  only:
    - master
```



```yaml
deploy-staging:
  extends: .deploy-stack
  variables:
    APP_STACK_NAME: ${CI_PROJECT_PATH_SLUG}-staging
    APP_STACK_FILE: docker-compose.staging.yml
    APP_IMAGE: ${CI_REGISTRY_IMAGE}:${CI_COMMIT_REF_NAME}
    APP_DNS_NAME: ${CI_PROJECT_PATH_SLUG}-staging.${CLUSTER_DNS_SUFFIX}

  stage: staging
  environment:
    name: staging
    url: https://${CI_PROJECT_PATH_SLUG}-staging.${CLUSTER_DNS_SUFFIX}
    on_stop: stop-staging
  only:
    - master

stop-staging:
  extends: .remove-stack
  stage: staging
  environment:
    name: staging
    action: stop
  variables:
    GIT_STRATEGY: none
    APP_STACK_NAME: ${CI_PROJECT_PATH_SLUG}-staging
  only:
    - master

backup-staging:
  extends: .mongodump
  stage: staging
  environment:
    name: staging
  variables:
    GIT_STRATEGY: none
    APP_STACK_NAME: ${CI_PROJECT_PATH_SLUG}-staging
    APP_MONGODB_SERVICE: mongodb
  when: manual
  only:
    - master
  except:
    - schedules
```

GitLab CI integration: overview

```
image: docker:latest

variables:
  DOCKER_DRIVER: overlay2

stages:
  - build
  - review
  - staging
  - backup
  - production

before_script:
  - docker login -u "$CI_REGISTRY_USER" -p "$CI_REGISTRY_PASSWORD" $CI_REGISTRY
  # Store Docker Swarm TLS certificates
```

Docker Stacks

Declarative specification of Docker elements
E.g. HTTP reverse proxy and load balancer Traefik:

```
version: "3.3"

services:
  traefik:
    image: traefik:alpine
    command: --web
    ports:
      - "80:80"
      - "8080:8080"
      - "443:443"
    volumes:
      - traefik_logs:/logs
      - /var/run/docker.sock:/var/run/docker.sock
    #labels:
    #  - "traefik.enable=false"
```

Lessons learned using GitLab CI/CD and Docker Swarm in research…

The Good...

Declarative workflows combined with version control!

Automated deployment of various research workflows

GitLab CI templating allows you to easily reuse and extend

GitLab is a great research tool (software PM, version control, CI/CD, automation, ...)!

Greatly improved research software and research data management

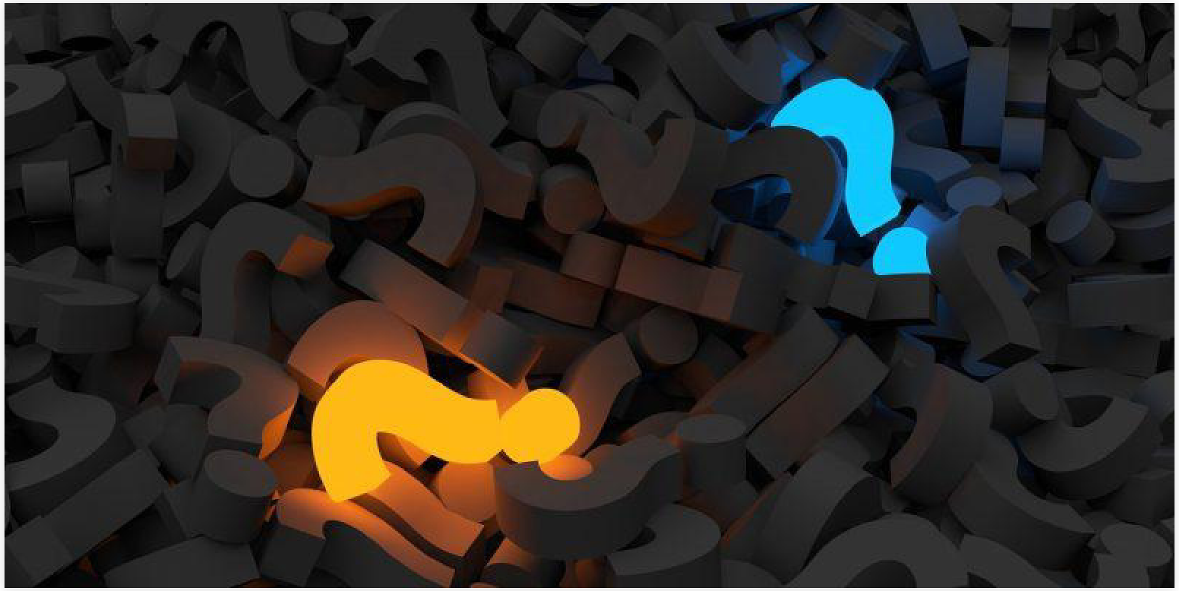
The Bad...

High learning curve from rapid prototyping to production
Yet Another Management tool for researchers to learn
Research is diverse, so difficult to develop generic tooling

... and the Ugly!

Docker Swarm / CoreOS combo not reliable...
Docker storage management is messy and requires frequent manual clean up
Discipline is required by researchers to optimally improve research reproducibility

Questions?



Source: Pixabay