Docker Datascience Pipeline

Running datascience models on Docker at ING

BIG DATA Conference Vilnius 2018

Lets introduce myself

- Lennard Cornelis
- Big Data Engineer at ING
- DB2, Oracle, AIX, Linux, Hadoop, Hive, Sqoop, Ansible
- Let it all work on the Exploration Environment
- @chiefware

Docker Hub

docker

Why I think you are here

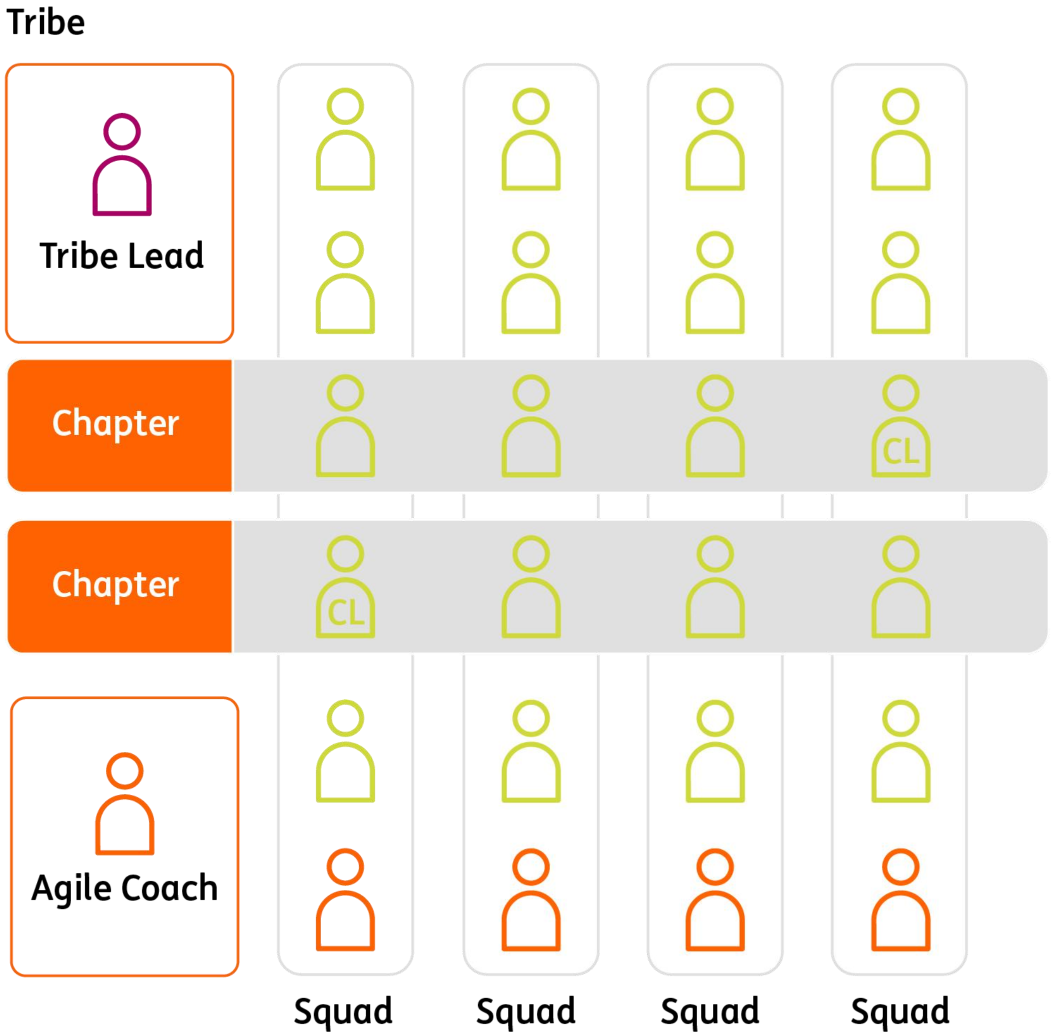
- You are running Datascience models
- Interested in Docker
- Asking yourself how to get to production
- Asking yourself how to get to production

  Depending on different tools

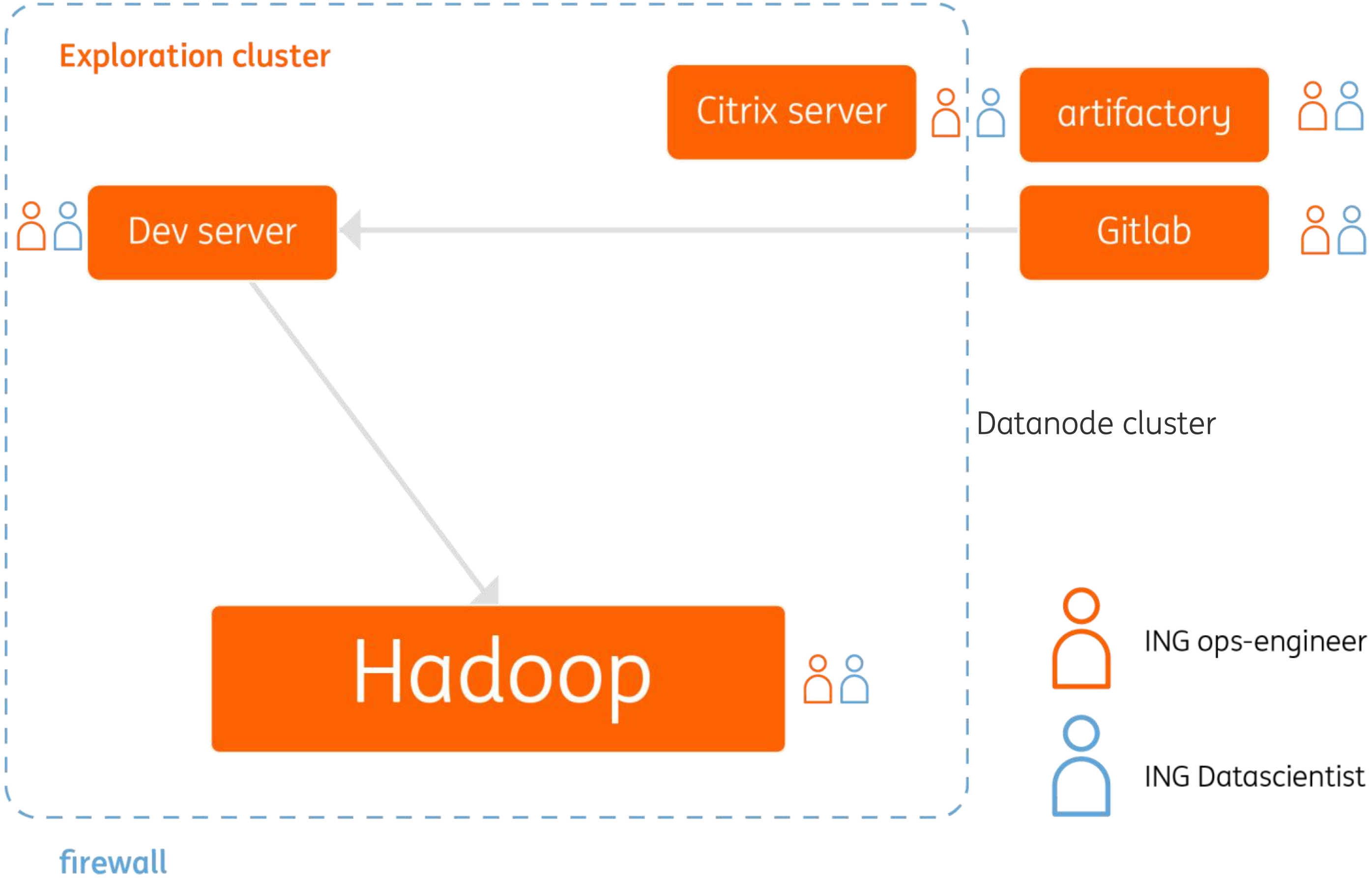
What is Docker

Docker is a tool designed to make it easier to create, deploy, and run applications by using containers. Containers allow a developer to package up an application with all of the parts it needs, such as libraries and other dependencies, and ship it all out as one package. By doing so, thanks to the container, the developer can rest assured that the application will run on any other Linux machine regardless of any customized settings that machine might have that could differ from the machine used for writing and testing the code

Agile way of working in Squads,Chapters and Tribes



big data prototype environment



Exploration cluster

Citrix server

artifactory

Dev server

Gitlab

Hadoop

Datanode cluster

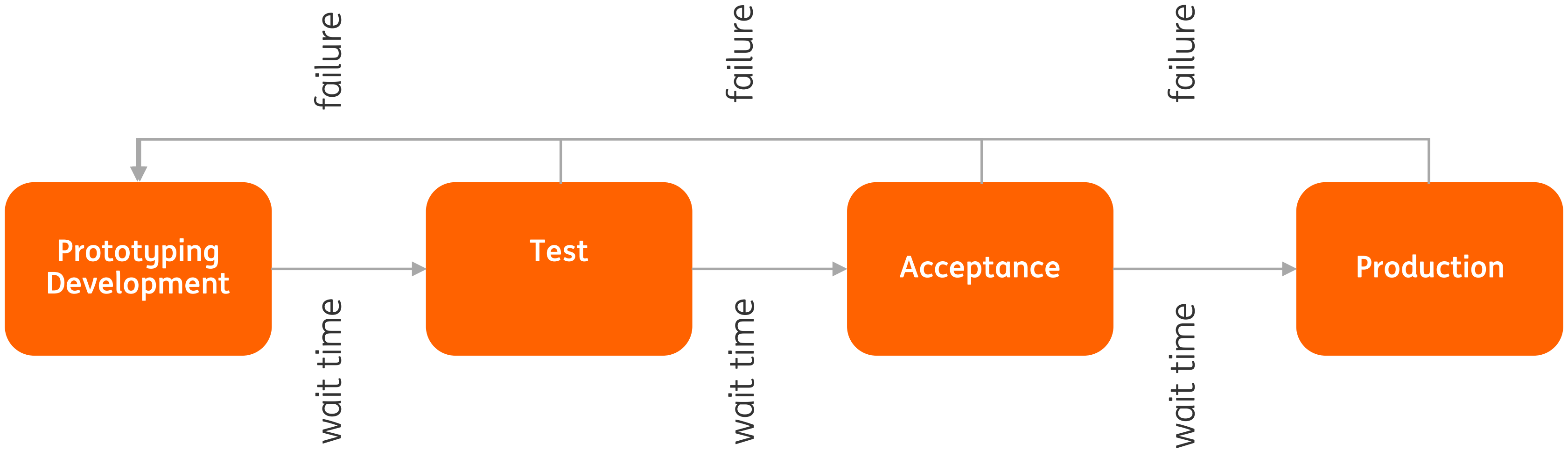
ING ops-engineer

ING Datascientist

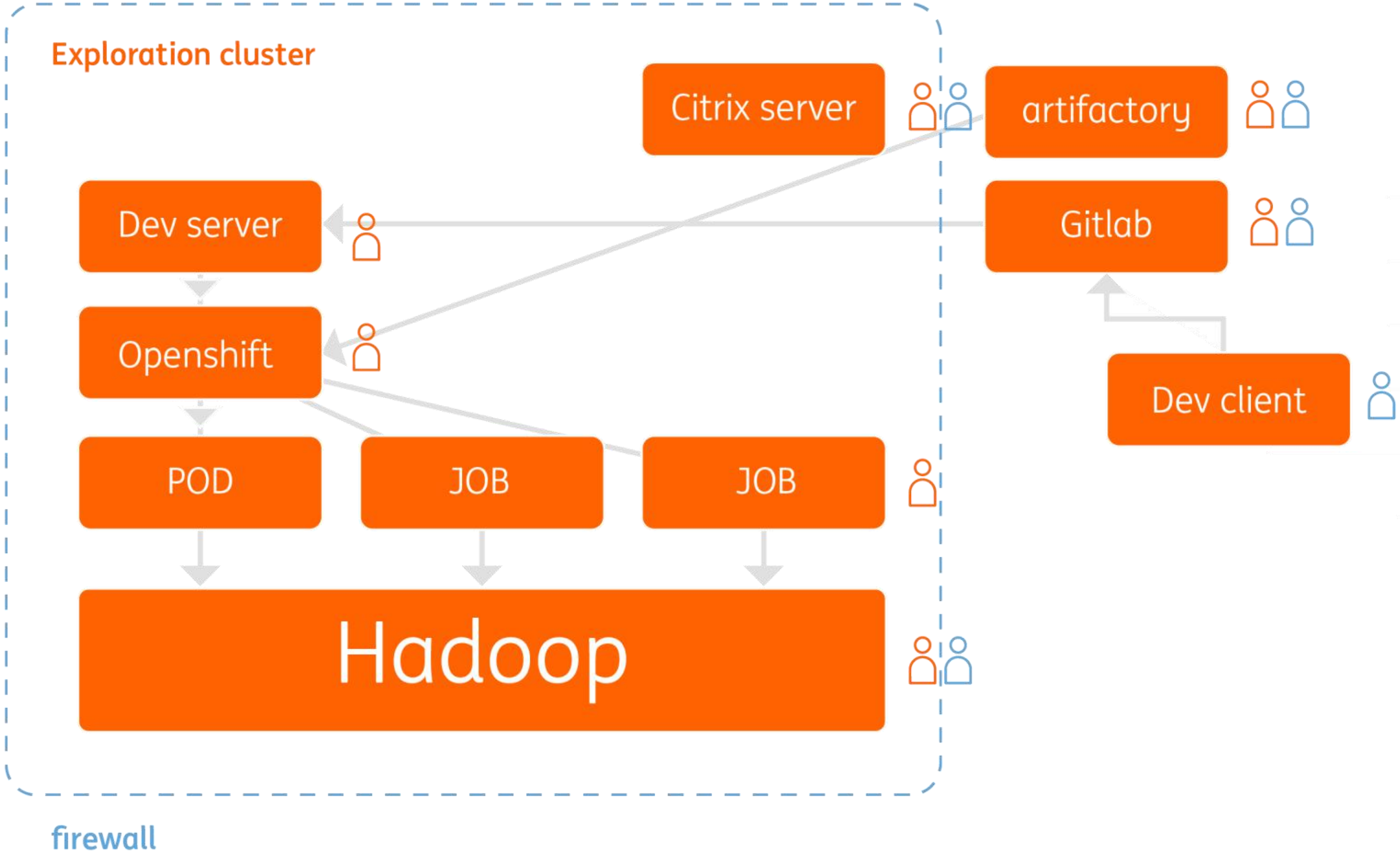
firewall

Server	Details about process
Gitlab runner	Automation Server
Citrix Server	Access to cluster rdp, browser, putty
Dev node	Node for datascientists with tools and xrdp
Hadoop	Datanode cluster
Artifactory	Pip repo
Gitlab	Datasciences projects

Pipeline


failure
failure
failure
Prototyping Development
Test
Acceptance
Production
wait time
wait time
wait time

Docker on big data prototype environment

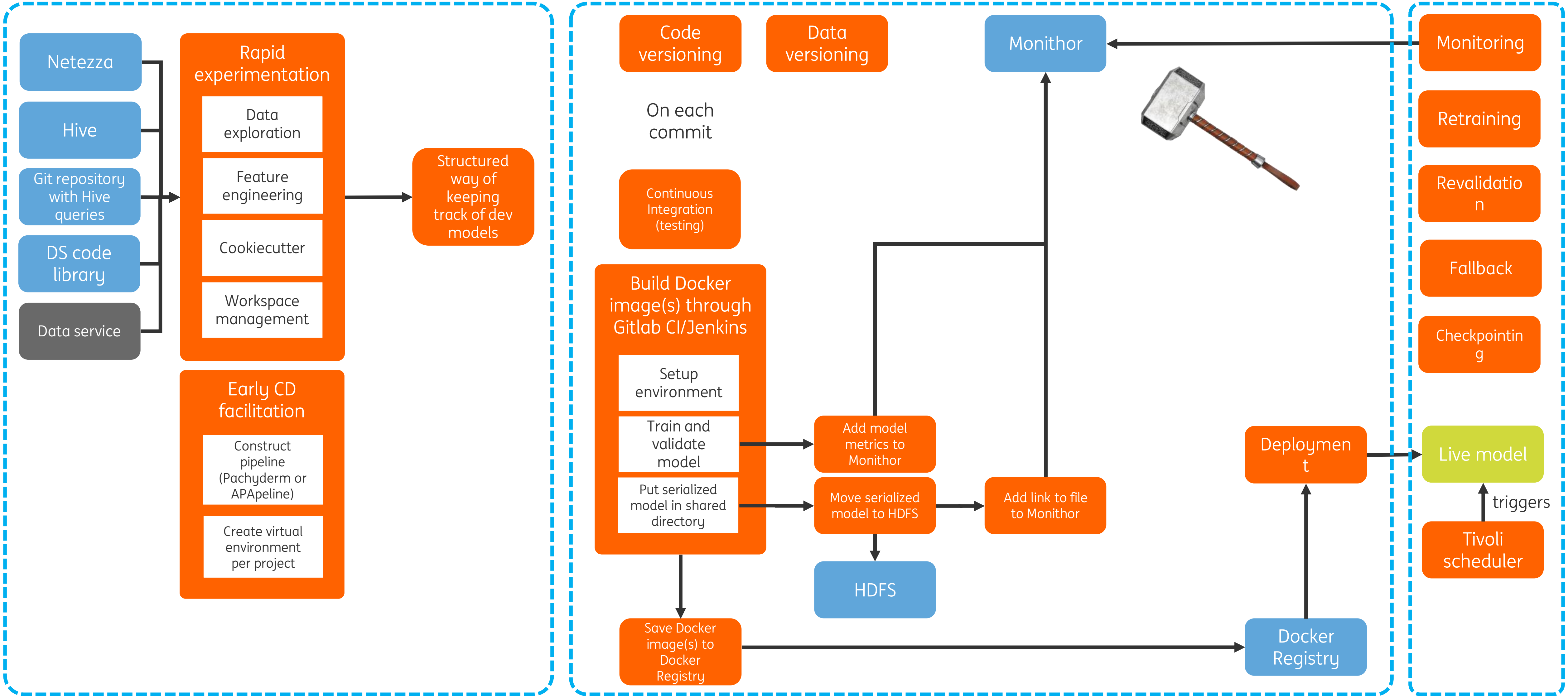


Gitlab runner	Automation Server
Citrix Server	Access to cluster rdp, browser, putty
Dev node	Node for datascientists with tools and xrdp
Hadoop	Datanode cluster
Openshift	Orchestration Containers
POD/JOB	docker containers
Artifactory	Docker base Images
Gitlab	Docker files and Datasciences projects

Development

Continuous Deployement

Live



Netezza

Hive

Git repository with Hive queries

DS code library

Data service

Rapid experimentation
- Data exploration
- Feature engineering
- Cookiecutter
- Workspace management

Structured way of keeping track of dev models

Early CD facilitation
- Construct pipeline (Pachyderm or APApeline)
- Create virtual environment per project

Code versioning

Data versioning

On each commit

Continuous Integration (testing)

Build Docker image(s) through Gitlab CI/Jenkins
- Setup environment
- Train and validate model
- Put serialized model in shared directory

Add model metrics to Monithor

Move serialized model to HDFS

Add link to file to Monithor

HDFS

Monithor

Save Docker image(s) to Docker Registry

Docker Registry

Deploymen t

Monitoring

Retraining

Revalidatio n

Fallback

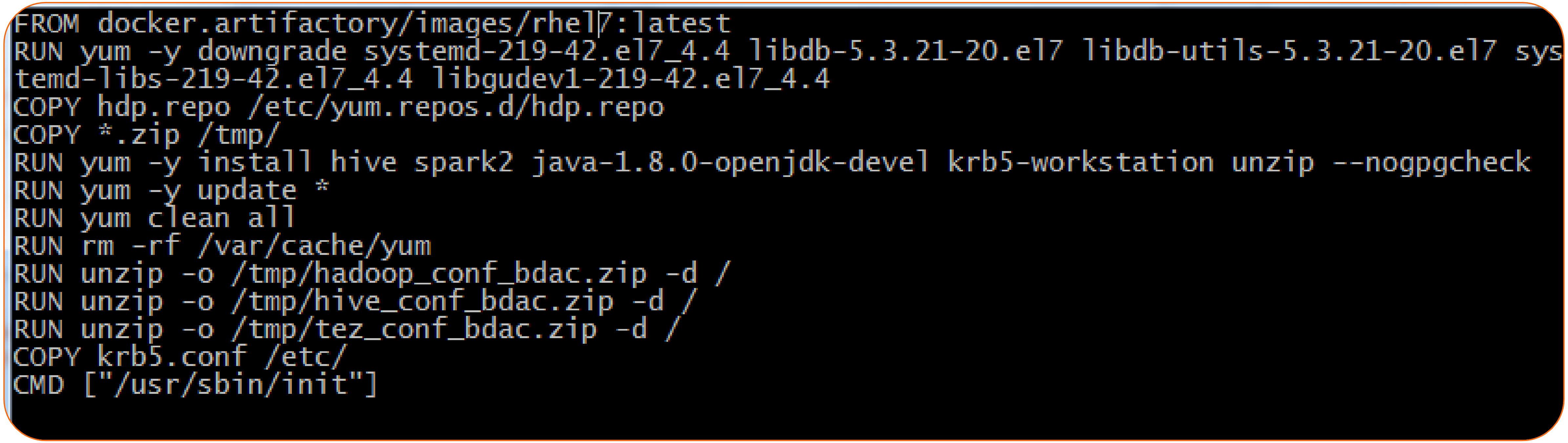
Checkpointin g

Live model

triggers

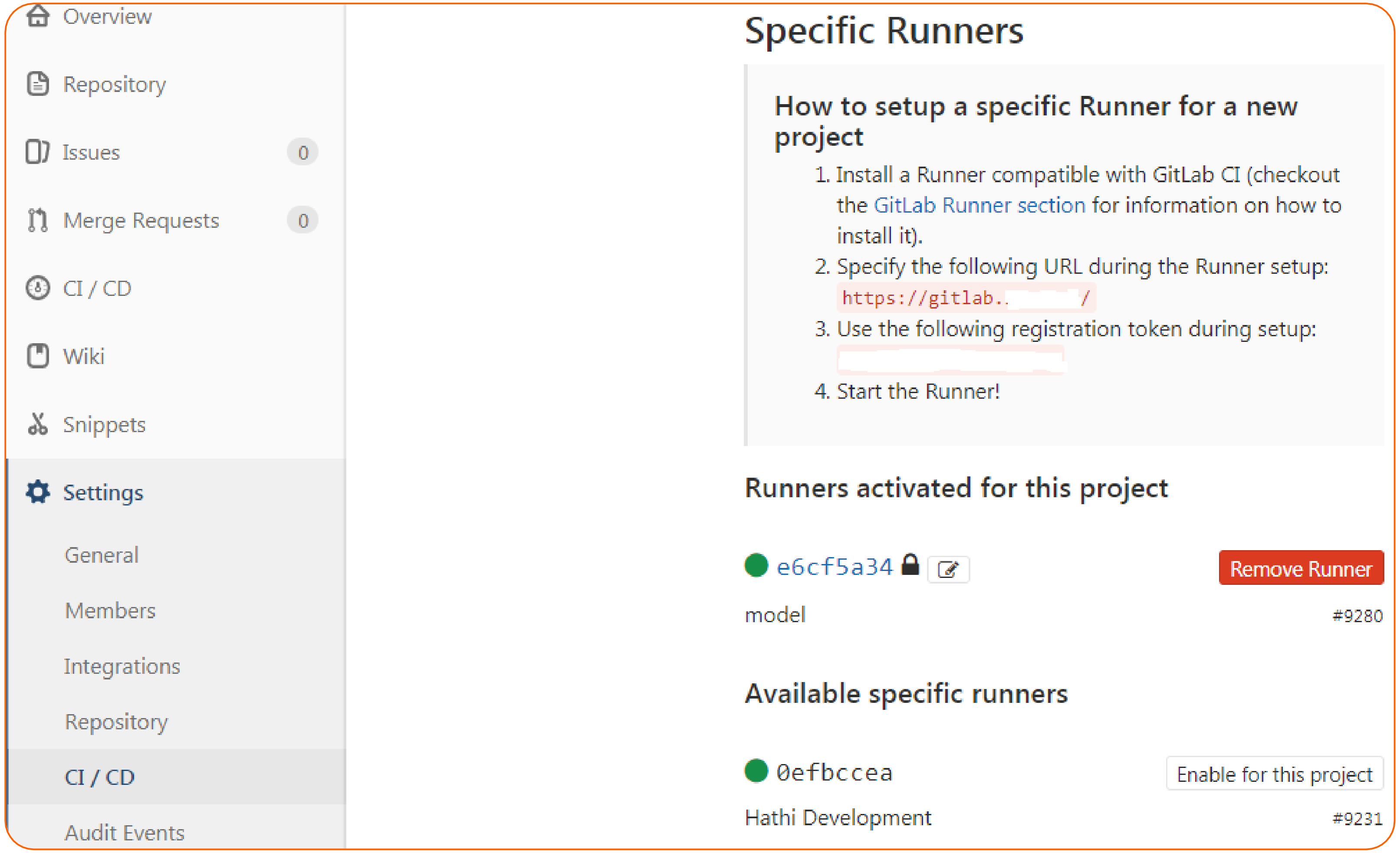
Tivoli scheduler

Dockerfile



```
FROM docker.artifactory/images/rhel7:latest
RUN yum -y downgrade systemd-219-42.el7_4.4 libdb-5.3.21-20.el7 libdb-utils-5.3.21-20.el7 sys
temd-libs-219-42.el7_4.4 libgudev1-219-42.el7_4.4
COPY hdp.repo /etc/yum.repos.d/hdp.repo
COPY *.zip /tmp/
RUN yum -y install hive spark2 java-1.8.0-openjdk-devel krb5-workstation unzip --nogpgcheck
RUN yum -y update *
RUN yum clean all
RUN rm -rf /var/cache/yum
RUN unzip -o /tmp/hadoop_conf_bdac.zip -d /
RUN unzip -o /tmp/hive_conf_bdac.zip -d /
RUN unzip -o /tmp/tez_conf_bdac.zip -d /
COPY krb5.conf /etc/
CMD ["/usr/sbin/init"]
```

Gitlab runner



Specific Runners

How to setup a specific Runner for a new project

1. Install a Runner compatible with GitLab CI (checkout the GitLab Runner section for information on how to install it).
2. Specify the following URL during the Runner setup: `https://gitlab.[redacted]/`
3. Use the following registration token during setup: [redacted]
4. Start the Runner!

Runners activated for this project

● e6cf5a34 🔒

model #9280

Available specific runners

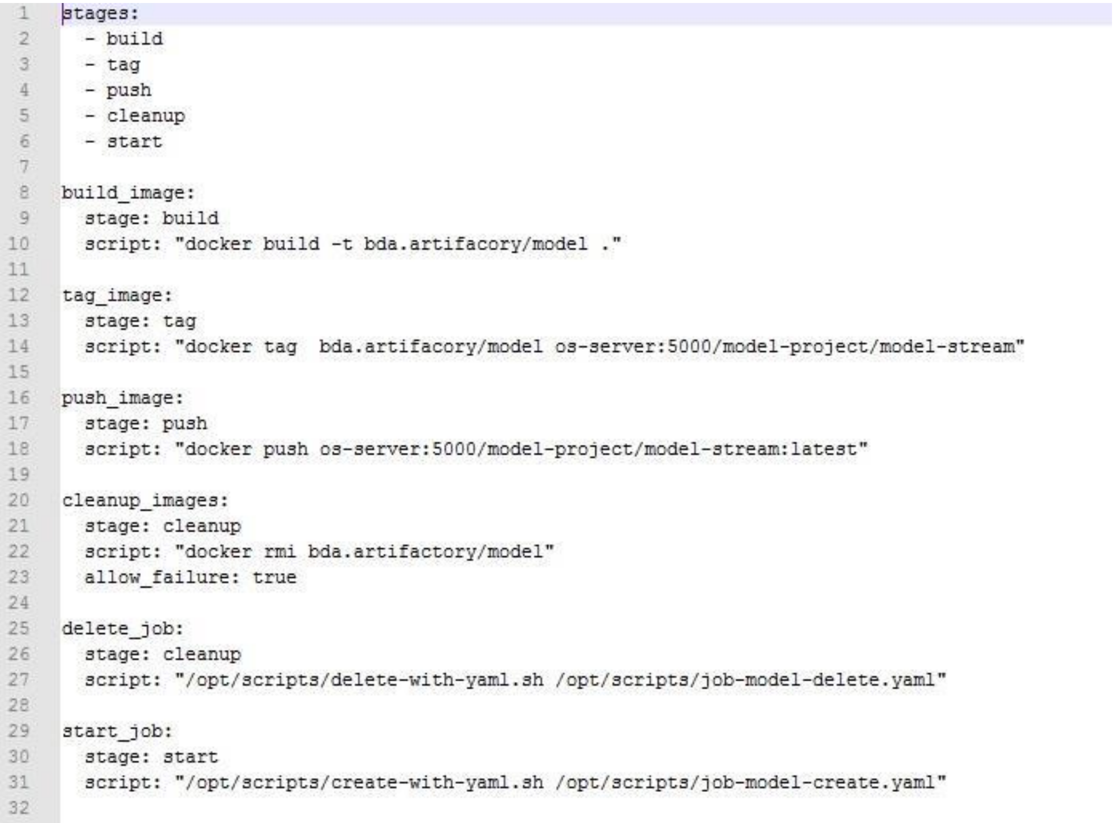
● 0efbccea

Hathi Development #9231

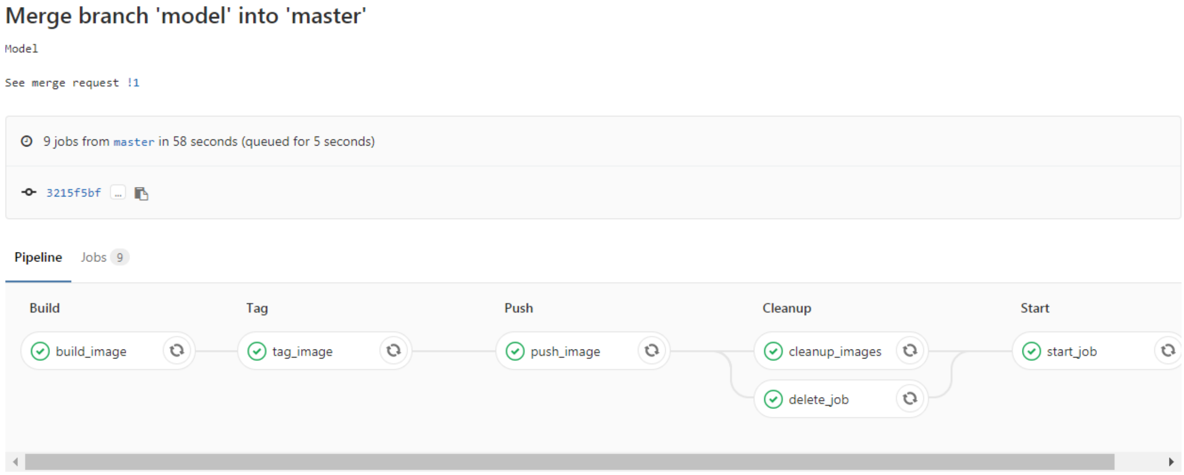
To register run:
gitlab-runner register
gitlab-runner list

Gitlab runner

.gitlab-ci.yml



```
stages:
  - build
  - tag
  - push
  - cleanup
  - start

build_image:
  stage: build
  script: "docker build -t bda.artifacory/model ."

tag_image:
  stage: tag
  script: "docker tag  bda.artifacory/model os-server:5000/model-project/model-stream"

push_image:
  stage: push
  script: "docker push os-server:5000/model-project/model-stream:latest"

cleanup_images:
  stage: cleanup
  script: "docker rmi bda.artifactory/model"
  allow_failure: true

delete_job:
  stage: cleanup
  script: "/opt/scripts/delete-with-yaml.sh /opt/scripts/job-model-delete.yaml"

start_job:
  stage: start
  script: "/opt/scripts/create-with-yaml.sh /opt/scripts/job-model-create.yaml"

```



Merge branch 'model' into 'master'

Model

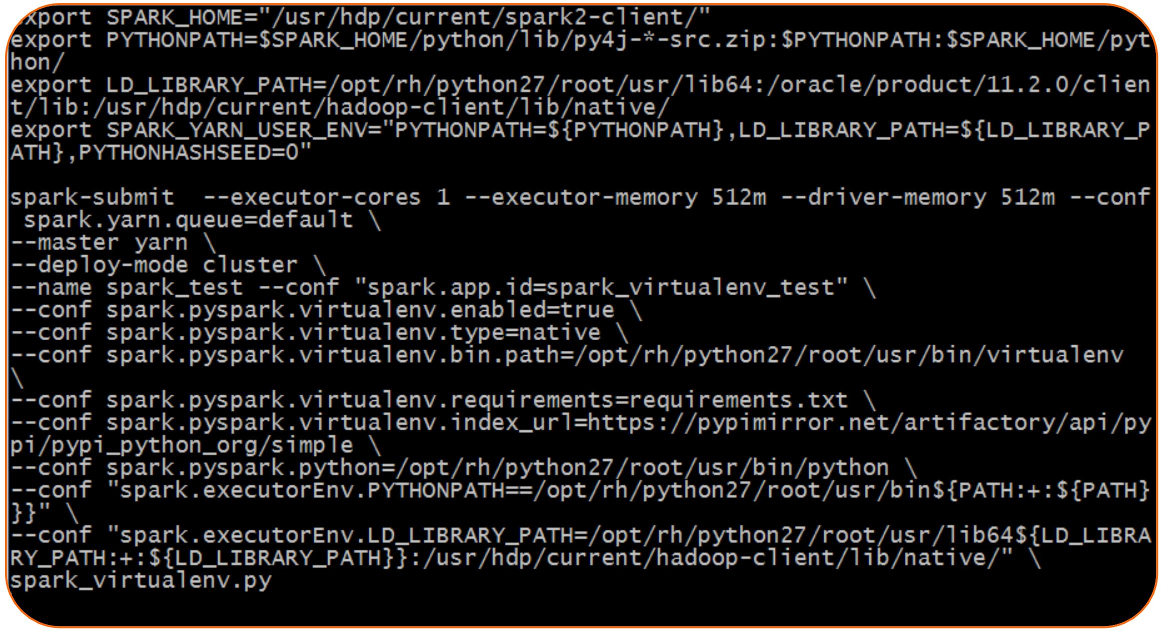
See merge request !1

9 jobs from master in 58 seconds (queued for 5 seconds)

3215f5bf

Spark

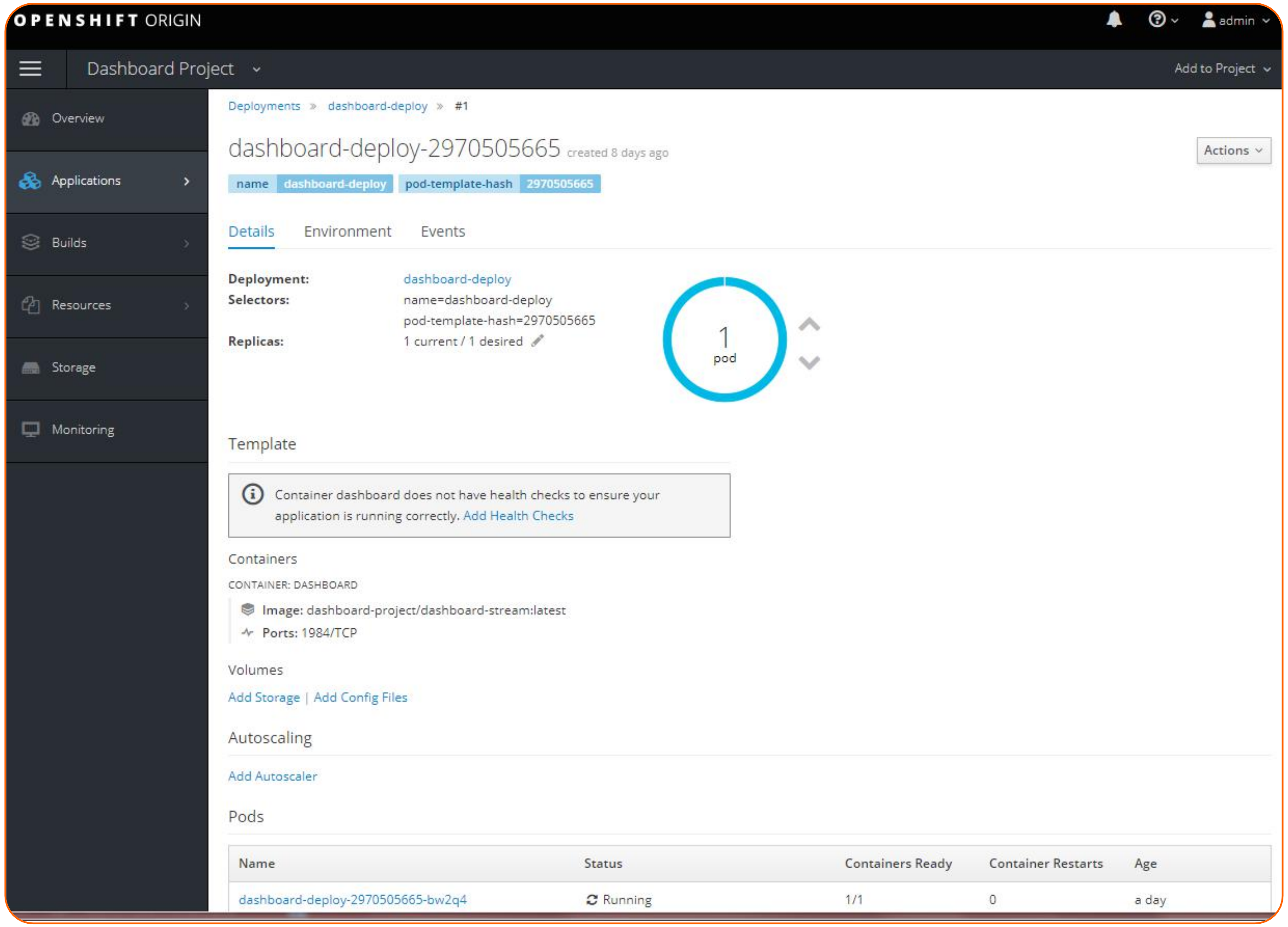
- Only submit and forget works in Docker
- spark-submit deploy-mode cluster master yarn
- kinit your keytab file for Kerberos
- create virtual env with conda and zip



```
xport SPARK_HOME="/usr/hdp/current/spark2-client/"
export PYTHONPATH=$SPARK_HOME/python/lib/py4j-*-src.zip:$PYTHONPATH:$SPARK_HOME/pyt
hon/
export LD_LIBRARY_PATH=/opt/rh/python27/root/usr/lib64:/oracle/product/11.2.0/clien
t/lib:/usr/hdp/current/hadoop-client/lib/native/
export SPARK_YARN_USER_ENV="PYTHONPATH=${PYTHONPATH},LD_LIBRARY_PATH=${LD_LIBRARY_P
ATH},PYTHONHASHSEED=0"

spark-submit  --executor-cores 1 --executor-memory 512m --driver-memory 512m --conf
 spark.yarn.queue=default \
--master yarn \
--deploy-mode cluster \
--name spark_test --conf "spark.app.id=spark_virtualenv_test" \
--conf spark.pyspark.virtualenv.enabled=true \
--conf spark.pyspark.virtualenv.type=native \
--conf spark.pyspark.virtualenv.bin.path=/opt/rh/python27/root/usr/bin/virtualenv
\
--conf spark.pyspark.virtualenv.requirements=requirements.txt \
--conf spark.pyspark.virtualenv.index_url=https://pypimirror.net/artifactory/api/py
pi/pypi_python_org/simple \
--conf spark.pyspark.python=/opt/rh/python27/root/usr/bin/python \
--conf "spark.executorEnv.PYTHONPATH==/opt/rh/python27/root/usr/bin${PATH:+:${PATH}
}}" \
--conf "spark.executorEnv.LD_LIBRARY_PATH=/opt/rh/python27/root/usr/lib64${LD_LIBRA
RY_PATH:+:${LD_LIBRARY_PATH}}:/usr/hdp/current/hadoop-client/lib/native/" \
spark_virtualenv.py
```

Openshift

PODS and JOBS



Deployments » dashboard-deploy » #1

dashboard-deploy-2970505665 created 8 days ago

name dashboard-deploy pod-template-hash 2970505665

Details Environment Events

Deployment: dashboard-deploy
Selectors: name=dashboard-deploy
pod-template-hash=2970505665
Replicas: 1 current / 1 desired

Template

Container dashboard does not have health checks to ensure your application is running correctly. Add Health Checks

Containers

CONTAINER: DASHBOARD

Image: dashboard-project/dashboard-stream:latest
Ports: 1984/TCP

Volumes

Add Storage | Add Config Files

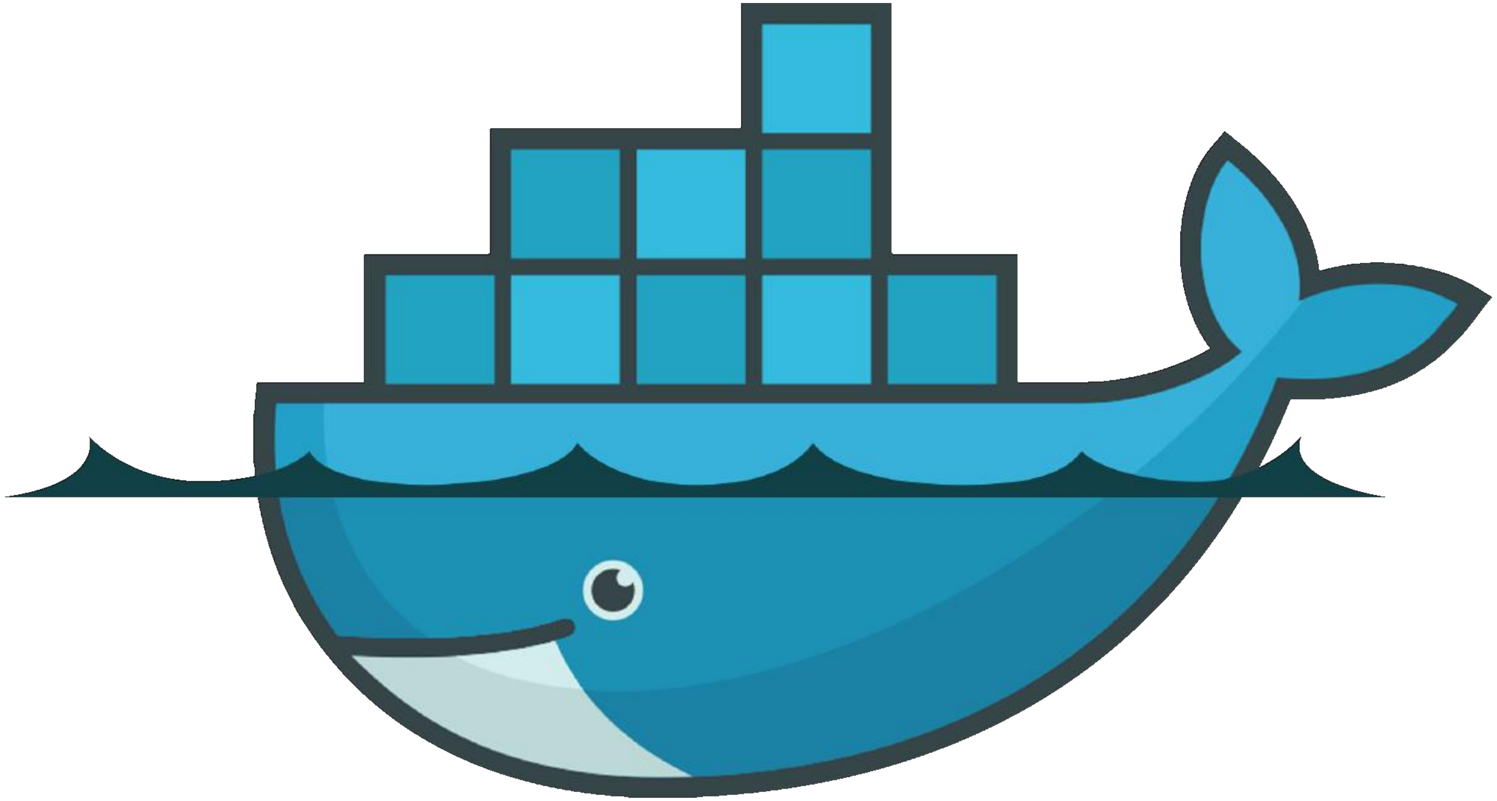
Autoscaling

Add Autoscaler

Pods

Name	Status	Containers Ready	Container Restarts	Age
dashboard-deploy-2970505665-bw2q4	Running	1/1	0	a day

Demo Time



considerations

- Jenkins instead of gitlab-runner
- How to use scheduler tool
- How to handle Kerberos files
- Add gpu nodes to openshift

Questions?