TOUCHSCREEN

select u.id as userID, u.ip as IP, u.createAt as userCreatedAT, e.event as actionName, e.value as actionValue, e.createAt as actionAT from events e

right join users u on u.id = e.user

where date(u.createAt) = '2021-11-08';

Run it here:

http://46.101.32.14:1061/db\_sql.php?db=eha2020&token=fb5dda469c4363b79b75342fe236ce0f

# SYMPOSIUM BACKEND DB:

COLLECTOR

<http://postgres.phaseiilabs.com/browser/>

# DMCS

+bin/console phaseii:import:ct --did=34 --env=dev -v

+bin/console phaseii:filter:ct:new --did=34 --env=dev -v

+bin/console p:i:p --did=34 --env=dev -v

+bin/console phaseii:filter:pm --did=34 --env=dev -v

+bin/console phaseii:filter:journal --did=34 --env=dev -v

+bin/console phaseii:filter:sponsor --deploymentId=34 --env=dev -v

+bin/console phaseii:filter:intervention --deploymentId=34 --env=dev -v

+bin/console phaseii:filter:kol --did=34 --env=dev -v --contact

+bin/console phaseii:merge:dump --env=dev -v --did=34

+bin/console phaseii:merge:experts --env=dev -v --did=34

+bin/console phaseii:statistics --env=dev -v --did=34

+bin/console phaseii:import:community --deploymentId=34 --justNulls=1 --env=dev –v

+bin/console phaseii:filter:community --deploymentId=34 --clearTable=1 --env=dev -v

+bin/console p:m:r --env=dev -v --did=34

+bin/console phaseii:kol:p2id --env=dev -v

+/bin/bash cron.sh -s="bin/console phaseii:extract:experts --deploymentId=1,16,29,9,24,22,35,36,37,39,33,34,40 --env=dev -v" -l="./es\_extract\_experts.log"

+/bin/bash cron.sh -s="bin/console phaseii:extract:pubs --env=dev -v" -l="./es\_extract\_pubs.log"

+/bin/bash cron.sh -s="bin/console phaseii:extract:trials --env=dev -v" -l="./es\_extract\_trials.log"

+/bin/bash cron.sh -s="bin/console phaseii:extract:centers --deploymentId=1,16,29,9,24,22,35,36,37,39,33,34,40 --env=dev -v" -l="./es\_extract\_centers.log"

# GCA

## Local

* yarn
* make doc
* make dev

SQL TRACKING:

SELECT name, count(id) counter FROM `downloads` group by name ORDER BY counter desc;

<http://46.101.32.14:1061/tbl_sql.php?db=gca&table=downloads>

HUBs

**Hubs admin access**

<https://lymphomahub.com/>: [admin@gmail.com](mailto:admin@gmail.com): 9WYOqP0Nd84YoQU9p

<https://aml-hub.com/>: [admin@gmail.com](mailto:admin@gmail.com) : adminpiipasswd#

<https://gvhdhub.com/>: [admin@gmail.com](mailto:admin@gmail.com) : adminpiipasswd#

<https://lymphoblastic-hub.com/> - [admin@gmail.com](mailto:admin@gmail.com) : allpasswdadmin56

<https://multiplemyelomahub.com/> - [admin@gmail.com](mailto:admin@gmail.com) : adminpiipasswd#

<https://mds-hub.com/> - [admin@gmail.com](mailto:admin@gmail.com): mdspasswdadmin59

<https://mpn-hub.com/> - [admin@gmail.com](mailto:admin@gmail.com) : adminpiipasswd#

## Release

* in Jenkins choose the .env and hit release

## ~~Backups (moved to readme repo)~~

* ~~On Kubernetes cluster there are jobs and cronjobs entries. Cronjobs should run on a scheduled time whereas the jobs are one-off actions.~~
* ~~doctl kubernetes cluster kubeconfig save k8s-hubs~~
* ~~k get cronjob –~~ *~~just to see the current cronjobs~~*
* ~~go to the hubs project dir~~
* ~~switch to a branch you want to make a backup of, eg. hub\_aml~~
* ~~cd docker/kbuild/backup~~
* ~~run `make deploy` or `make pod` (this will just list all the commands that can be run)~~
  + *~~make deploy~~* ~~will print the commands in order for you to release a new version on a~~ **~~cronjob~~**
  + *~~make pod~~* ~~will print commands that you can~~ **~~execute~~** ~~in order to make a backup immediately~~
* ~~copy and paste, for example:~~
* ~~/bin/bash cron.yaml.sh .env.kub.prod backupmysql.sh~~

Setup

* yarn
* make preprocessor
* make doc
* /bin/bash bash/mysql/copytablesbetweendatabases.sh .env.kub.prod .env --force
* make dev

Add new migration

* add/change columns in for example: runtime/migrations/src/entity/Tags.ts
* cd migrations
* make diff (this will generate a new migration file)
* remove all rubbish
* make torun (to check how many migrations will be executed next time)
* make migrate

## Monorepo setup:

(basically it is best to follow this readme file: <https://bitbucket.org/scientificeducationsupport/hub/src/master/INSTALLATION-MONOREPO.md>)

* Once you produce a code that can be shared across all the branches, simply switch from your working branch (say hub\_lh or api\_hub\_lh) to either master, or hub\_master or api\_master. If you go for master, your change will spread across all the children branches. If you go for hub\_master or api\_master then the code will only be shared for theirs sub branches only
* Once on \*master, commit (you can also push it, eg: **git push hub hub\_master**) and run `make merge`, press `e`
* If conflicts, resolve them, commit, and hit enter in CLI

## New HUB eg. xxx hub

* Create a branch based on eg. hub\_lh, or api\_hub\_lh
* Push the branch to a remote, eg.: git push hub api\_hub\_xxx

Now in order to modify the monorepo config:

* Go to your \_git folder (which was introduced in Monorepo setup)
* Open config.yml file and copy/paste the entries from eg. api\_hub\_lh, rename those to its equivalent, eg:

preventmergeto\_local:

master: # disallowed branches to merge with master

……

- hub\_mpn

- hub\_xxx

………… etc …………

-

type: merge

remotebranch: hub\_master

localbranch: hub\_xxx

default: on

merge:

-

type: pullpush

localbranch: hub\_xxx

remote: hub

remotebranch: hub\_xxx

default: on

Now you have to commit the changes by executing those scripts:

* /bin/bash gitstorage.sh diff
* /bin/bash gitstorage.sh push (this will show some errors but don’t worry)
* /bin/bash gitstorage.sh push --force

# Kubernetes wildcards domains

Once you have your cluster up and running you have to install a wildcard issuer, like so:

* Go to eg. [secrets\_repo]/optimus/loadbalancers/ingress-nginx/cert-issuers/wildcard-issuers
* Modify the file to your needs: cluster-issuer.yaml
* Run the .sh scrips. This will register the wildcard cluster issuer in to your cluster
* Now you have to issue a certificate for for example stage env projects:
* ~~Go to DO panel and create a subdomain~~
* cd ./certificate-stage/ and adjust the .yaml file to your needs
* apply your changes by running the sh script
* now you can create ingresses per application. They may look like this: optimus/loadbalancers/ingress-nginx/cert-issuers/ingress/knowall/stage/ingress.yaml or:

apiVersion: extensions/v1beta1

kind: Ingress

metadata:

name: ingress-knowall-wildcard-stage-optimus-phaseiilabs

annotations:

kubernetes.io/ingress.class: "nginx"

cert-manager.io/cluster-issuer: "clusterissuer-wildcard-optimus-phaseiilabs"

nginx.ingress.kubernetes.io/proxy-body-size: "0"

nginx.ingress.kubernetes.io/proxy-read-timeout: "600"

nginx.ingress.kubernetes.io/proxy-send-timeout: "600"

namespace: knowall

spec:

tls:

- hosts:

- "\*.stage.optimus2.phaseiilabs.com" # leave it as a wildcard here

secretName: secret-certificate-wildcard-stage-optimus-phaseiilabs

rules:

- host: "knowall.stage.optimus2.phaseiilabs.com"

http:

paths:

- backend:

serviceName: knowall-stage-service

servicePort: 80

ASH2020:

* make zip
* <script type="text/babel" src="/js/common.js"></script>
* ==> <script type="text/javascript" src="/js/common.js"></script>
* "/js/ ==> "js/
* location.href = "/ ===> location.href = "./
* crossorigin="anonymous" ===>
* integrity=" \*\*\*\*\*\* “ ===>

Server:

* npx npx-server
* node server.js --dir .

New Websites test

* H1/h2/h3 for SEO purposes
* GA or GA4
* Check against the designs
* Must be t&c
* Responsiveness
* Skipping between pages should not flash the entire template (eg. Main navigation panel should remain rendered)
* Og image & texts
* Sitemaps
* Favicon
* SSR

NEW HUBS:

* Admins
* Repo
* Assets
* Cloud
* Emails users
* Domain
* Ssl certs
* Backups
* Ci/cd pipelines
* New cluster maybe
* Recaptcha
* GA
* CM
* hotjar

April to have empty hub.

NEW HUB:

* Add git monorepo setting (see above)
* Modify the envs and config files (include cm, captcha otherwise it will brake)
* Add test,stage,prod database and the privileges with full access
* Copy local db structure (migrated) to stage and prod db as test will try to copy over the dbs from there
* Add jenkins entry
* Run first jenkins test manually
* Add mega menu options to the homepage feature

New kubernetes cluster

Kubernetes (<https://github.com/stopsopa/kuber/tree/master/DO/002-nginx-ingress%5Bhttps%5D>)

- create kuber in DO (3nodes, 8ram, 4cpu, 480hdd)

- create folders structure in secrets repo (folder, env inside)

- download the latest ingress deploy.yaml - <https://kubernetes.github.io/ingress-nginx/deploy/#digital-ocean> and put it inside secrets/horizon/loadbalancers/ingress-nginx/deploy.yaml

- add those three lines to a Service:

service.beta.kubernetes.io/do-loadbalancer-enable-proxy-protocol: 'true'

service.beta.kubernetes.io/do-loadbalancer-name: "horizon-loadbalancer"

# service.beta.kubernetes.io/do-loadbalancer-hostname: "horizon.lb.phaseiilabs.com" # check using kubectl get svc -A

- **the third line should be commented out for now**

- also check if `externalTrafficPolicy` flag is set to Local

- make sure the yaml file has the first line empty as the apply.sh script ignorantly removes it

- run /bin/bash deploy.yaml.sh (may throw some errors but those are JS errors, nothing to worry about - one day I have to rebuild the script)

- the above may take some time, especially the **LOAD BALANCER.** Go to the DO panel to see if the lb is ready before you continue

- check if load balancer is up and running: kubectl describe svc --namespace=ingress-nginx

- kubectl get svc -A — this should show external-ip address

- now create a domain for the load balancer in DO panel: horizon.lb.phaseiilabs.com

- redirect that domain to that load balancer

- modify the **deploy.yaml** file and add the domain name:

service.beta.kubernetes.io/do-loadbalancer-hostname: "hubs.lb.scientificeducationsupport.com"

- kubectl get svc -A — this should now show domain name in external-ip column

-

- (option 1)

- go to: <https://cert-manager.io/docs/installation/kubectl/>

- and follow the instruction (technically it’s just one single command, like so: kubectl apply -f <https://github.com/cert-manager/cert-manager/releases/download/v1.10.1/cert-manager.yaml>)

- kubectl get pods --namespace cert-manager — to verify the installation

- You should see the **cert-manager**, cert-manager-**cainjector**, and cert-manager-**webhook** pods in a Running state

- (option 2)

- find a new version of a cert-manager: <https://github.com/cert-manager/cert-manager/tags>

- execute: kubectl apply --validate=false -f <https://github.com/cert-manager/cert-manager/releases/download/v1.7.2/cert-manager.yaml>

- (option 3)

- using helm: <https://cert-manager.io/docs/installation/helm/>

-

-

- copy docker-registry-secret from other secret/clusters

- run /bin/bash regcred.sh

-

-

- PROD ISSUER:

- Now we must copy the stage and prod cert issuers to: horizon/loadbalancers/ingress-nginx/cert-issuers (prod.yaml + prod.yaml.sh + stage....) - take a look here for the proper yml schema: <https://cert-manager.io/docs/configuration/acme/>

- run /bin/bash prod.yaml.sh (we won’t be using stage for now)

-

-

- in order to use below wildcard reflector (mechanism to copy/sync certs/secrets between the namespaces we have to install: kubectl -n kube-system apply -f https://github.com/emberstack/kubernetes-reflector/releases/latest/download/reflector.yaml

- also we have to include annotation in certificate.yaml (the one below in red):

spec:

secretName: secret-certificate-wildcard-stage-horizon-phaseiilabs

…......

**secretTemplate:**

**annotations:**

**reflector.v1.k8s.emberstack.com/reflection-allowed: "true"**

**reflector.v1.k8s.emberstack.com/reflection-allowed-namespaces: ""**

**…........**

-

-

- STAGE **WILDCARD** ISSUER:

- copy/paste optimus/loadbalancers/ingress-nginx/cert-issuers/**wildcard-issuers** folder

- modify cluster-issuer.yaml

- create a subdomain for example: \*.stage.horizon.phaseiilabs.com

- run apply

- go to **wildcard-issuers**/certificate-stage

- modify certificate.yaml

- run apply

-

- kubectl apply -f <https://github.com/kubernetes-sigs/metrics-server/releases/latest/download/components.yaml> to install metrics in order to use kubectl top commands

* Now you can use eg.: **kubectl top pods -A**

-

-

- first app deploy

* Create docker repo secret: kubectl get secret regcred --namespace=default -oyaml | grep -v '^\s\*namespace:\s' | kubectl apply --namespace=heatmap -f - || true
* Change your .env.kub.prod to use a proper cluster name
* Modify your pipeline.yaml for buildkite