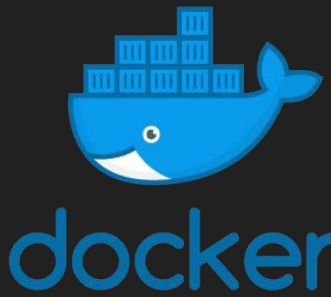


DOCKER AND DISTRIBUTED OPTIMIZATION

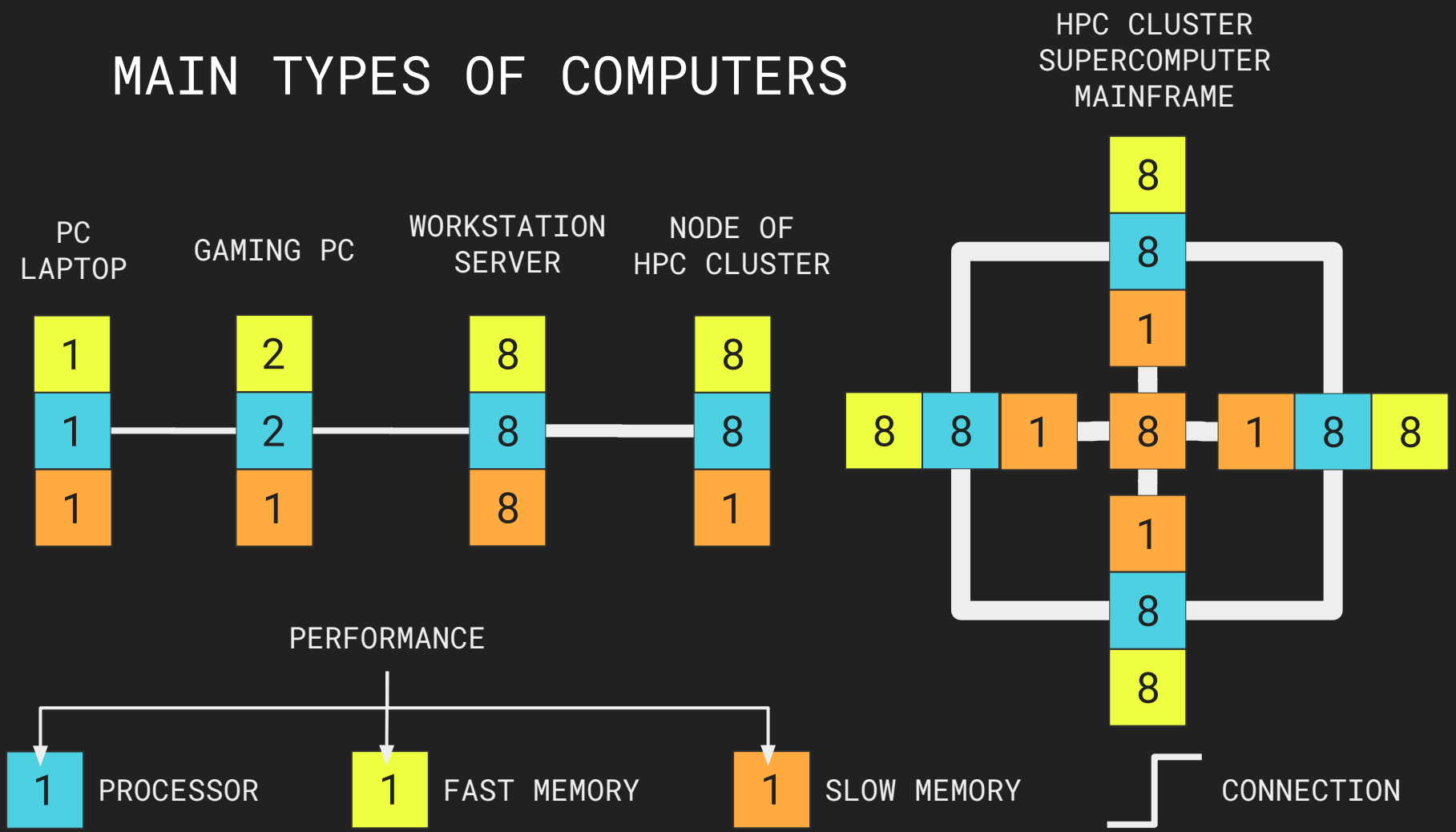
OPTUNA

PLAN

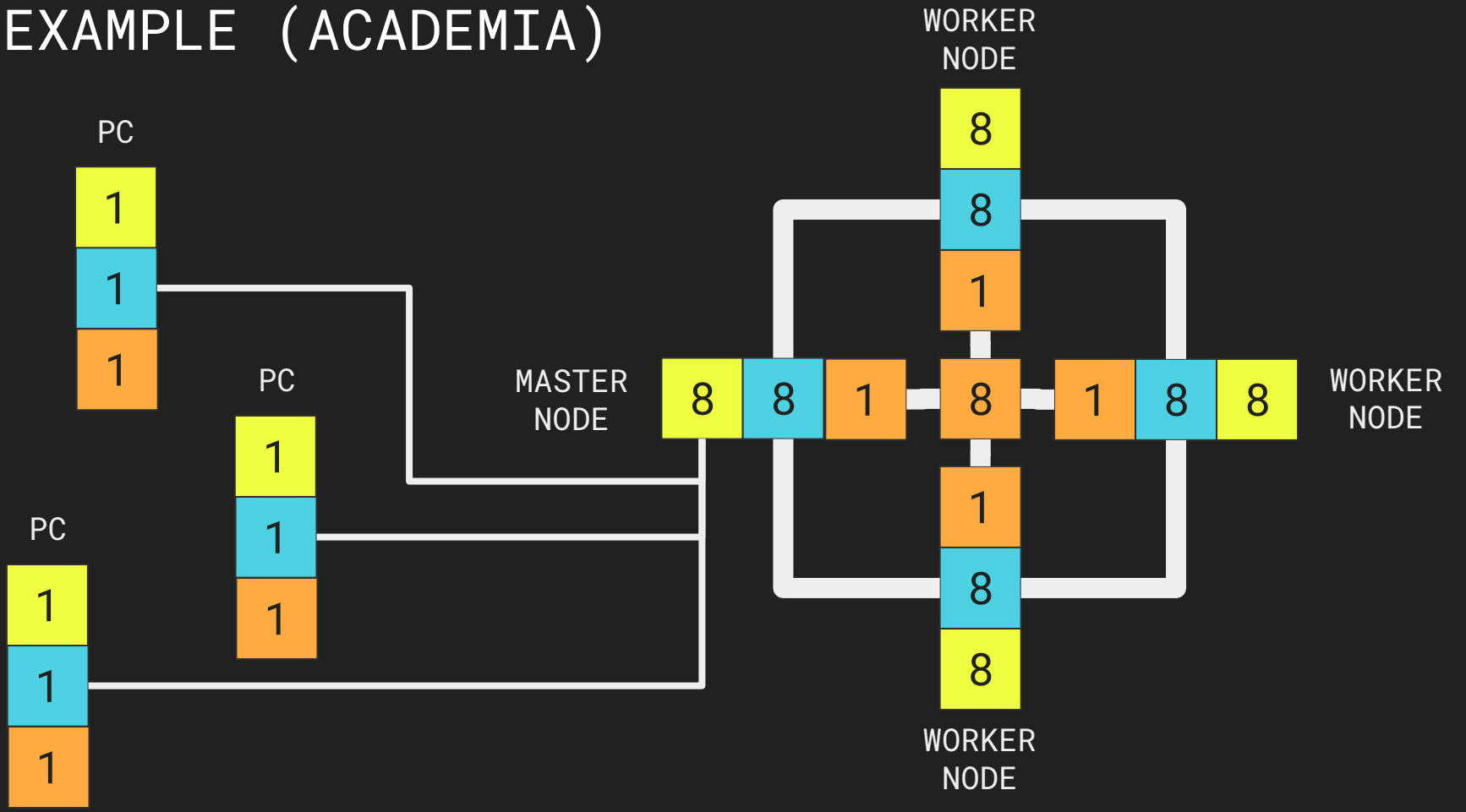
1. Computers
2. Virtualization
3. Docker
4. Orchestration (Docker Swarm)
5. Optimization
6. Hyperparameter optimization
7. Distributed hyperparameter optimization (Optuna)
8. Directed Acyclic Graph (DAG)
9. Use case
10. Useful links

1/10 COMPUTERS

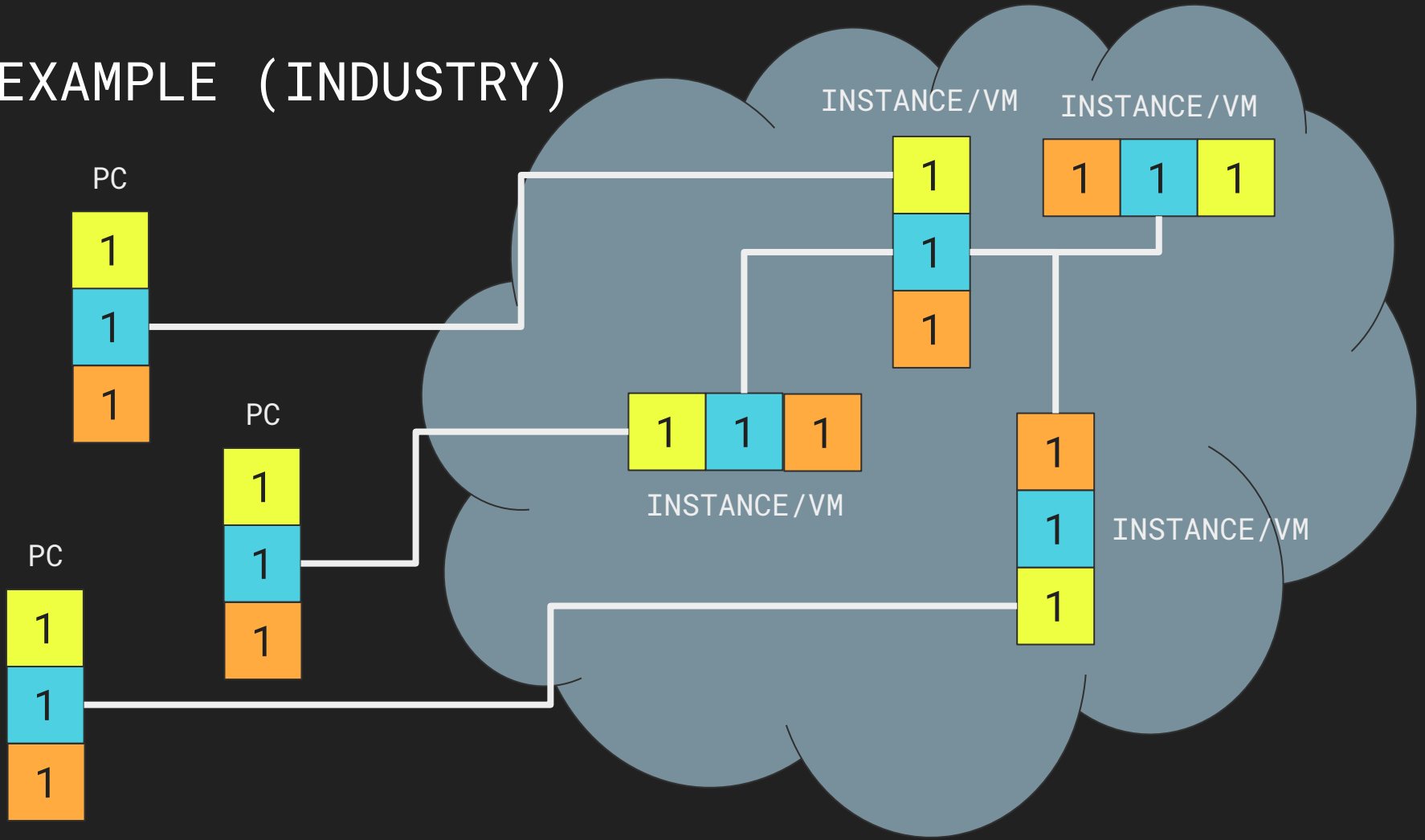
MAIN TYPES OF COMPUTERS



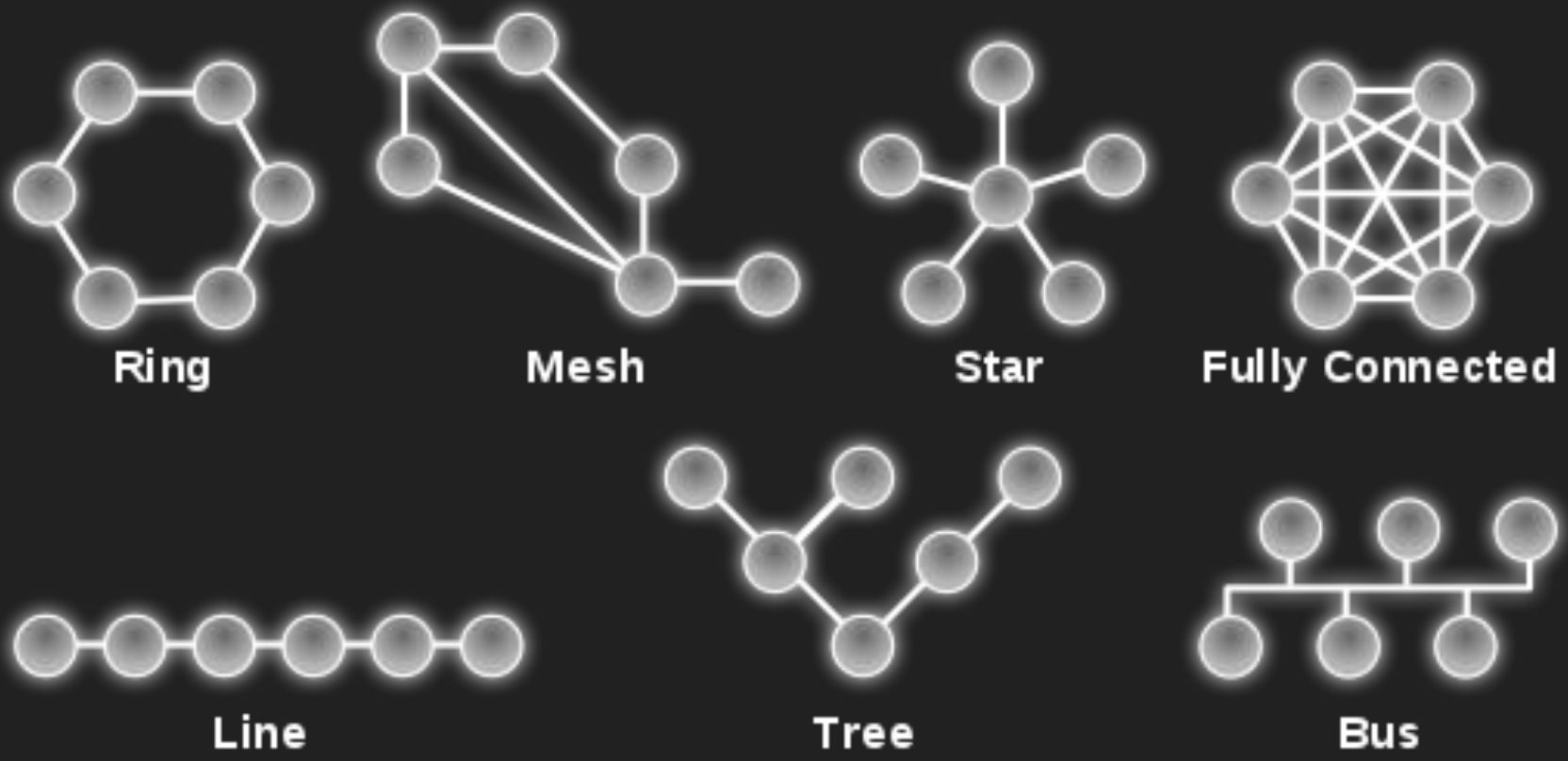
EXAMPLE (ACADEMIA)



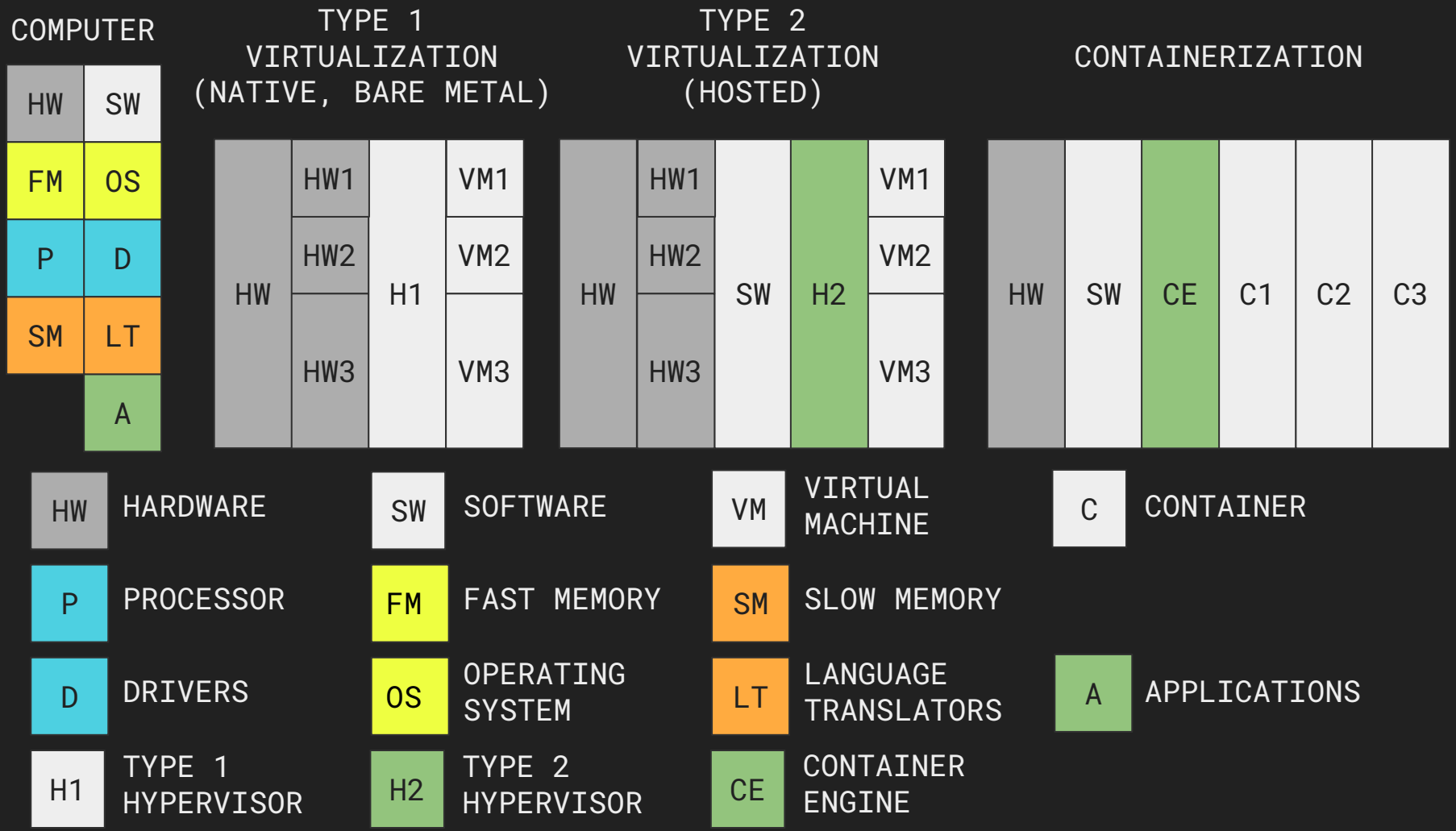
EXAMPLE (INDUSTRY)



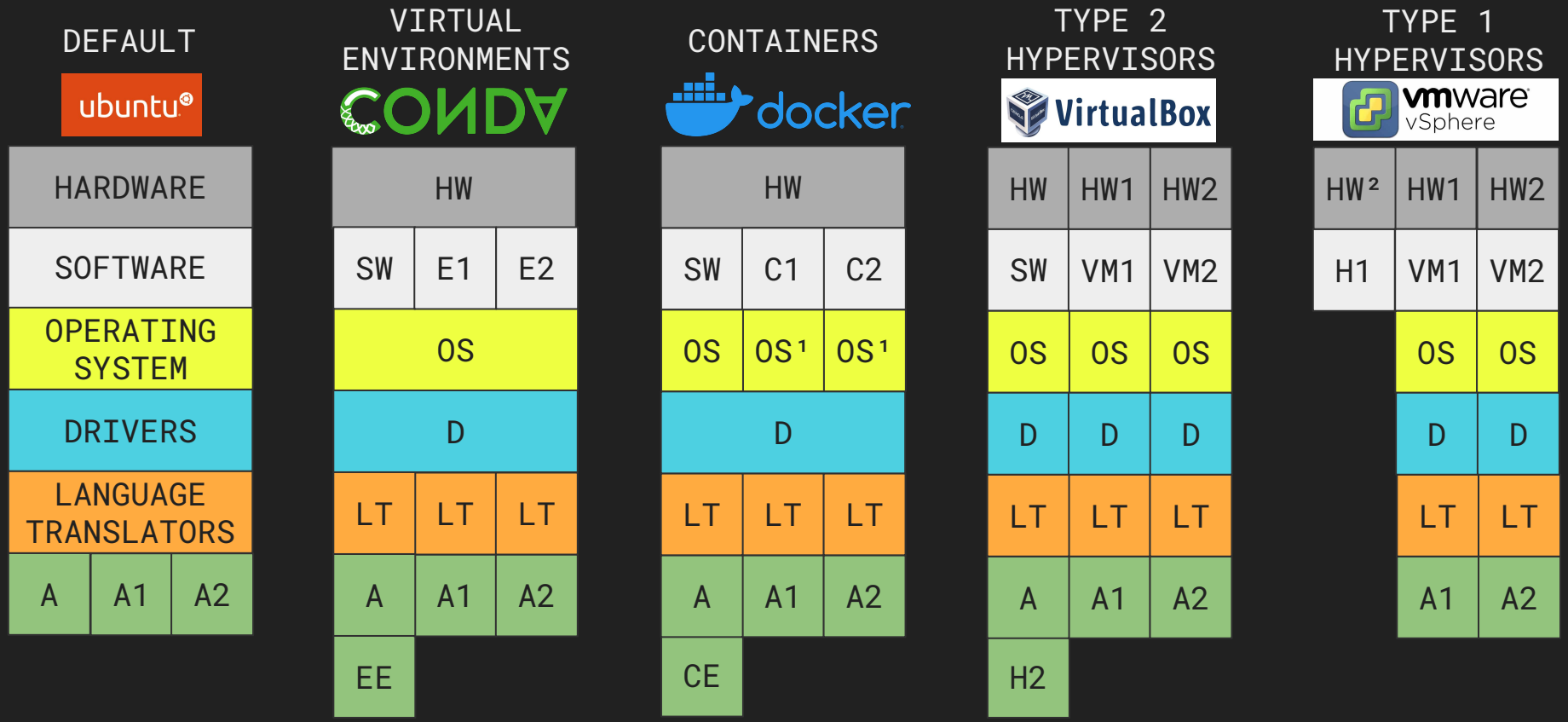
NETWORK TOPOLOGY EXAMPLES



2/10 VIRTUALIZATION


COMPUTER
HW
SW
FM
OS
P
D
SM
LT
A
TYPE 1 VIRTUALIZATION (NATIVE, BARE METAL)
HW
HW1
HW2
HW3
H1
VM1
VM2
VM3
TYPE 2 VIRTUALIZATION (HOSTED)
HW
HW1
HW2
HW3
SW
H2
VM1
VM2
VM3
CONTAINERIZATION
HW
SW
CE
C1
C2
C3
HW HARDWARE
SW SOFTWARE
VM VIRTUAL MACHINE
C CONTAINER
P PROCESSOR
FM FAST MEMORY
SM SLOW MEMORY
D DRIVERS
OS OPERATING SYSTEM
LT LANGUAGE TRANSLATORS
A APPLICATIONS
H1 TYPE 1 HYPERVISOR
H2 TYPE 2 HYPERVISOR
CE CONTAINER ENGINE

VIRTUAL ENVIRONMENTS VS CONTAINERS VS HYPERVISORS



COMPUTER

H HARDWARE

S SOFTWARE

P PROCESSOR

FM FAST MEMORY

SM SLOW MEMORY

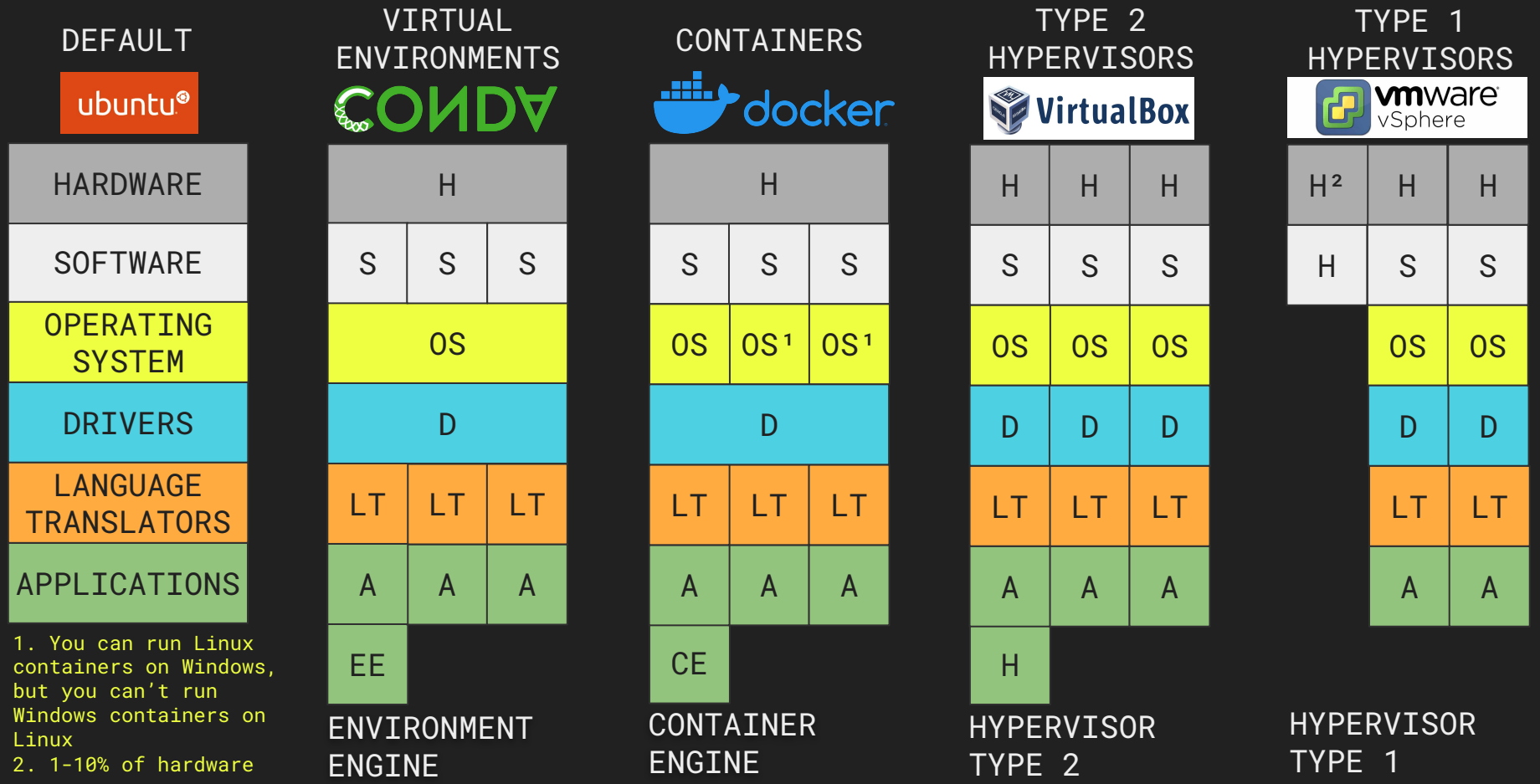
D DRIVERS

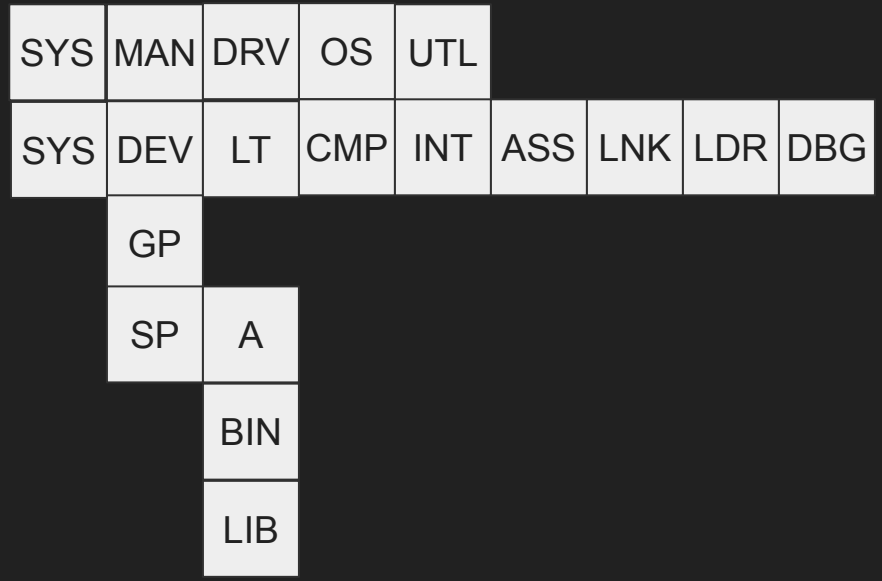
OS OPERATING SYSTEM

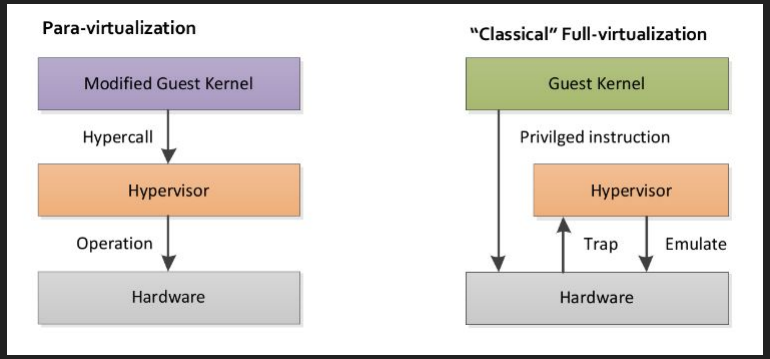
LT LANGUAGE TRANSLATORS

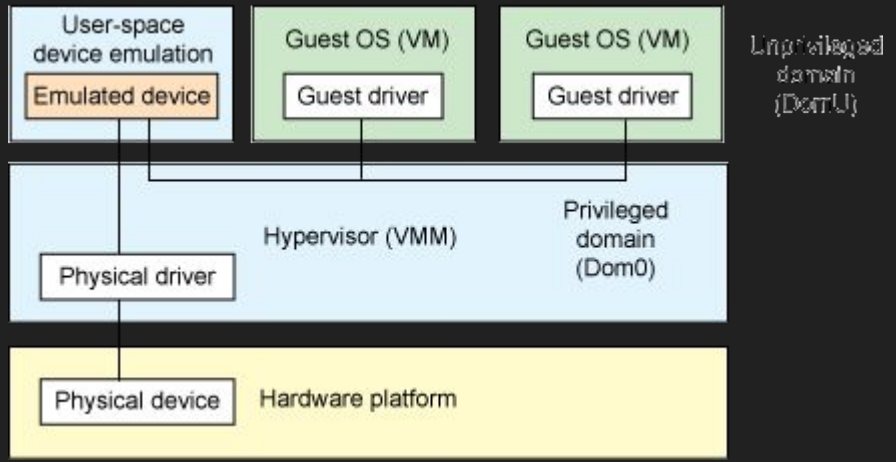
A APPLICATIONS

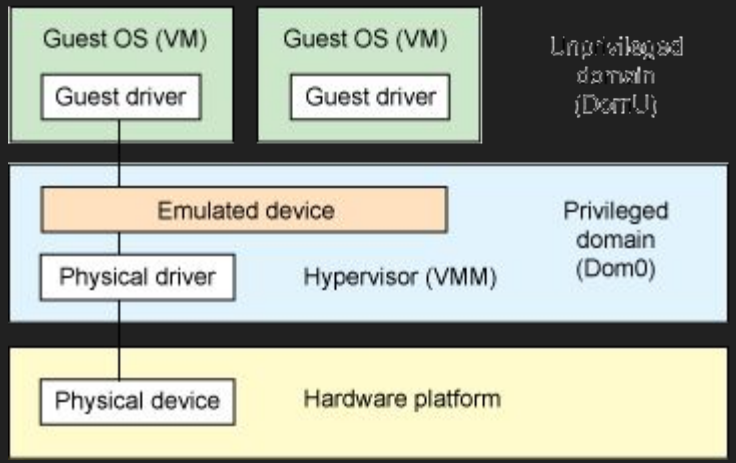
VIRTUAL ENVIRONMENTS VS CONTAINERS VS HYPERVISORS

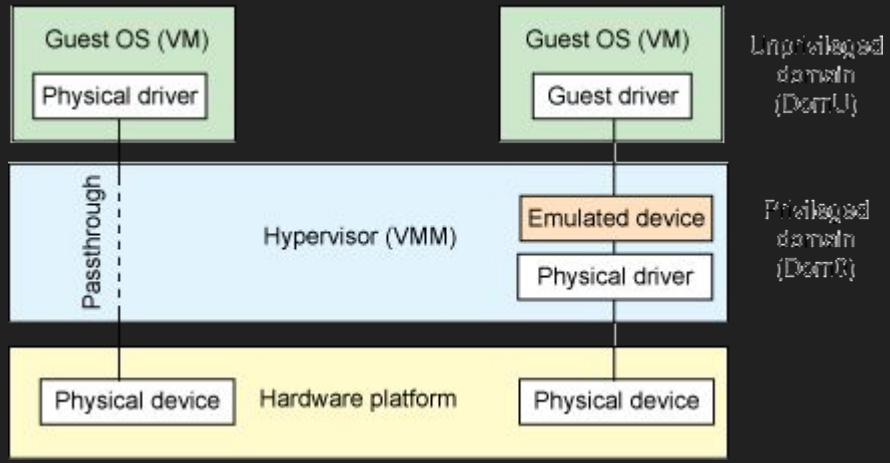


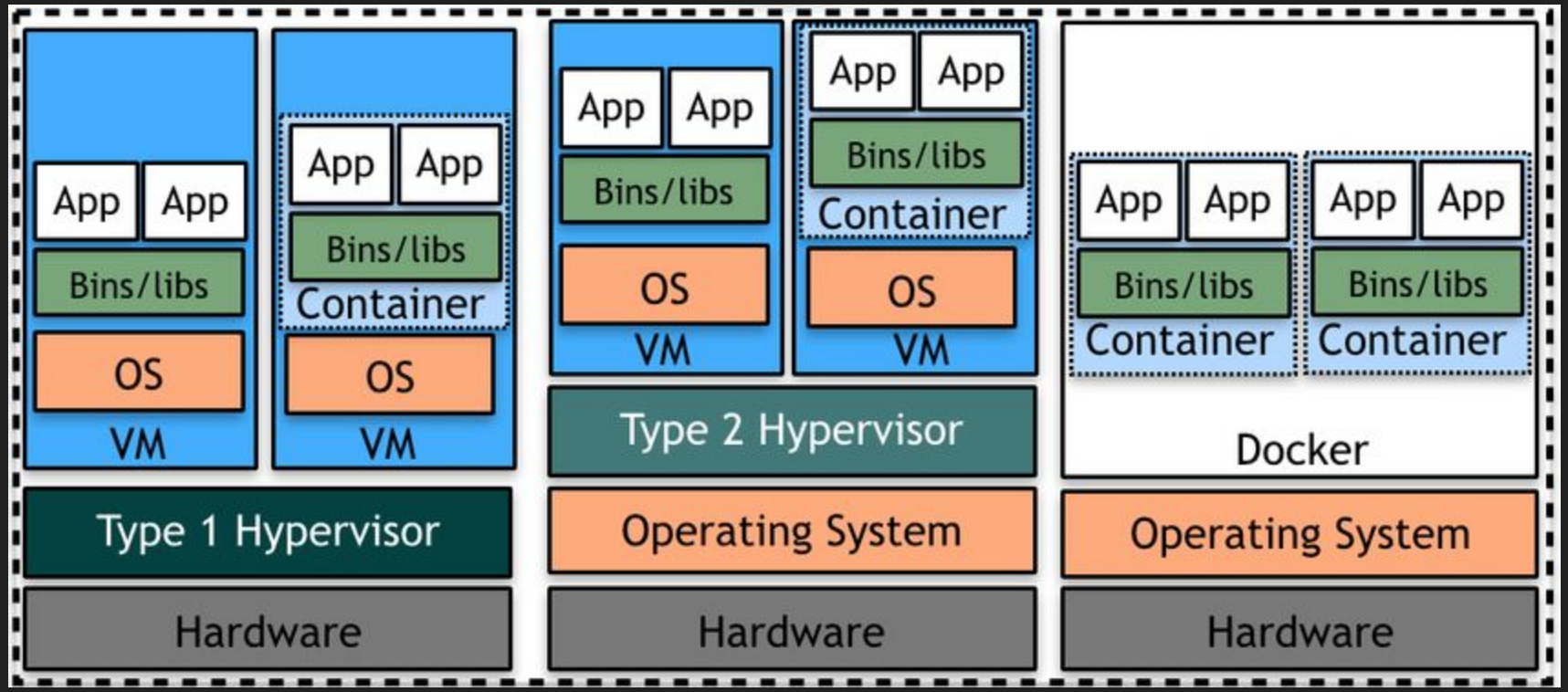

SYS
MAN
DRV
OS
UTL
SYS
DEV
LT
CMP
INT
ASS
LNK
LDR
DBG
GP
SP
A
BIN
LIB

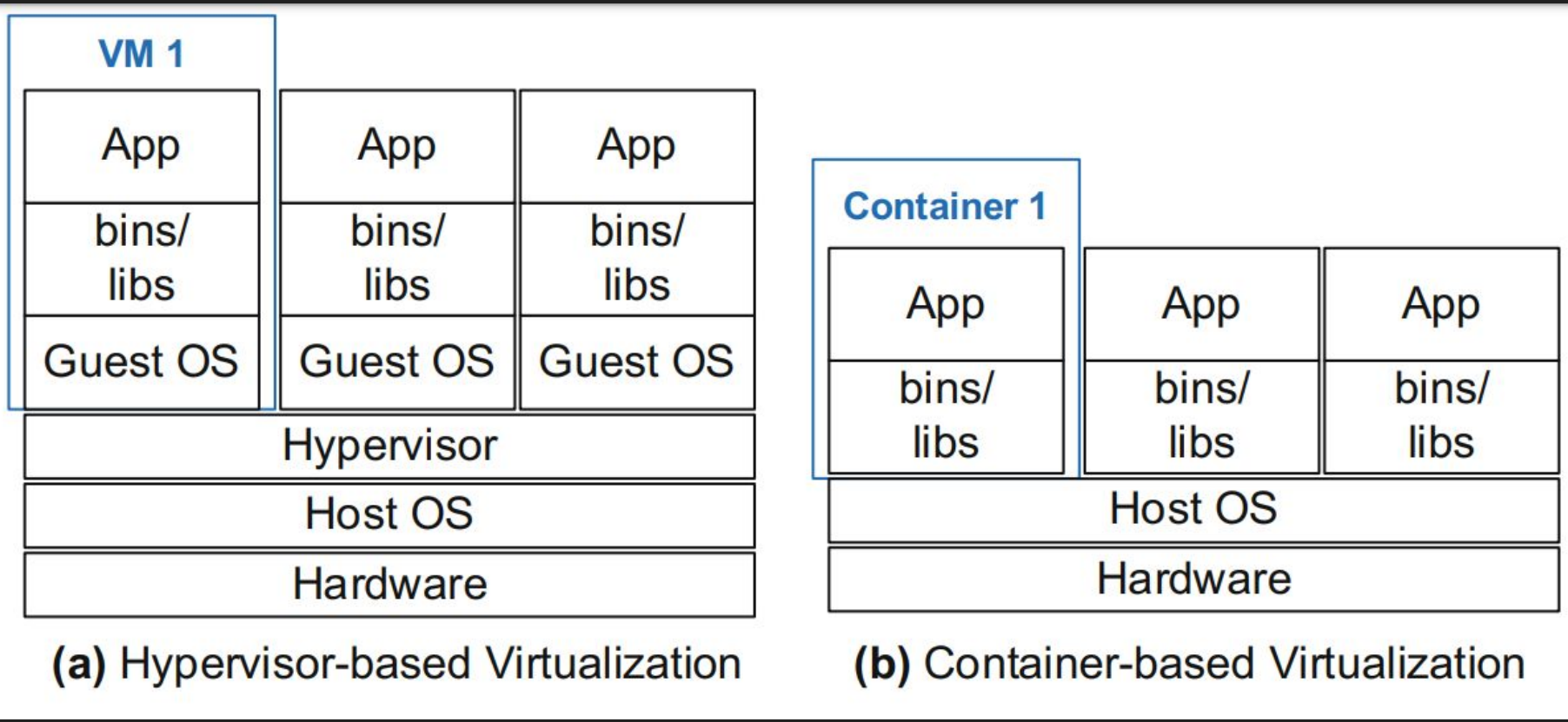

Para-virtualization
Modified Guest Kernel
Hypercall
Hypervisor
Operation
Hardware
"Classical" Full-virtualization
Guest Kernel
Privilged instruction
Hypervisor
Trap
Emulate
Hardware

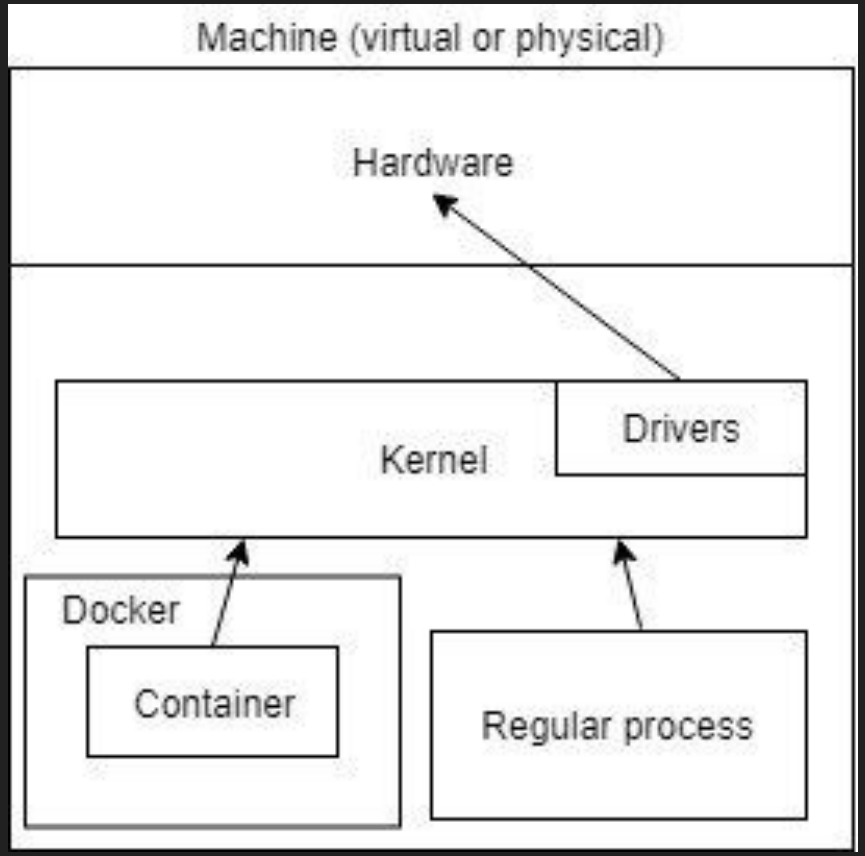

User-space device emulation
Emulated device
Guest OS (VM)
Guest driver
Guest OS (VM)
Guest driver
Unprivileged domain (DomU)
Physical driver
Hypervisor (VMM)
Privileged domain (Dom0)
Physical device
Hardware platform


Guest OS (VM)
Guest driver
Guest OS (VM)
Guest driver
Unprivileged domain (DomU)
Emulated device
Physical driver
Hypervisor (VMM)
Privileged domain (Dom0)
Physical device
Hardware platform


Guest OS (VM)
Physical driver
Guest OS (VM)
Guest driver
Unprivileged domain (DomU)
Passthrough
Hypervisor (VMM)
Emulated device
Physical driver
Privileged domain (Dom0)
Physical device
Hardware platform
Physical device


App
App
Bins/libs
OS
VM
App
App
Bins/libs
Container
OS
VM
Type 1 Hypervisor
Hardware
App
App
Bins/libs
OS
VM
App
App
Bins/libs
Container
OS
VM
Type 2 Hypervisor
Operating System
Hardware
App
App
Bins/libs
Container
App
App
Bins/libs
Container
Docker
Operating System
Hardware


VM 1
App
bins/
libs
Guest OS
App
bins/
libs
Guest OS
App
bins/
libs
Guest OS
Hypervisor
Host OS
Hardware
(a) Hypervisor-based Virtualization
Container 1
App
bins/
libs
App
bins/
libs
App
bins/
libs
Host OS
Hardware
(b) Container-based Virtualization


Machine (virtual or physical)
Hardware
Kernel
Drivers
Docker
Container
Regular process

3/10 DOCKER

docker

F I C

C
OS
D
LT
A

Dockerfile

```
FROM node:12-alpine
RUN apk add --no-cache python2 g++ make
WORKDIR /app
COPY ..
RUN yarn install --production
CMD ["node", "src/index.js"]
EXPOSE 3000
```

docker-compose.yml

```
version: "3.9"
services:
  redis:
    build: .
    ports:
      - "3000:3000"
    volumes:
      -
logvol:/var/log
volumes:
  logvol:
```

F Dockerfile

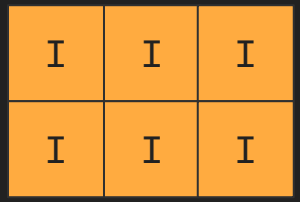
I DOCKER IMAGE

C docker-compose.yml

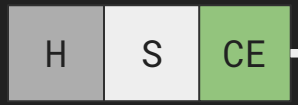
C DOCKER CONTAINER

4/10 ORCHESTRATION

EXAMPLE

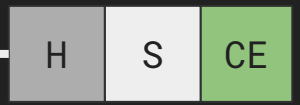


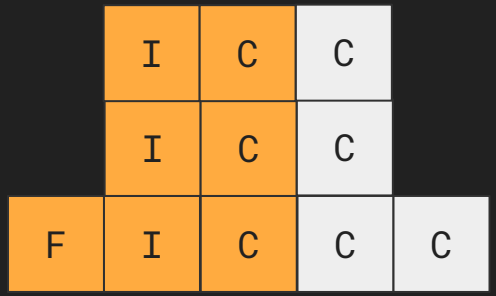
REGISTRY



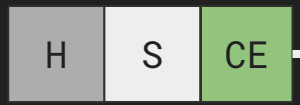
ORCHESTRATION

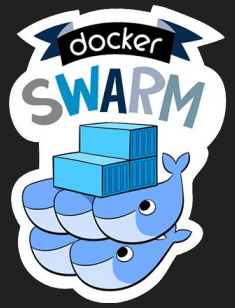
HOST WORKER





HOST WORKER



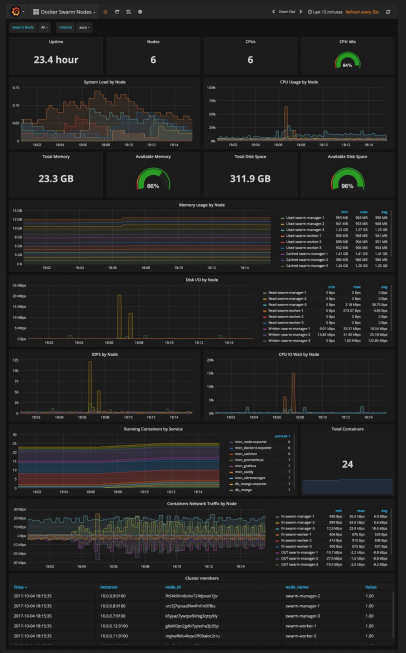


HOST MANAGER

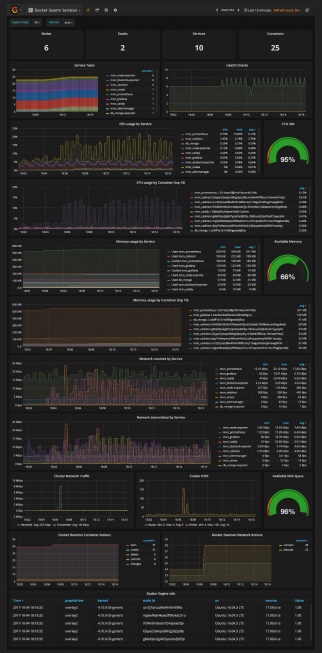
docker-swarm-visualizer

https://github.com/dockersamples/docker-swarm-visualizer

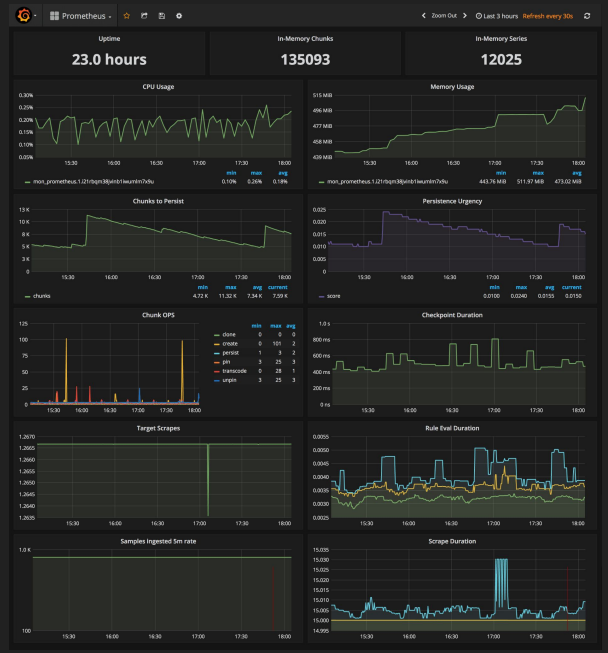
swarmprom



Nodes

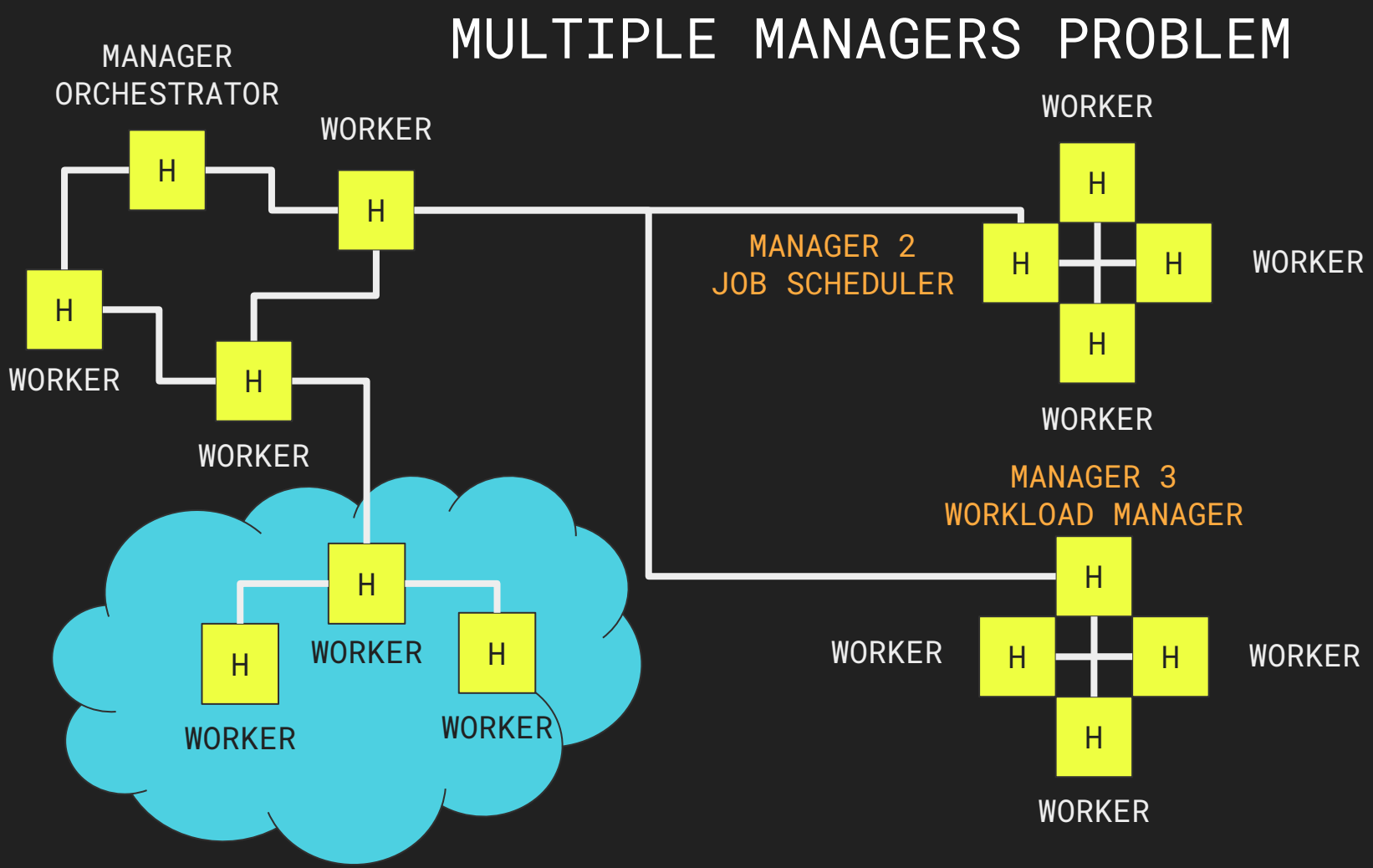


Services



Prometheus Stats

https://github.com/stefanprodan/swarmprom


MULTIPLE MANAGERS PROBLEM
MANAGER
ORCHESTRATOR
H
WORKER
H
H
WORKER
H
WORKER
MANAGER 2
JOB SCHEDULER
WORKER
H
H
H
H
WORKER
MANAGER 3
WORKLOAD MANAGER
H
WORKER
H
H
WORKER
H
WORKER
H
WORKER
H
WORKER
H
H
WORKER

CONTAINERS, ORCHESTRATORS AND JOB SCHEDULERS

1. Charliecloud
2. Docker
3. UDOCKER
4. Rootless Docker
5. Kubernetes Pods
6. Shifter
7. Singularity
8. ENROOT
9. Podman
10. Sarus

1. Kubernetes (k8s)
2. Openshift
3. Nomad
4. Docker Swarm
5. Rancher
6. Mesos
7. Google Container Engine (GKE)
8. Google Cloud Run
9. AWS Elastic Kubernetes Service (EKS)
10. Amazon EC2 Container Service (ECS)
11. AWS Fargate
12. Azure AKS Service
13. Azure Managed Openshift Service
14. Azure Container Instances
15. DigitalOcean Kubernetes Service
16. Red Hat OpenShift Online
17. Linode Kubernetes Engine

1. Apache Airflow
2. Cron
3. DIET
4. HTCondor
5. Maui Cluster Sched
6. Nextflow
7. OpenLava
8. OpenPBS
9. Oracle Grid E
10. Platform LSF
11. ProActive
12. Quartz
13. Slurm Workload Manager
14. Univa Grid Engine
15. VisualCron
16. Fuzzball
17. Luigi
18. Argo
19. Prefect
20. Comunda
21. Ray
22. Celery

CONTAINERS, ORCHESTRATORS AND JOB SCHEDULERS

1. Charliecloud
2. Docker
3. UDOCKER
4. Rootless Docker
5. Kubernetes Pods
6. Shifter
7. Singularity
8. ENROOT
9. Podman
10. Sarus

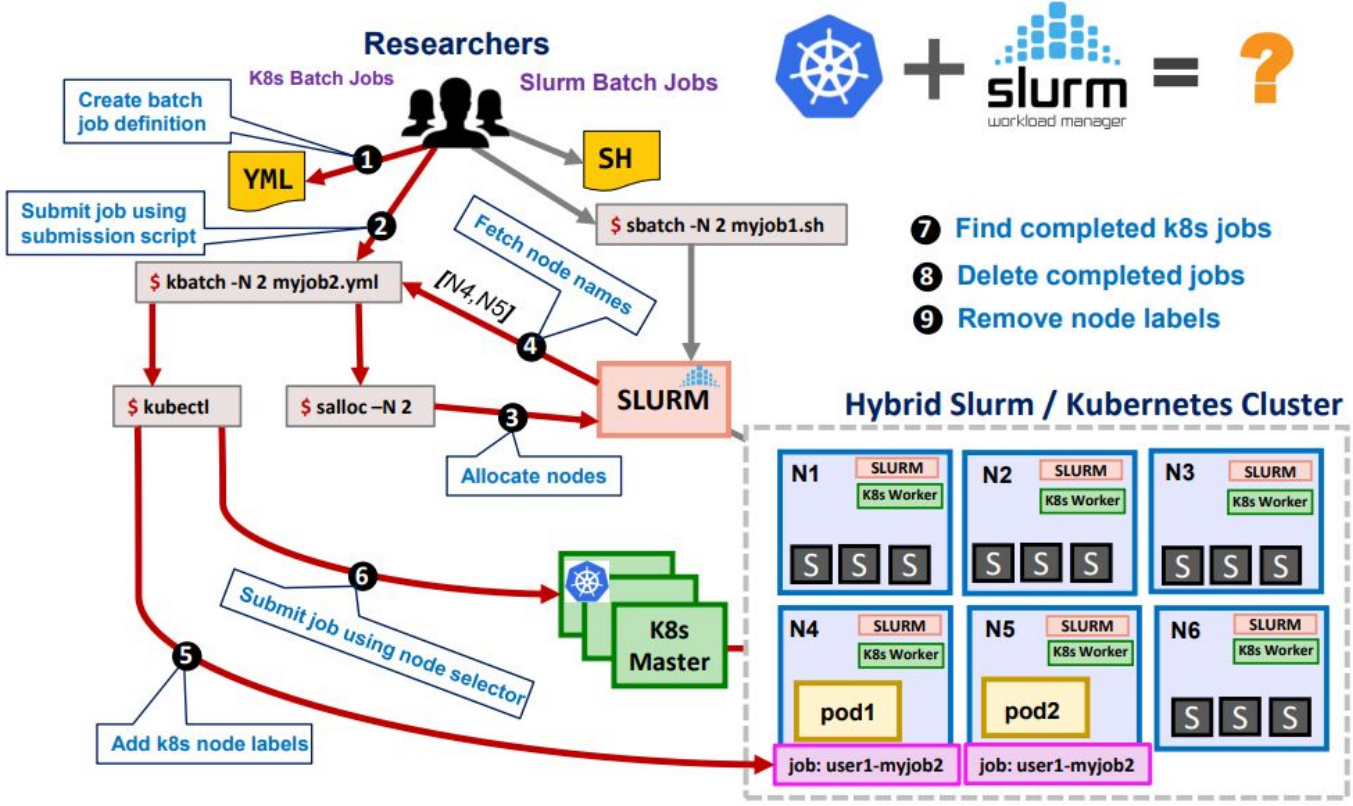
1. Kubernetes (k8s)
2. Openshift
3. Nomad
4. Docker Swarm
5. Rancher
6. Mesos
7. Google Container Engine (GKE)
8. Google Cloud Run
9. AWS Elastic Kubernetes Service (EKS)
10. Amazon EC2 Container Service (ECS)
11. AWS Fargate
12. Azure AKS Service
13. Azure Managed Openshift Service
14. Azure Container Instances
15. DigitalOcean Kubernetes Service
16. Red Hat OpenShift Online
17. Linode Kubernetes Engine

1. Apache Airflow
2. Cron
3. DIET
4. HTCondor
5. Maui Cluster Sched
6. Nextflow
7. OpenLava
8. OpenPBS
9. Oracle Grid E
10. Platform LSF
11. ProActive
12. Quartz
13. Slurm Workload Manager
14. Univa Grid Engine
15. VisualCron
16. Fuzzball
17. Luigi
18. Argo
19. Prefect
20. Comunda
21. Ray
22. Celery

FRED HUTCH

Traditional HPC and Kubernetes Co-Existence

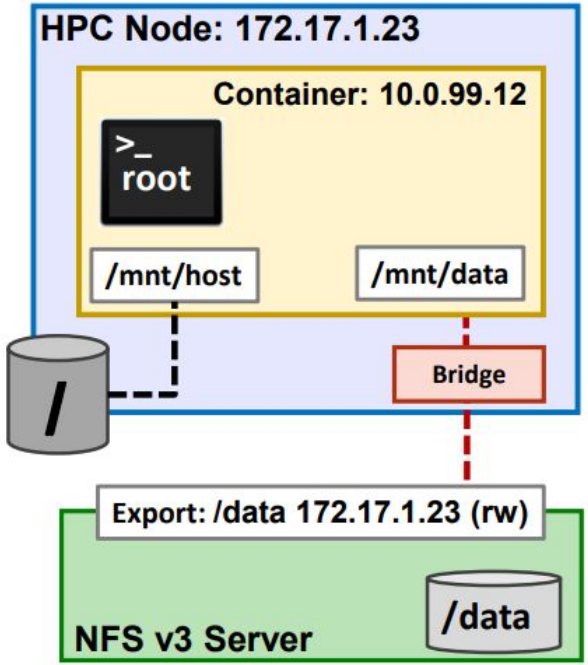
- **Goal**: Allow Slurm and K8s batch jobs to share the same cluster hardware



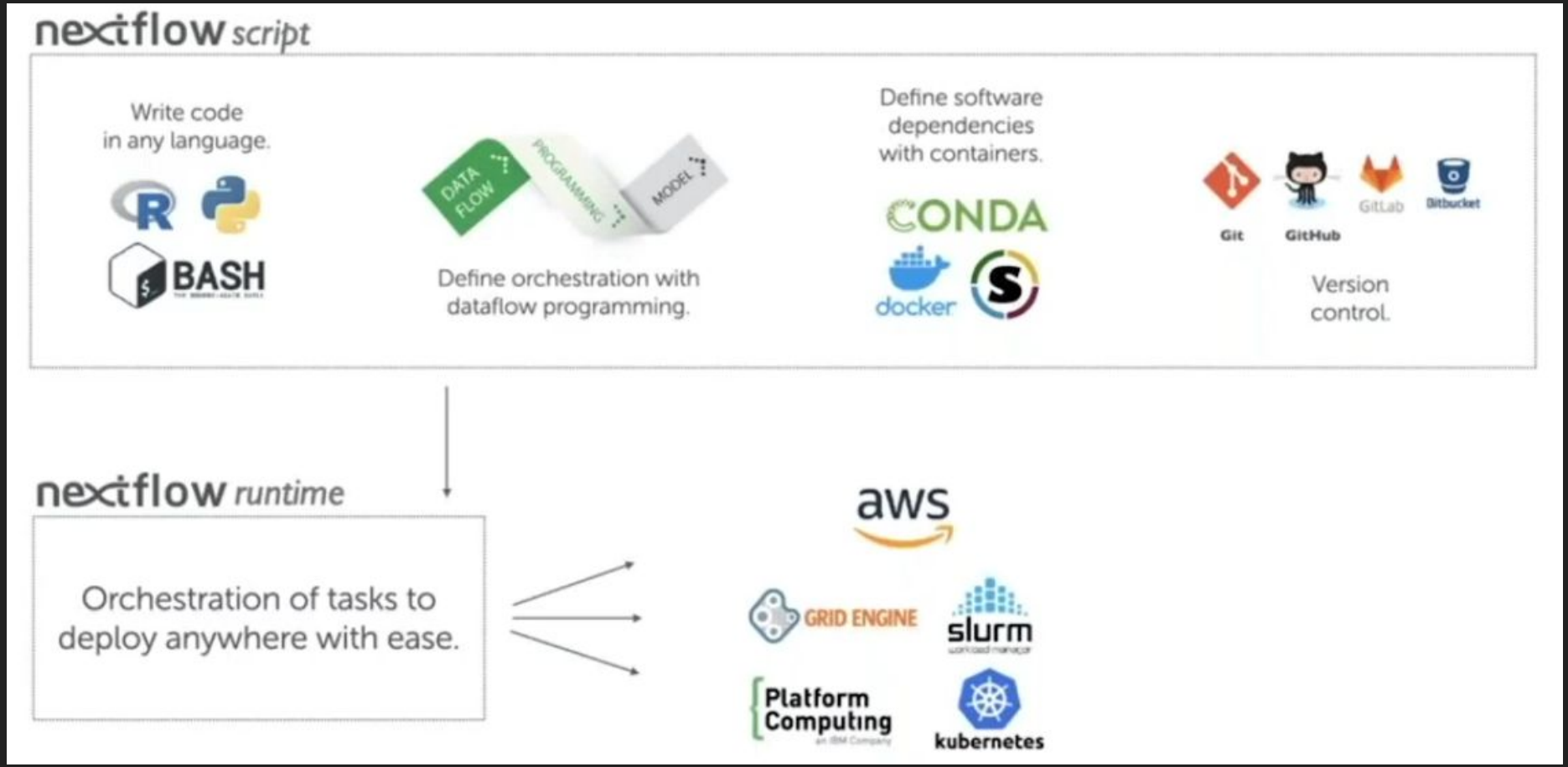
FRED HUTCH

Traditional HPC and Kubernetes Co-Existence

- **Is it possible?** Yes, but I wouldn't recommend it on a shared HPC cluster
- Allowing users to run docker containers on a multi-user host with shared file systems and NFSv3 exports secured only via AUTH_SYS is **risky**
- You can use pod security policies to mitigate some of this risk
- The only safe way to give users Docker like functionality on a shared host is with Singularity: https://www.sylabs.io/singularity/



Nextflow?


nextflow script
Write code in any language.
R
BASH
DATA FLOW
PROGRAMMING
MODEL
Define orchestration with dataflow programming.
Define software dependencies with containers.
CONDA
docker
Git
GitHub
GitLab
Bitbucket
Version control.
nextflow runtime
Orchestration of tasks to deploy anywhere with ease.
aws
GRID ENGINE
slurm
Platform Computing
kubernetes

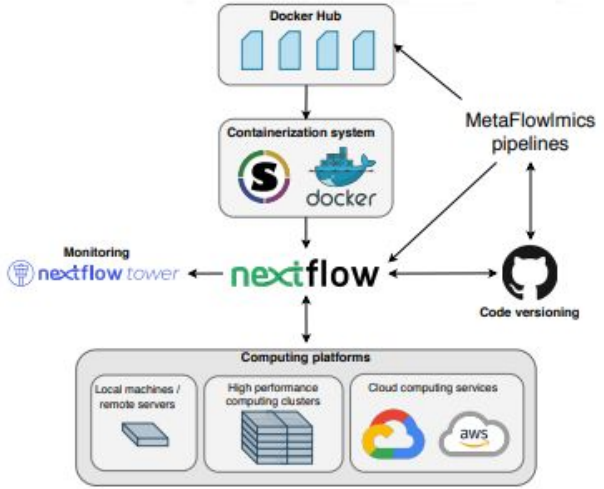


Figure 2: MetaFlow|mics workflow: MetaFlow|mics source code is hosted on GitHub, and required containers were built and uploaded on Docker Hub. Nextflow builds an abstraction layer on any configured platforms, and can run any of the pipeline using the provided docker images. Job monitoring is then easily achieved using Nextflow Tower.

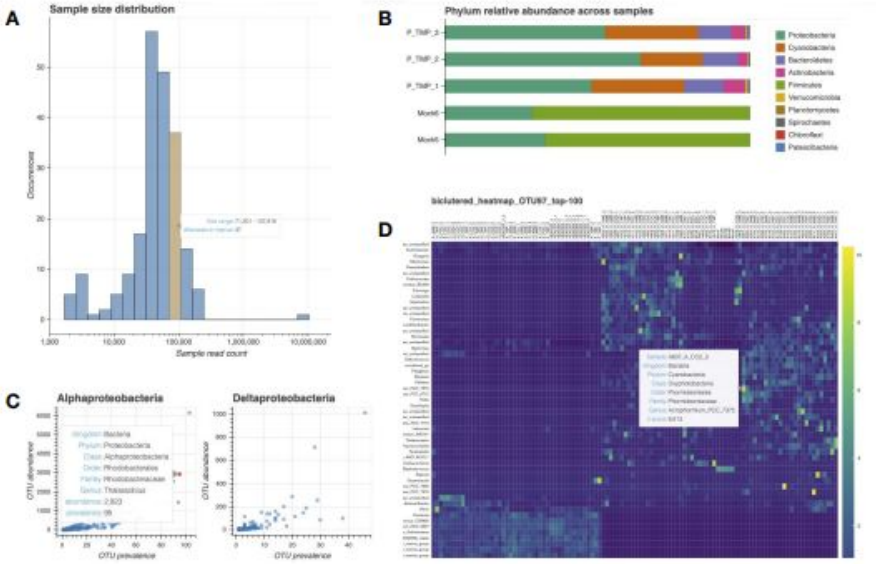


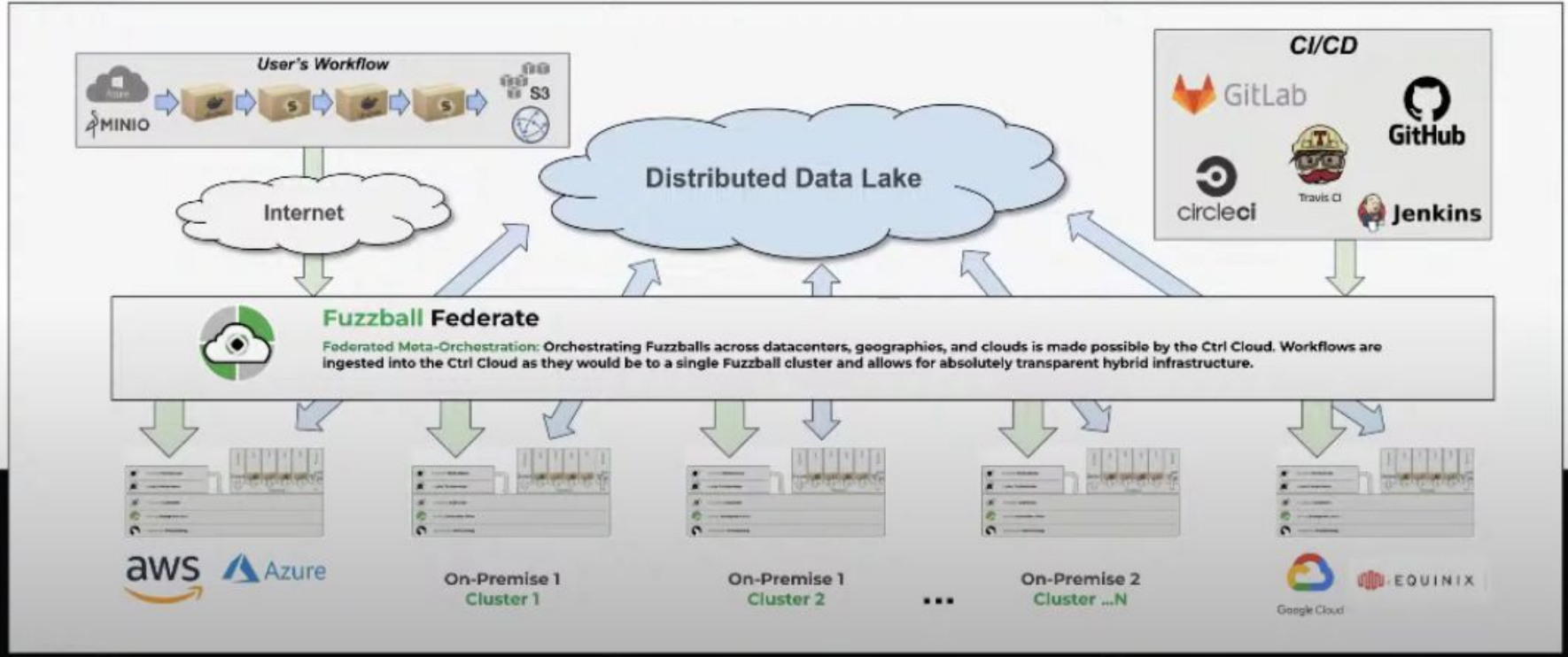
Figure 3: Pipelines summary figures. Each figure is interactive and displays additional information about the data points. A: Log sample size distribution. B: Stacked barplot of phyla proportions in each sample. C: Scatter plot of OTU prevalence vs abundance for the Alphaproteobacteria and Deltaproteobacteria classes. D: Heatmap of the top 100 OTU by samples. Both rows and columns are clustered (hierarchical bi-clustering) to identify patterns.

Fuzzball: HPC 2.0?

FUZZBALL

Federate

Fuzzball Federate allows us to **scale out** and **unite** Fuzzball clusters at massive scale into a **single seamless platform** for users and CI/CD to interact with over **geographically distributed** sites and clouds.



Executive Summary

- Container virtualization can solve many HPC problems
- Wide and growing popularity of container runtimes
  - OCI effort helped a lot! Especially runc
- Not all containers are created equal, isolation, …
- For HPC the race is still running
  - But Singularity has a significant headstart
  - How about *run**hp**c from Docker?*
  - *What do YOU and your users really need?*
- Scheduling HPC workloads beyond traditional workload managers „tricky"
  - But everybody wants Kubernetes….

Containers TODO

- Rootless Docker
- Kubernetes
- Nextflow
- Singularity
- New HPC cluster seamless integration

5/10 OPTIMIZATION

OPTIMIZATION PROBLEM

X

set of [random] variables (arguments, features, parameters)
e.g. {continuous, discrete, categorical, discrete, continuous, …}

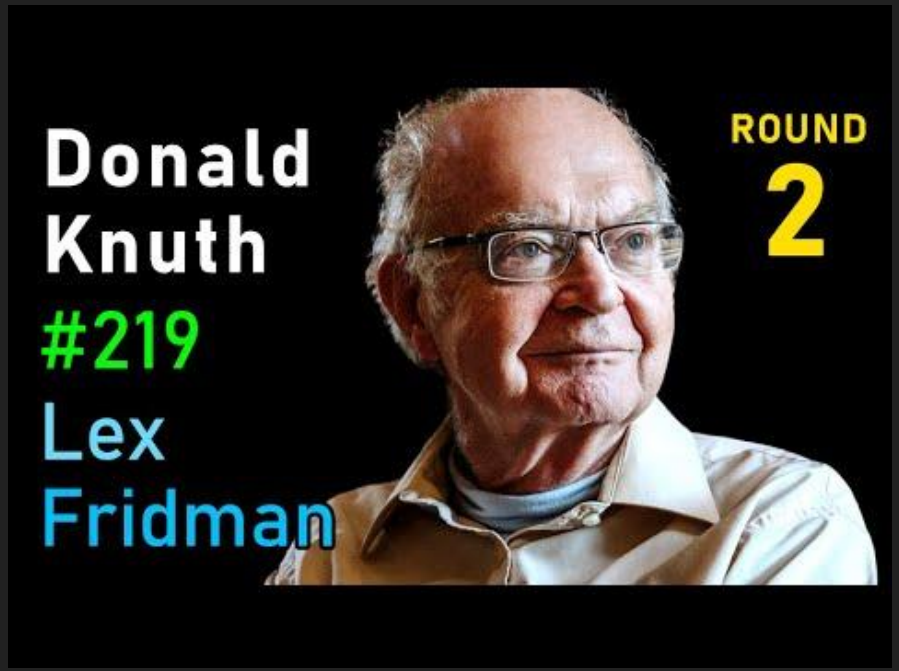
objective(x)

objective (cost, loss, error) function
where x is a sample from X
e.g for x equals to {3.14, 1, apple, -1, 1.62, …} the objective is 0.5

argmin[objective(x)]

optimization problem
i.e. such x that objective(x) is the minimum objective for all x from X
e.g. for x equals to {-2.72, 42, banana, 0, 0.0, …} the objective is -100

WHAT IS OPTIMIZATION?

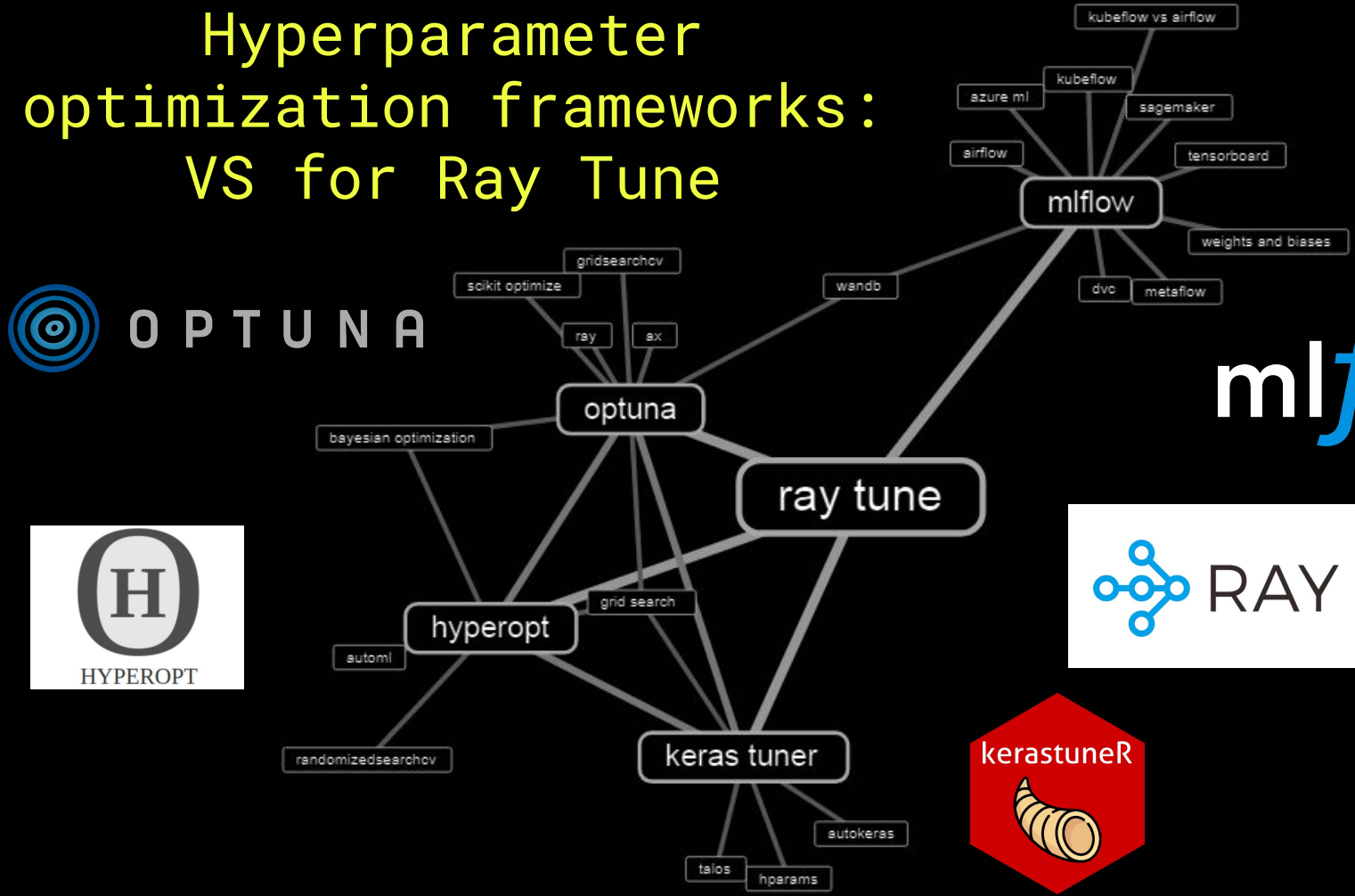


Optimize - to view with optimism!

Webster's Third New International Dictionary of the English Language, Unabridged, 1960s

6/10 HYPERPARAMETER OPTIMIZATION

Hyperparameter optimization frameworks: VS for Ray Tune



RAY tune

Random search/grid search	Random search/grid search
AxSearch	Bayesian/Bandit Optimization
BlendSearch	Blended Search
CFO	Cost-Frugal hyperparameter Optimization
DragonflySearch	Scalable Bayesian Optimization
SkoptSearch	Bayesian Optimization
HyperOptSearch	Tree-Parzen Estimators
BayesOptSearch	Bayesian Optimization
TuneBOHB	Bayesian Opt/HyperBand
NevergradSearch	Gradient-free Optimization
OptunaSearch	Optuna search algorithms
ZOOptSearch	Zeroth-order Optimization
SigOptSearch	Closed source
HEBOSearch	Heteroscedastic Evolutionary Bayesian Optimization

OPTUNA

Samplers

- Tree-structured Parzen Estimator
- CMA-ES based algorithm
- Grid Search
- Random Search

Multi-objective samplers

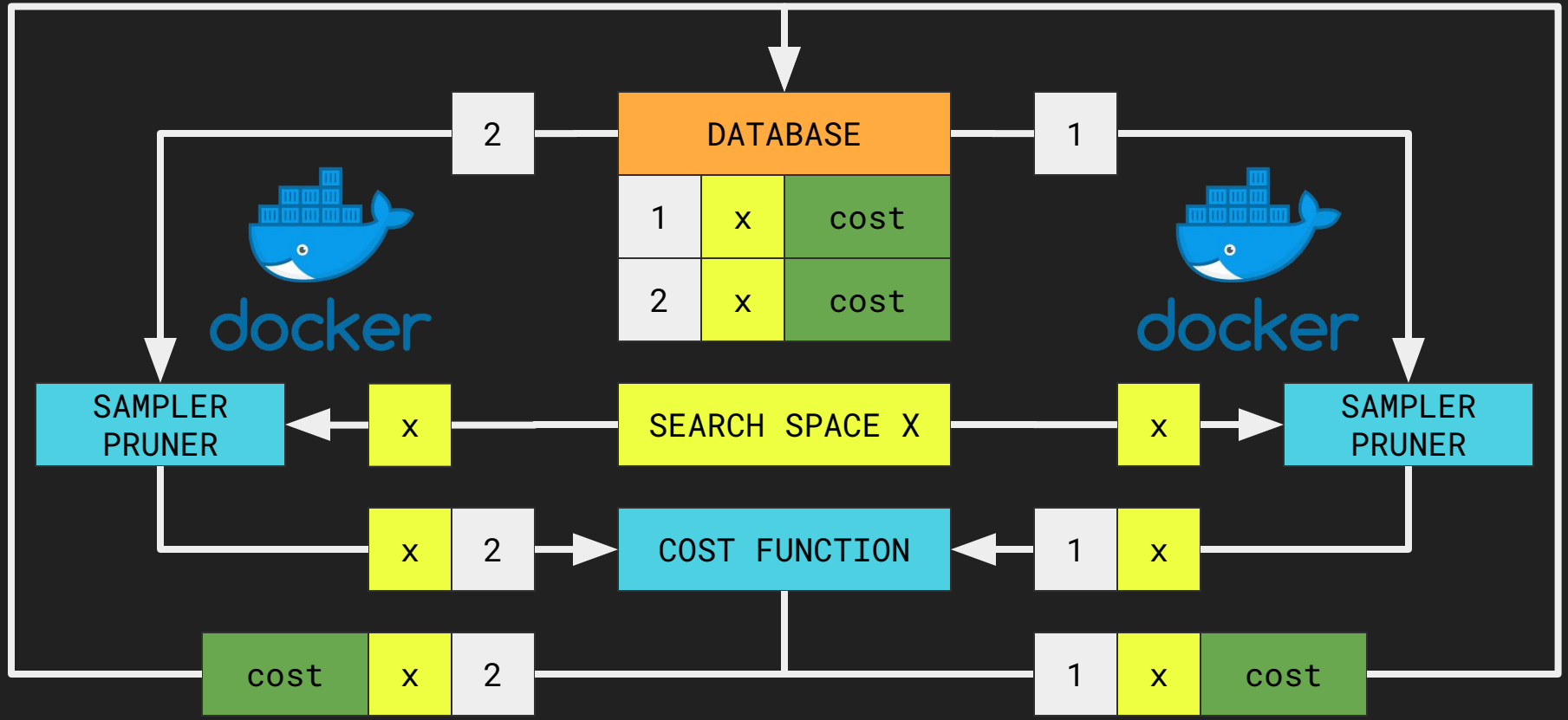
- NSGA-II
- MOTPE

Pruners

- Median Pruner
- Patient
- Percentile
- Successive Halving
- Hyperband
- Threshold

7/10 DISTRIBUTED HYPERPARAMETER OPTIMIZATION

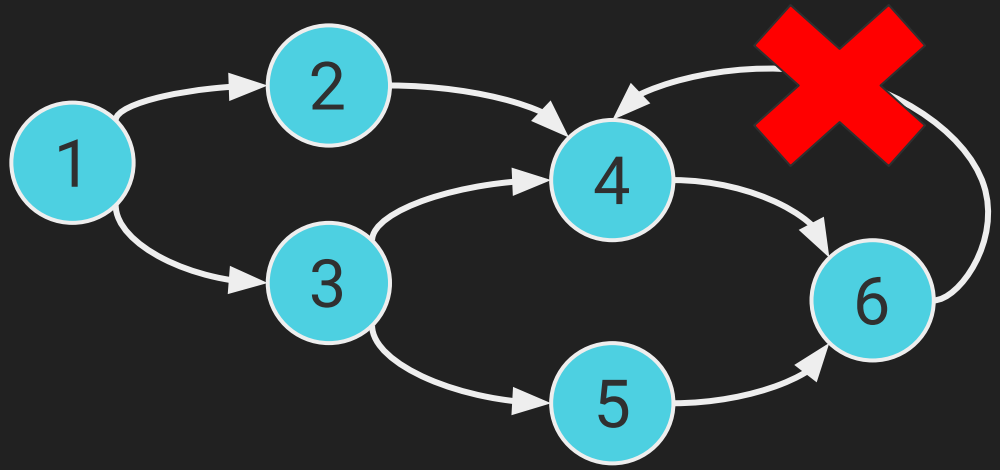
DISTRIBUTED OPTIMIZATION



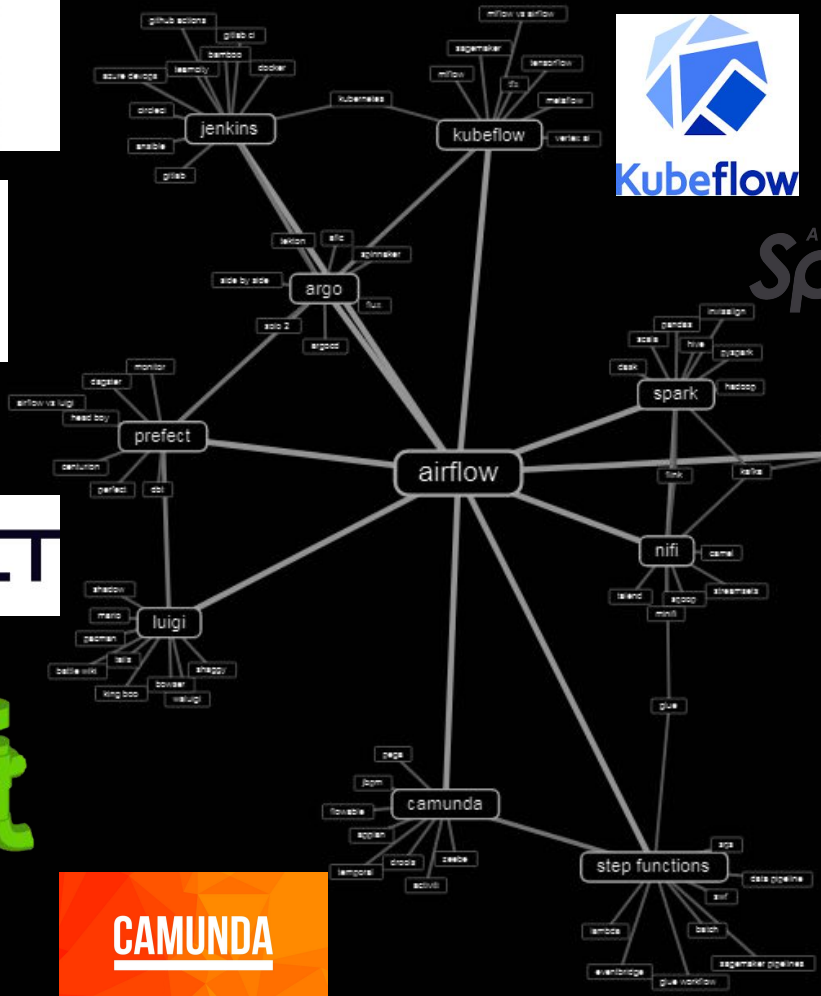
https://optuna.readthedocs.io/en/stable/tutorial/10_key_features/004_distributed.html

8/10 DIRECTED ACYCLIC GRAPH (DAG)

EXAMPLE



DAG makers: VS for Airflow



Jenkins

Kubeflow

argo

APACHE Spark

CELERY

PREFECT

APACHE nifi

Apache Airflow

RAY

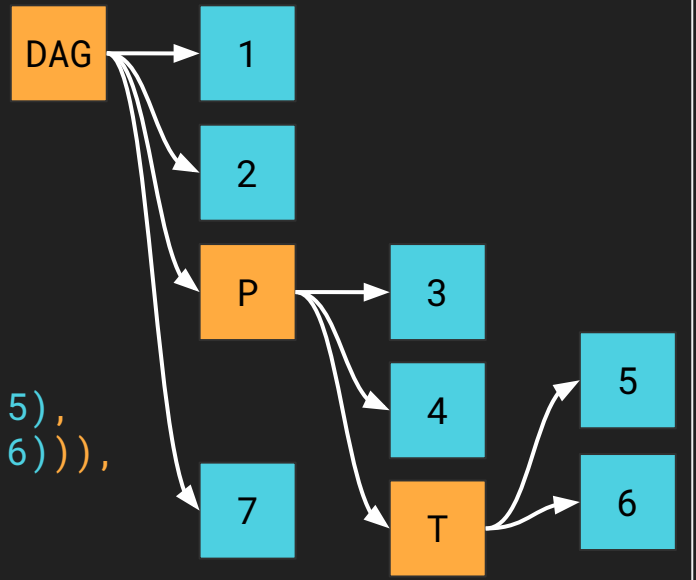
Luigi

CAMUNDA

AWS Step Functions

PYTHON DAG (TREE*) WITH CONCURRENT.FUTURES PSEUDOCODE

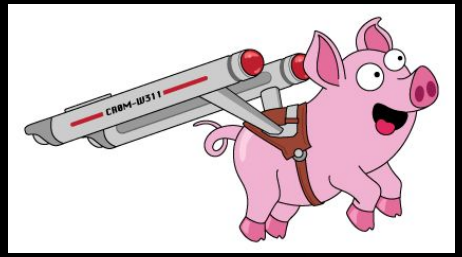
```
from concurrent.futures import ThreadPoolExecutor, ProcessPoolExecutor

thread = ThreadPoolExecutor()
parallel = ProcessPoolExecutor()
sequence = SequentialExecutor() # “for loop”

dag = sequence(
                lambda: print(1),
                lambda: print(2),
                parallel(
                        lambda: print(3),
                        lambda: print(4),
                        thread(
                                lambda: print(5),
                                lambda: print(6))),
                lambda: print(7))
dag()
```



*Actually it’s possible to create only trees (one parent per node) but not general dags (multiple parents) with this approach. A DAG has no directed cycles. A tree has no cycles at all.

https://github.com/PC-Trip/runner

DAG makers: VS for Nextflow



snakemake
A framework for reproducible data analysis

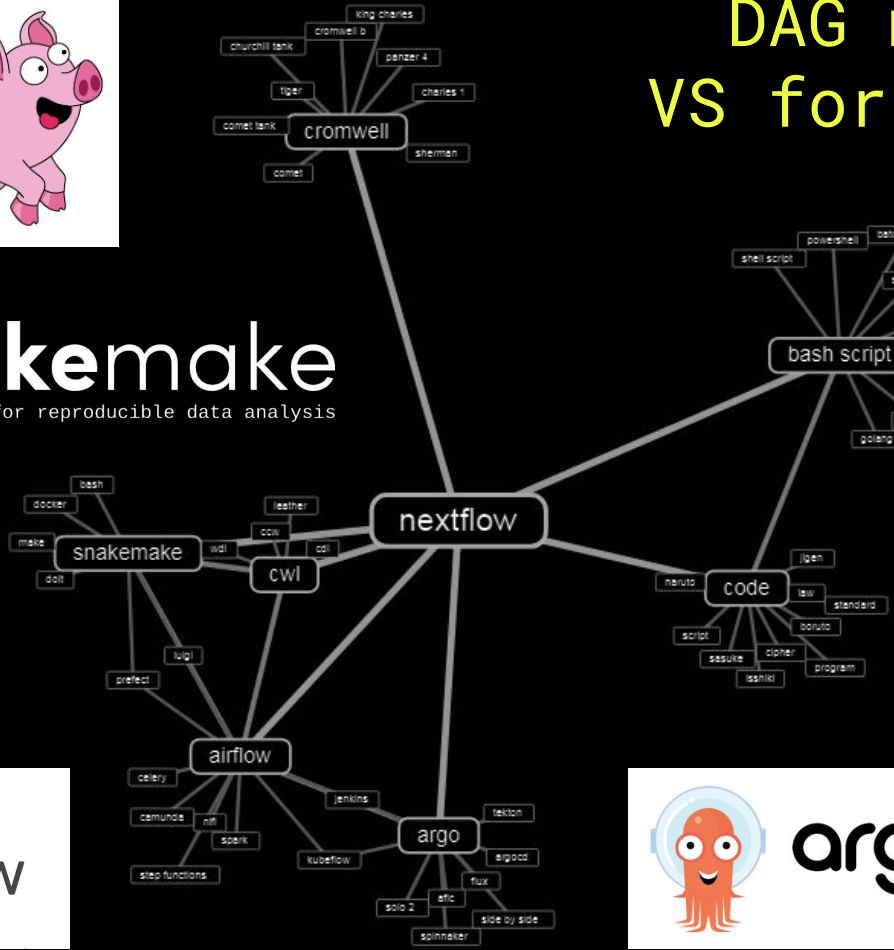
nextflow

COMMON WORKFLOW LANGUAGE

BASH
THE BOURNE-AGAIN SHELL

Apache Airflow

argo



9/10 USE CASE

OUR EXAMPLE

OPTIMIZE

MESHER

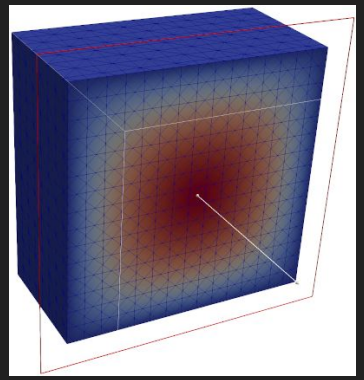
PARAMETERS

RUN

RESULTS

CONSTRAINTS

OBJECTIVES



SOLVER

CONVERT MESH

CHECK MESH

PARAMETERS

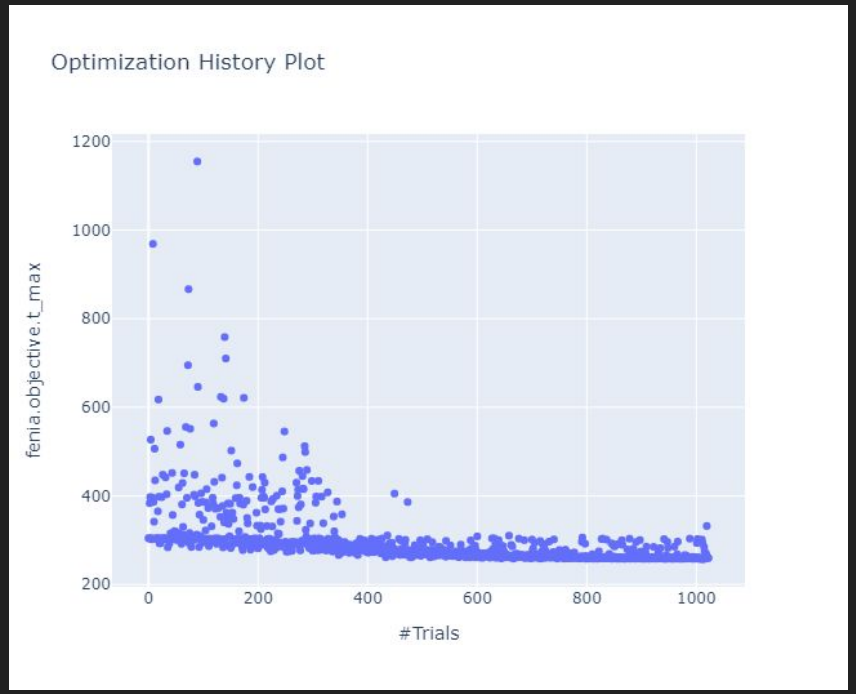
RUN

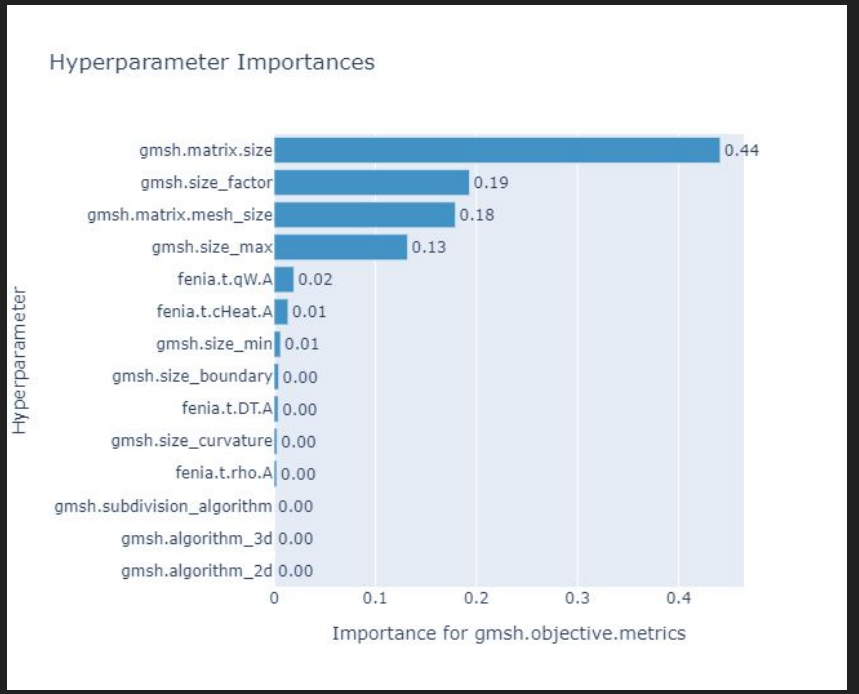
RESULTS

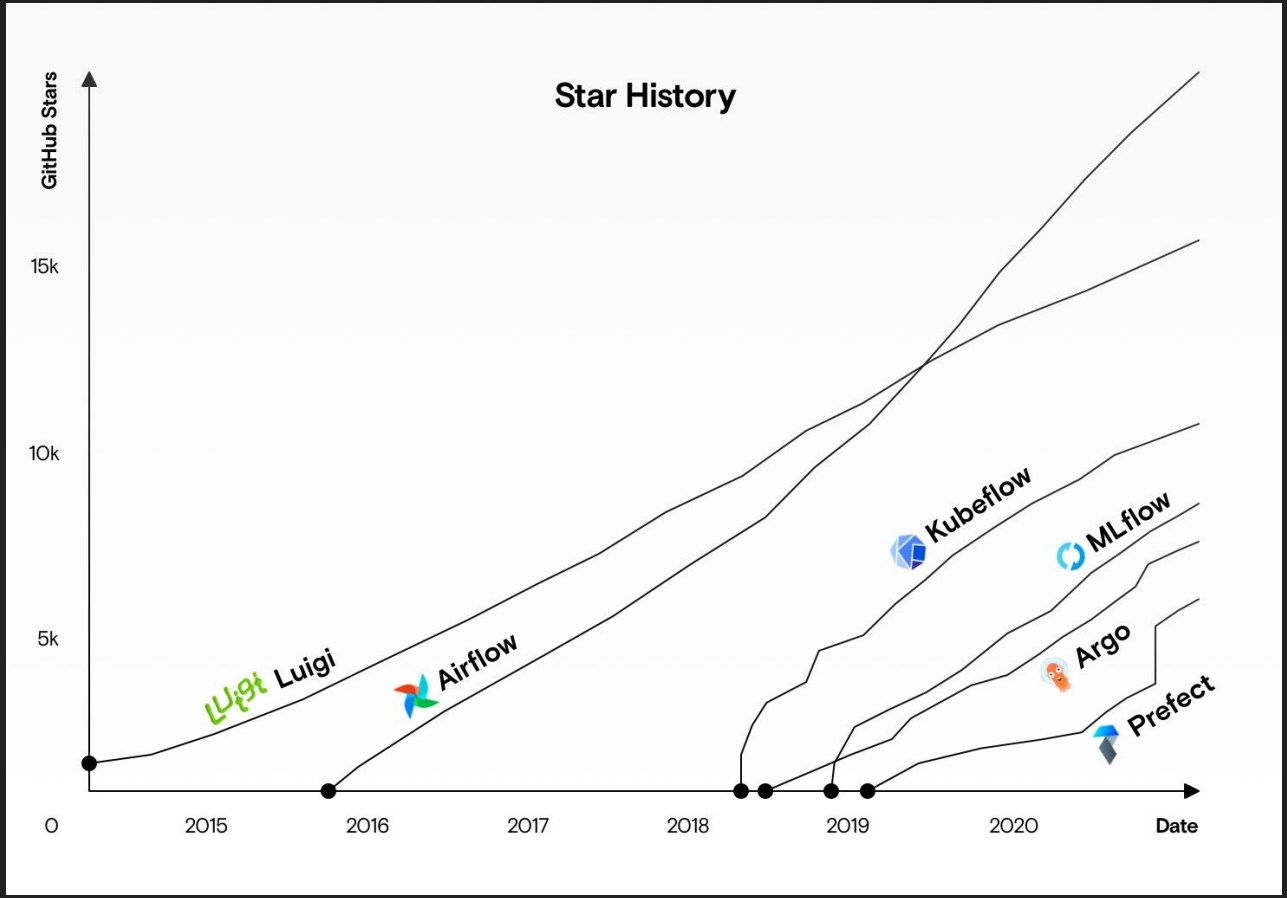
CONSTRAINTS

OBJECTIVES

OPTIMIZATION RESULTS






Star History
GitHub Stars
15k
10k
5k
0
2015
2016
2017
2018
2019
2020
Date
Luigi
Airflow
Kubeflow
MLflow
Argo
Prefect

OPTIMIZATION TODO

- MOUSE integration
- Ray Tune
- Airflow or Nextflow or Ray or ?
- Analytics of optimization algorithms

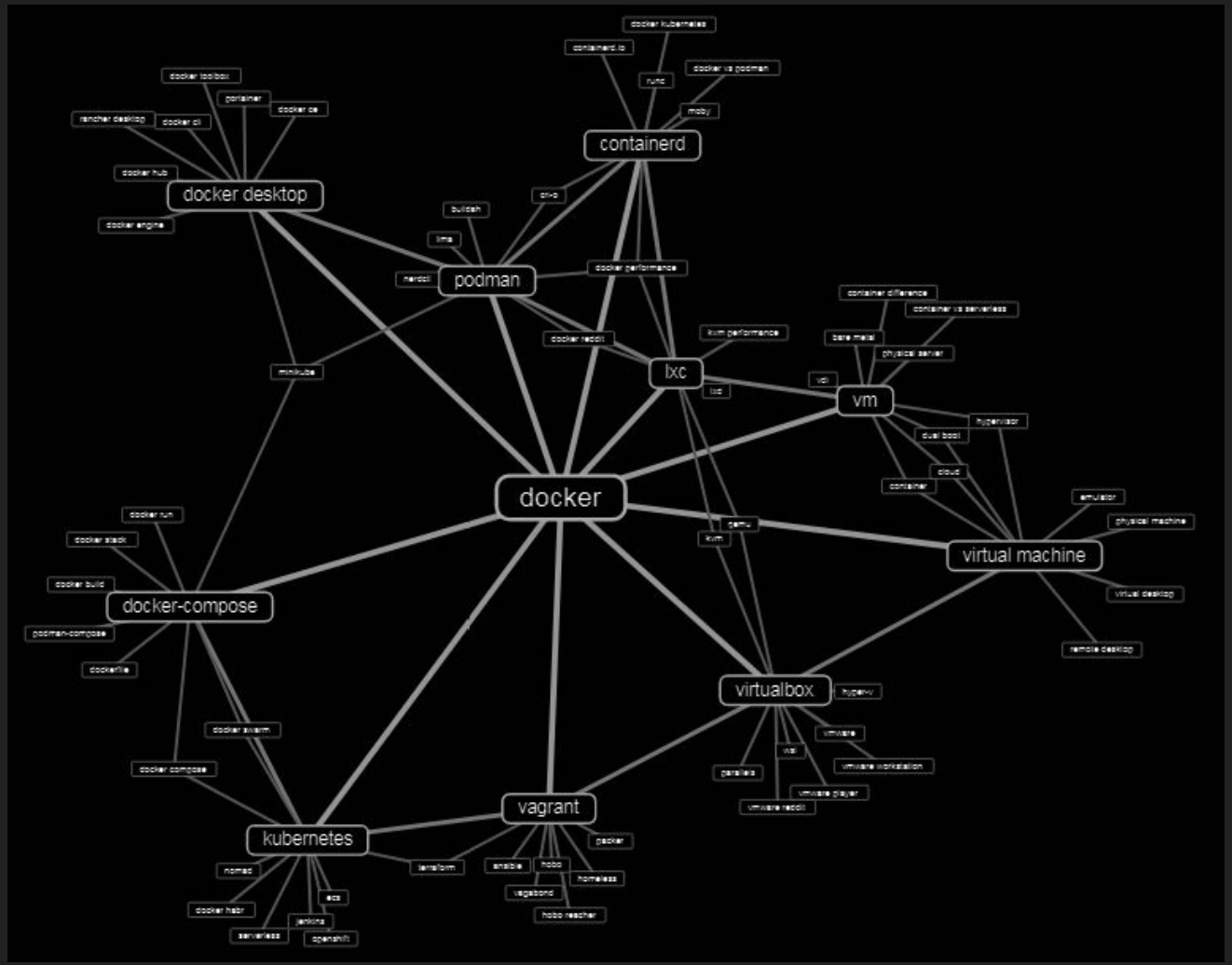
CURRENT STACK

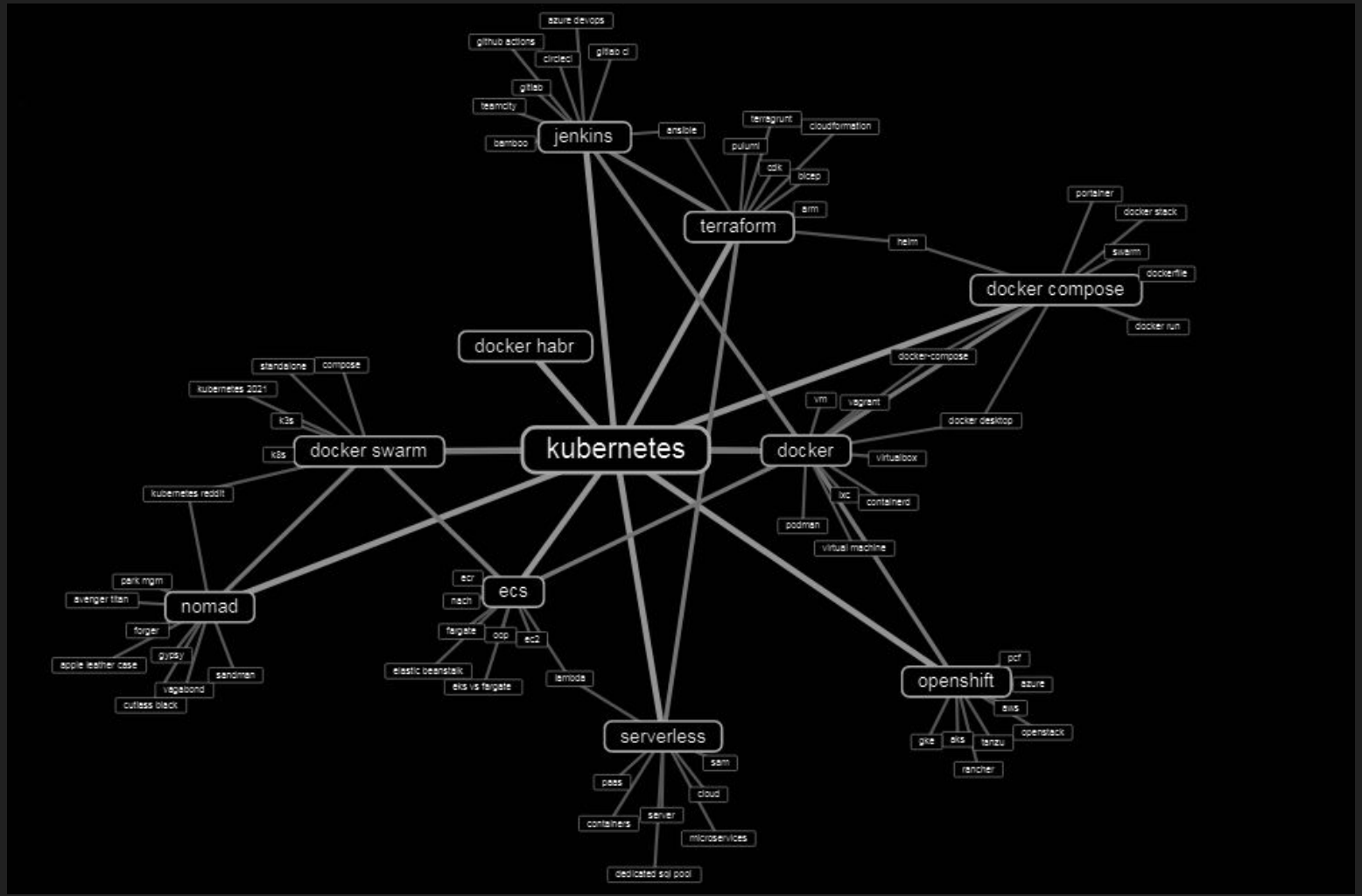
- Python concurrent.futures
- Optuna
- Docker
- Docker Swarm

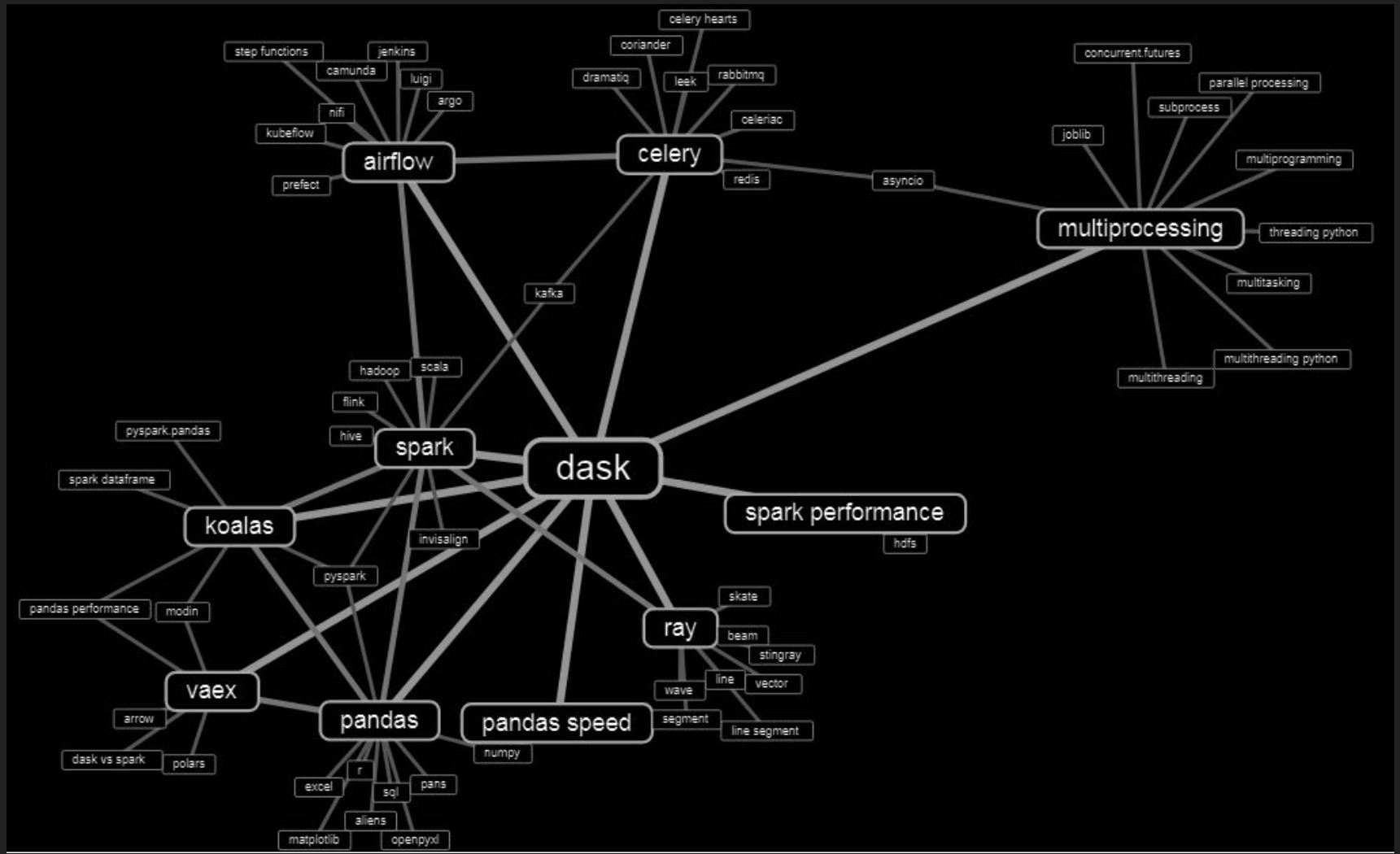
10/10 USEFUL LINKS

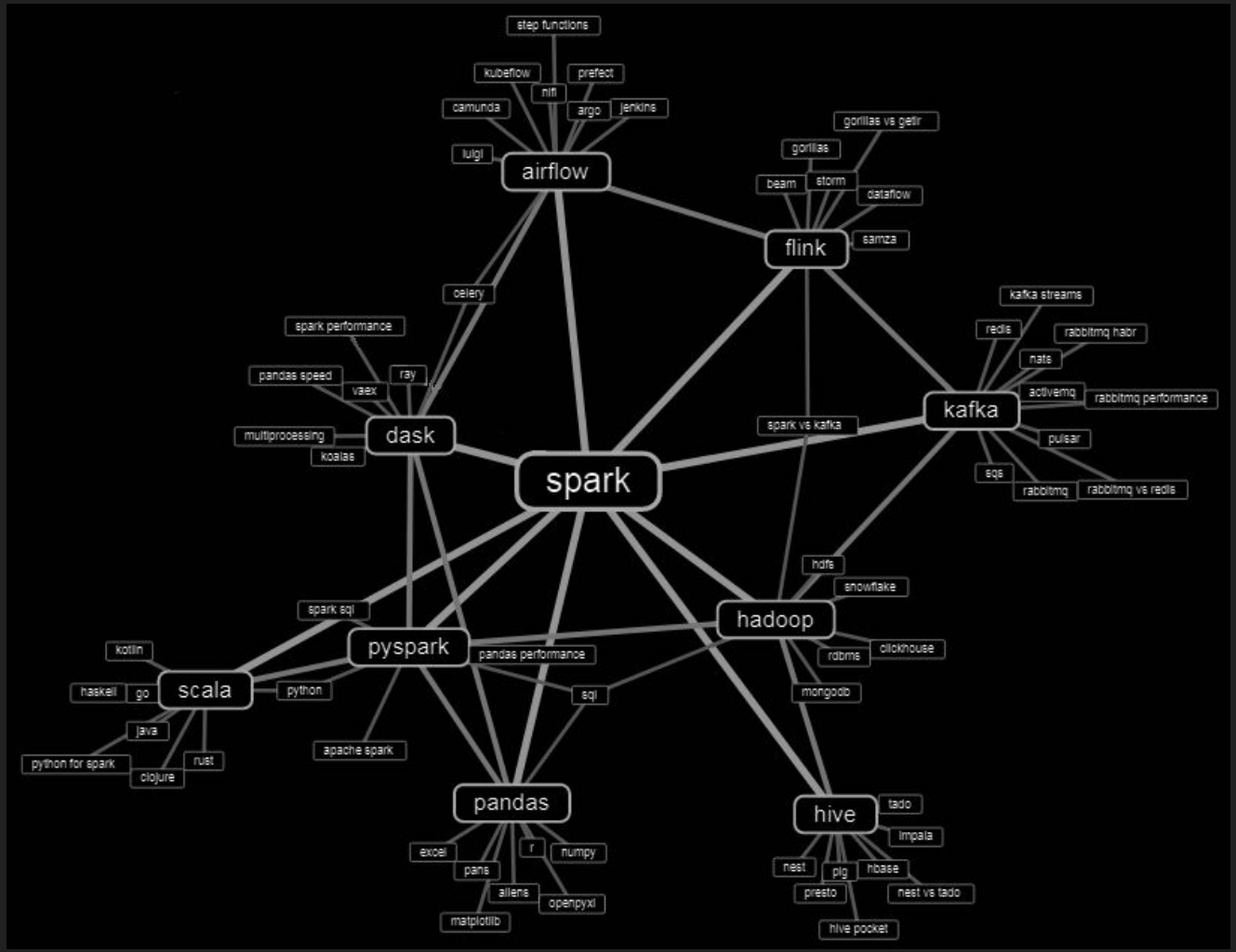
https://anvaka.github.io/vs

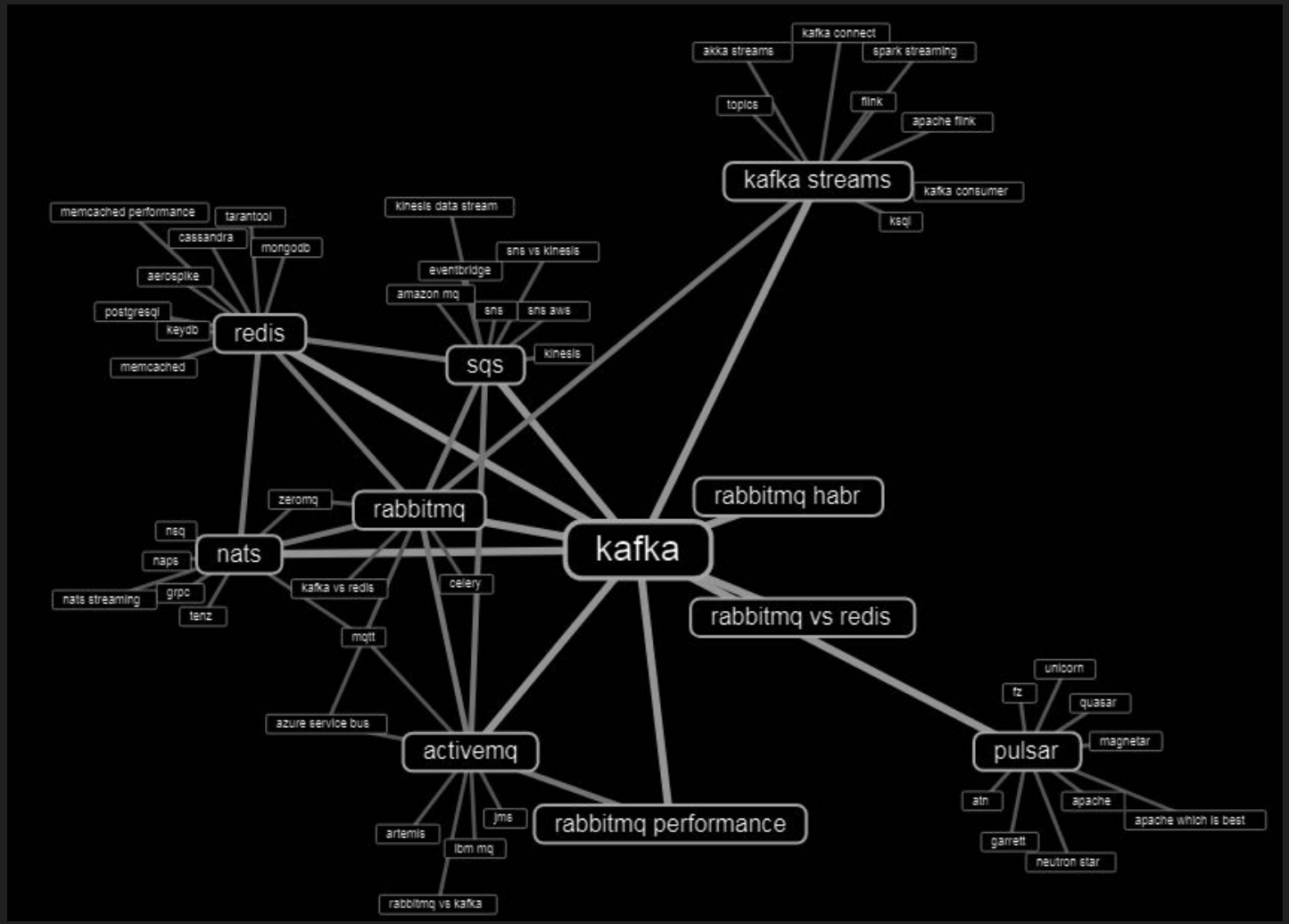
https://stackshare.io

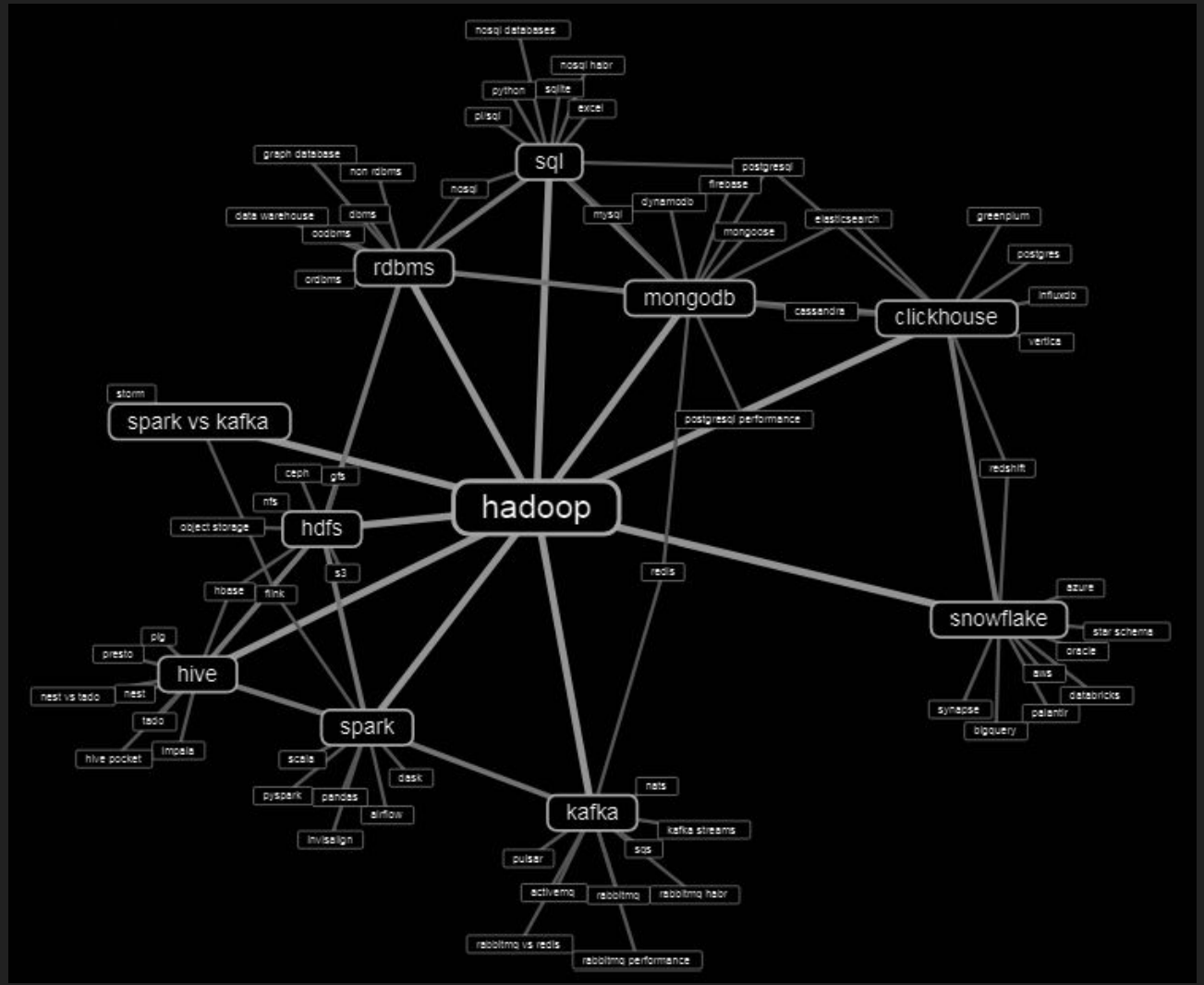

docker kubernetes
containerd.io
runc
docker vs podman
moby
containerd
docker toolbox
container
docker ce
rancher desktop
docker ci
docker hub
docker desktop
docker engine
buildah
cri-o
lxma
podman
nerdctl
docker performance
minikube
container difference
container vs serverless
bare metal
physical server
kvm performance
docker reddit
lxc
vci
vm
hypervisor
dual boot
cloud
container
docker
docker run
docker stack
docker build
docker-compose
podman-compose
dockerfile
qemu
kvm
emulator
physical machine
virtual machine
virtual desktop
remote desktop
virtualbox
hyper-v
vmware
vdi
vmware workstation
parallels
vmware player
vmware reddit
docker swarm
docker compose
vagrant
kubernetes
packer
nomad
terraform
ansible
hobo
homeless
vagabond
hobo reacher
ecs
docker helm
jenkins
serverless
openshift

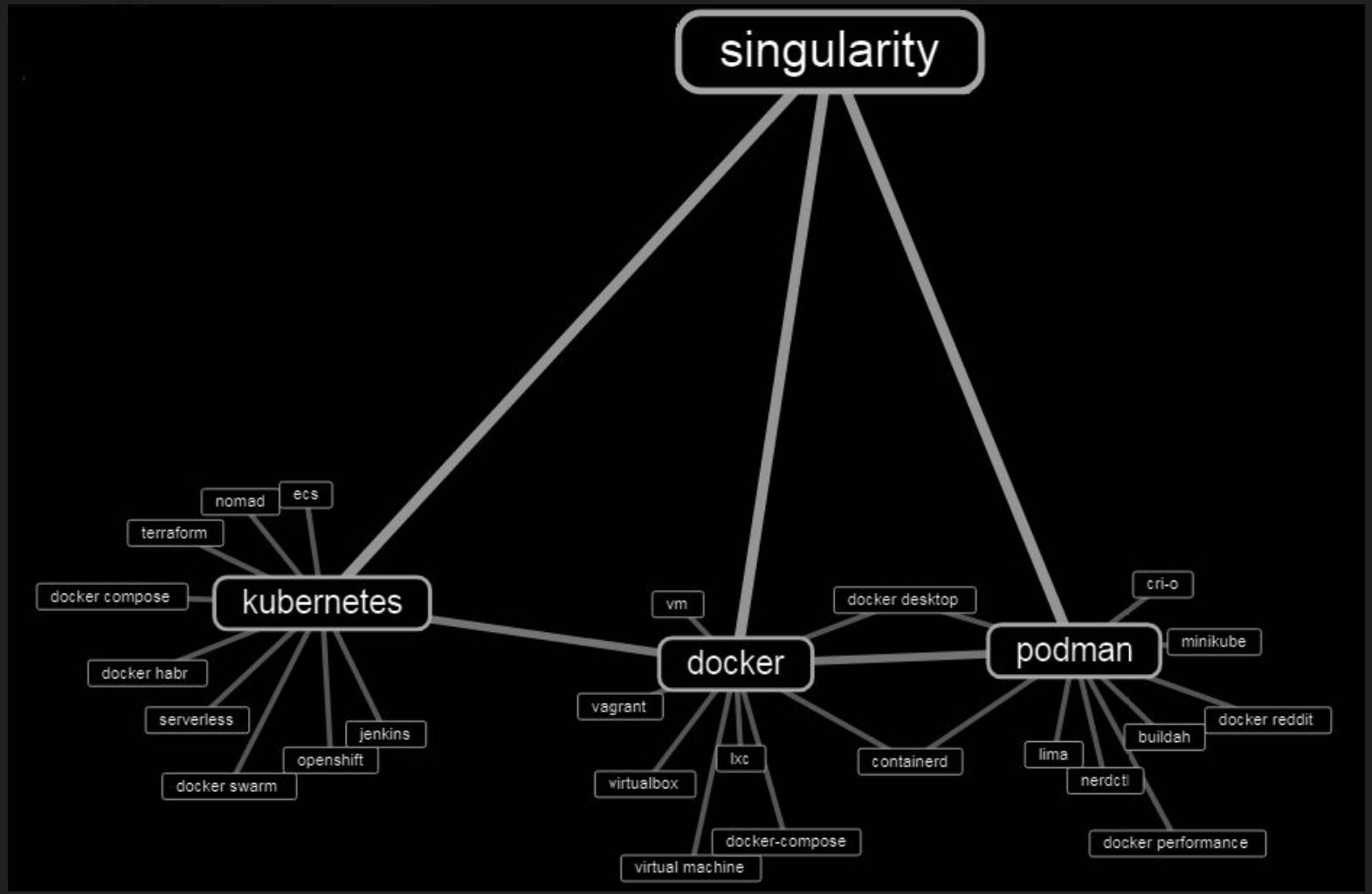

azure devops
github actions
circleci
gitlab ci
gitlab
teamcity
bamboo
jenkins
ansible
terragrunt
cloudformation
pulumi
cdk
bicep
arm
terraform
helm
portainer
docker stack
swarm
dockerfile
docker compose
docker run
docker habr
docker-compose
standalone
compose
kubernetes 2021
k3s
k8s
docker swarm
kubernetes
vm
vagrant
docker desktop
docker
virtualbox
kubernetes reddit
lxc
containerd
podman
virtual machine
park mgm
avenger titan
nomad
forger
gypsy
apple leather case
sandman
vagabond
cutlass black
ecr
nach
ecs
fargate
cop
ec2
elastic beanstalk
eks vs fargate
lamda
serverless
sam
pass
cloud
containers
server
microservices
dedicated sql pool
pcf
azure
openshift
aks
openstack
gke
eks
tanzu
rancher

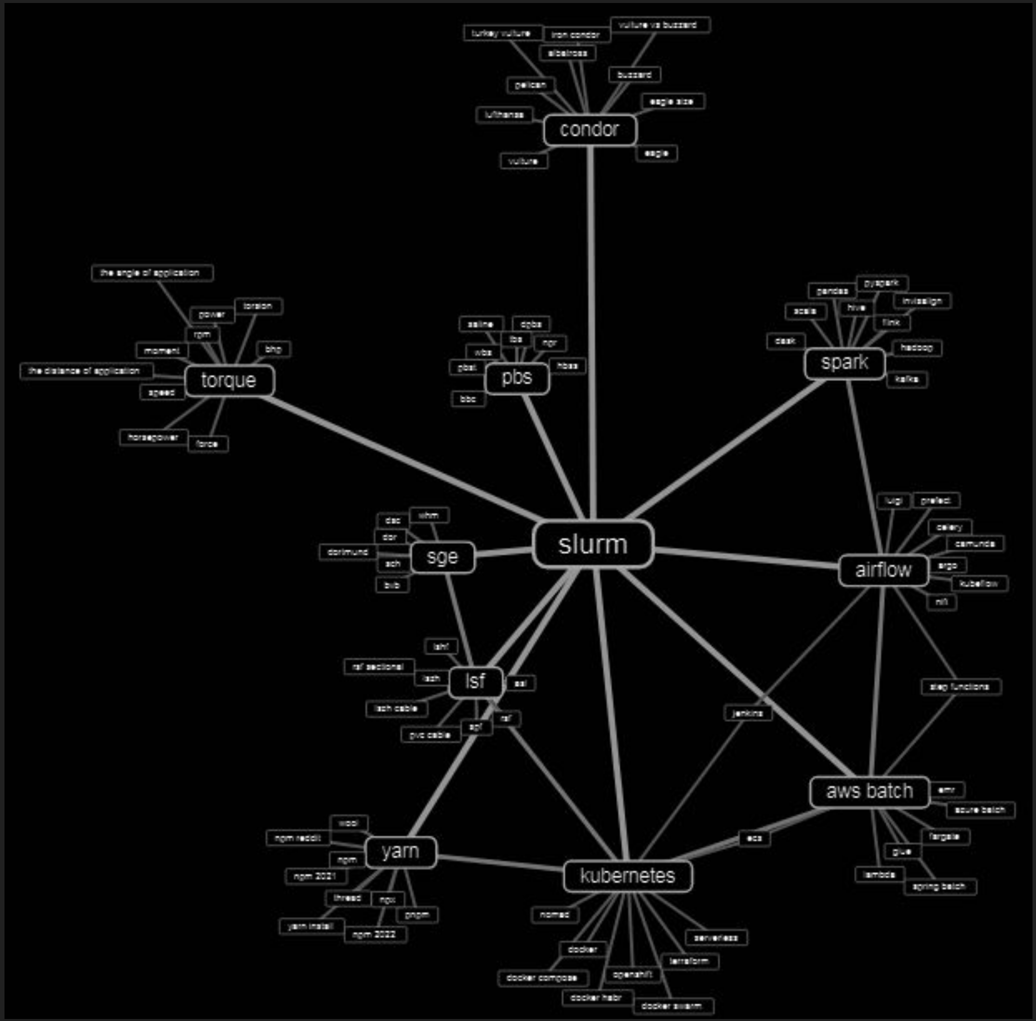

step functions
jenkins
camunda
luigi
argo
nifi
kubeflow
airflow
prefect
celery hearts
coriander
dramatiq
leek
rabbitmq
celeriac
celery
redis
asyncio
concurrent.futures
parallel processing
subprocess
joblib
multiprogramming
multiprocessing
threading python
multitasking
multithreading python
multithreading
kafka
hadoop
scala
flink
hive
spark
pyspark.pandas
spark dataframe
dask
spark performance
hdfs
koalas
invisalign
pyspark
pandas performance
modin
skate
ray
beam
stingray
vaex
wave
line
vector
arrow
pandas
pandas speed
segment
line segment
dask vs spark
polars
numpy
r
excel
sql
pans
aliens
matplotlib
openpyxl

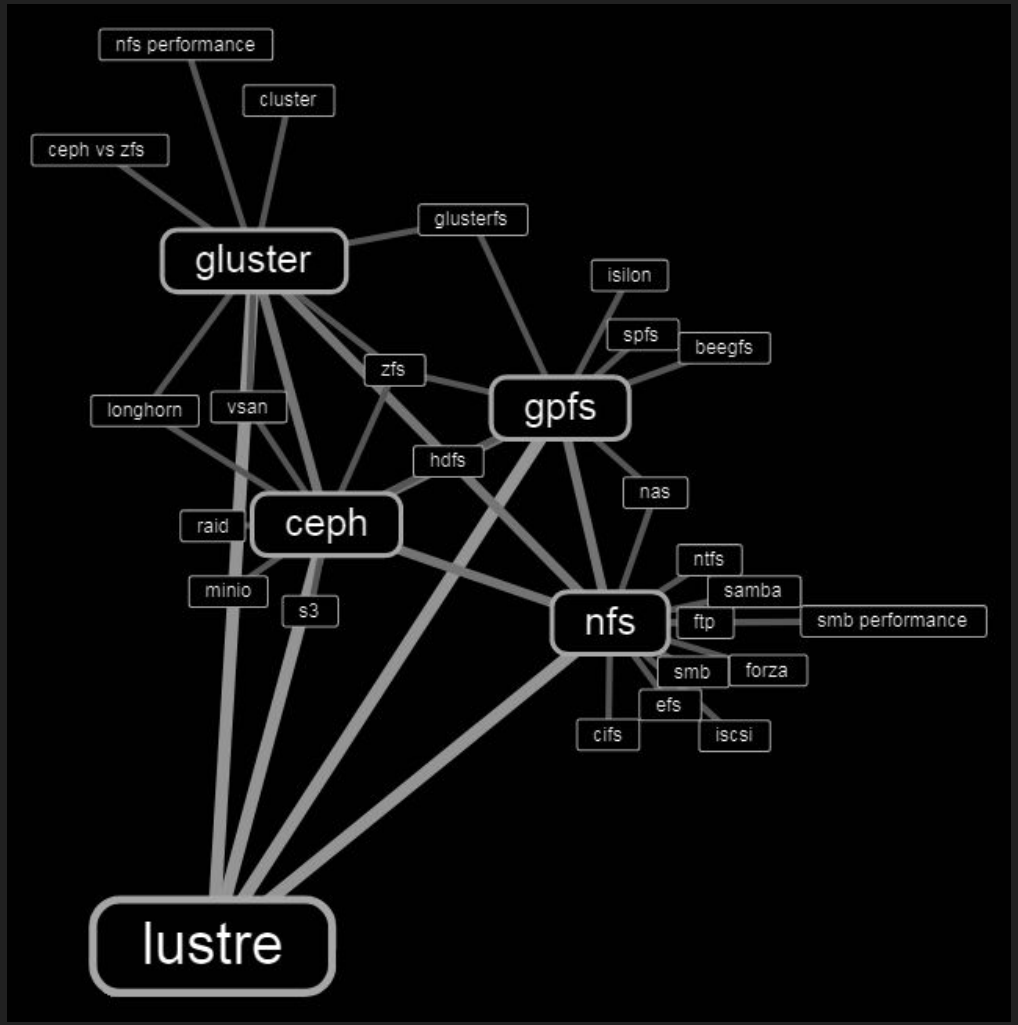

step functions
kubeflow
prefect
nifi
camunda
argo
jenkins
luigi
airflow
gorillas vs getir
gorillas
beam
storm
dataflow
flink
samza
celery
spark performance
pandas speed
ray
vaex
multiprocessing
dask
koalas
spark
spark vs kafka
kafka streams
redis
rabbitmq habr
nats
activemq
rabbitmq performance
kafka
pulsar
sqs
rabbitmq
rabbitmq vs redis
hdfs
snowflake
spark sql
hadoop
pyspark
pandas performance
clickhouse
rdbms
kotlin
haskell
go
scala
python
sql
mongodb
java
python for spark
rust
clojure
apache spark
pandas
excel
r
numpy
pans
allens
openpyxl
matplotlib
hive
tado
impala
nest
plg
hbase
presto
nest vs tado
hive pocket


kafka connect
akka streams
spark streaming
topics
flink
apache flink
kafka streams
kafka consumer
ksql
memcached performance
tarantool
cassandra
mongodb
aerospike
postgresql
keydb
memcached
redis
kinesis data stream
sns vs kinesis
eventbridge
amazon mq
sns
sns aws
kinesis
sqs
zeromq
rabbitmq
rabbitmq habr
nsq
naps
nats
kafka
nats streaming
grpc
tenz
kafka vs redis
celery
mqtt
rabbitmq vs redis
unicorn
fz
quasar
azure service bus
activemq
pulsar
magnetar
atn
apache
apache which is best
garrett
neutron star
jms
rabbitmq performance
artemis
ibm mq
rabbitmq vs kafka

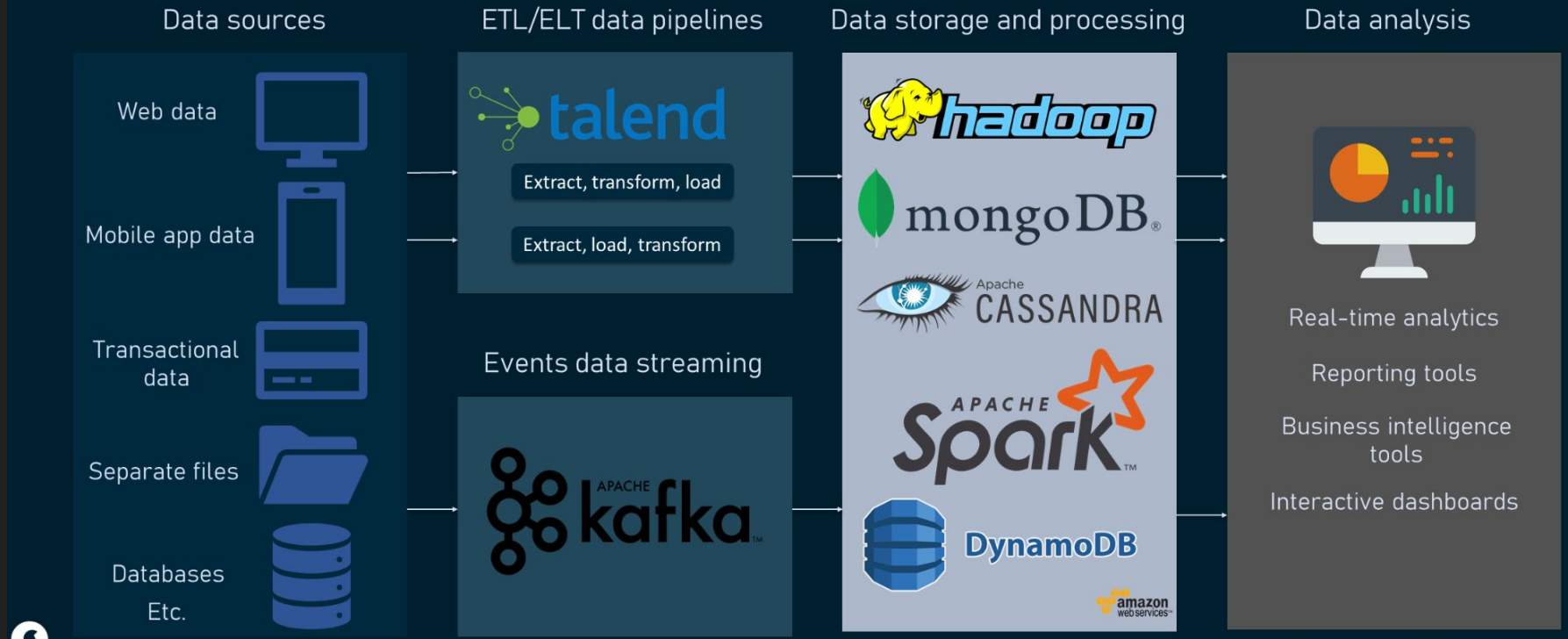

nosql databases
nosql habr
python
sqlite
plsql
excel
sql
graph database
non rdbms
nosql
data warehouse
dbms
oodbms
ordbms
rdbms
postgresql
firebase
dynamodb
mysql
mongoose
elasticsearch
greenplum
postgres
influxdb
mongodb
cassandra
clickhouse
vertica
storm
spark vs kafka
postgresql performance
redshift
hadoop
cech
gfs
nfs
hdfs
object storage
s3
hbase
flink
redis
azure
snowflake
star schema
oracle
aws
databricks
palantir
bigquery
synapse
pig
presto
hive
nest vs tado
nest
tado
hive pocket
impala
spark
scala
dask
pyspark
pandas
airflow
invisalign
kafka
nats
kafka streams
sqs
pulsar
activemq
rabbitmq
rabbitmq habr
rabbitmq vs redis
rabbitmq performance


singularity
kubernetes
docker
podman
nomad
ecs
terraform
docker compose
docker habr
serverless
jenkins
openshift
docker swarm
vm
docker desktop
vagrant
lxc
virtualbox
containerd
virtual machine
docker-compose
cri-o
minikube
buildah
docker reddit
lima
nerdctl
docker performance

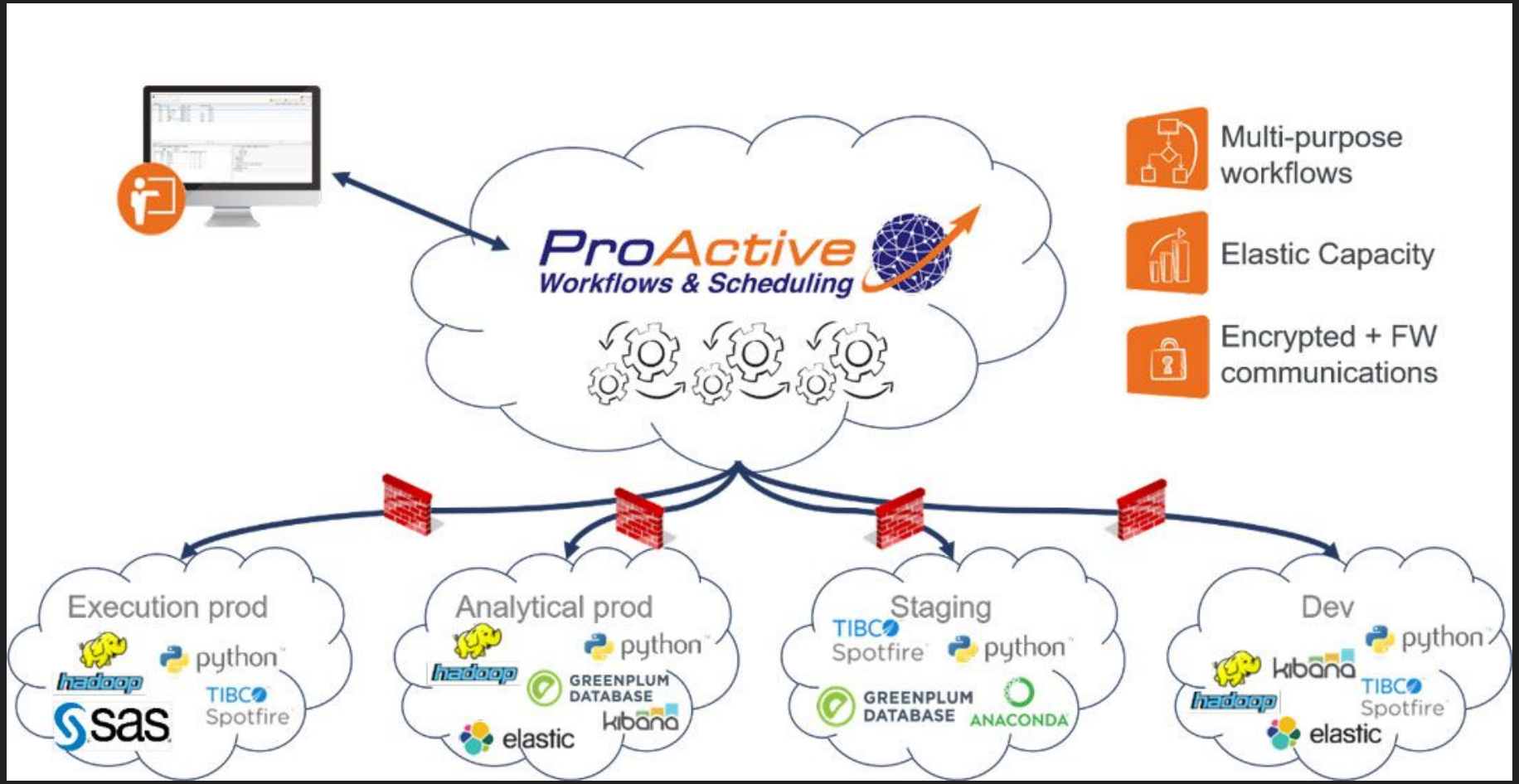

slurm
condor
turkey vulture
iron condor
vulture vs buzzard
albatross
buzzard
pelican
eagle size
lufthansa
eagle
vulture
torque
the angle of application
torsion
power
rpm
moment
bhp
the distance of application
speed
horsepower
force
pbs
saline
pbst
bbc
spark
pyspark
pandas
invisalign
scala
hive
flink
dask
hadoop
kafka
sge
dortmund
airflow
luigi
prefect
celery
camunda
argo
kubeflow
nifi
step functions
jenkins
lsf
rsf sectional
lsch cable
pvc cable
aws batch
emr
azure batch
fargate
glue
lambda
spring batch
ecs
yarn
npm reddit
npm
npm 2021
thread
npx
pnpm
yarn install
npm 2022
kubernetes
nomad
serverless
docker
terraform
docker compose
openshift
docker hub
docker swarm

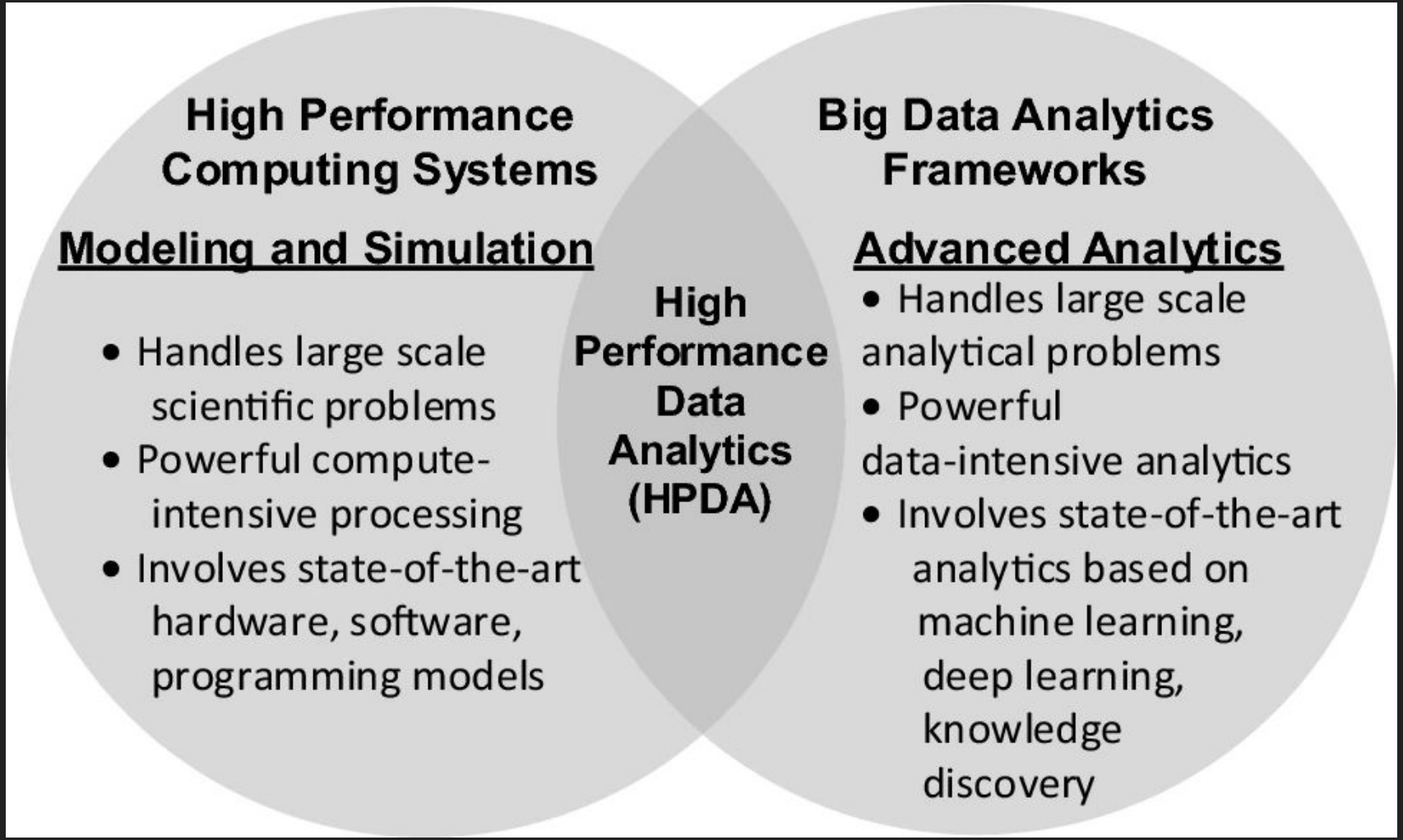

nfs performance
cluster
ceph vs zfs
gluster
glusterfs
isilon
spfs
beegfs
zfs
gpfs
longhorn
vsan
hdfs
nas
raid
ceph
ntfs
samba
minio
s3
nfs
ftp
smb performance
smb
forza
efs
cifs
iscsi
lustre

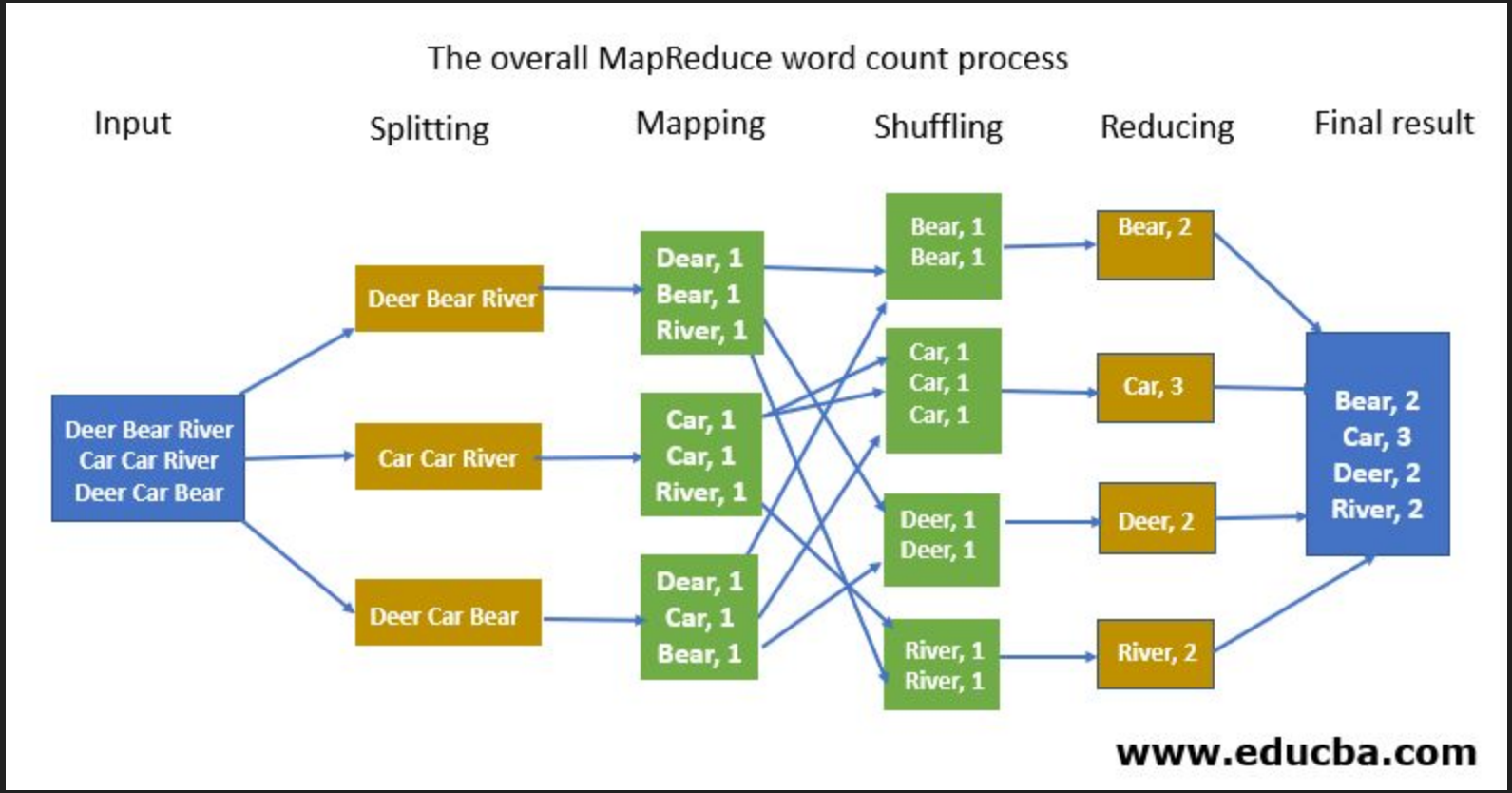
BIG DATA ANALYTICS PROCESSES



altexsoft
software r&d engineering

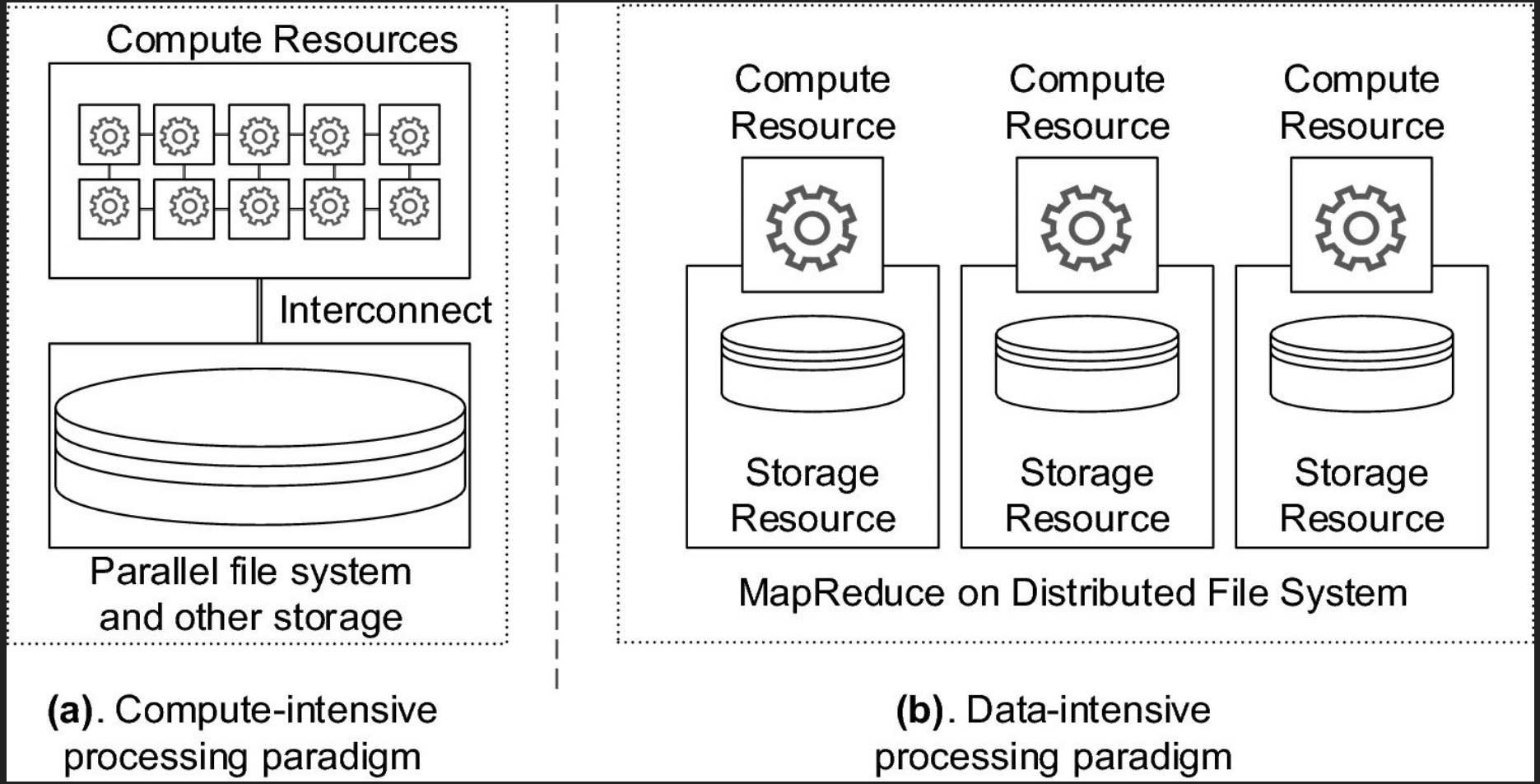

ProActive
Workflows & Scheduling
Multi-purpose workflows
Elastic Capacity
Encrypted + FW communications
Execution prod
python
hadoop
TIBCO Spotfire
sas
Analytical prod
python
hadoop
GREENPLUM DATABASE
elastic
kibana
Staging
TIBCO Spotfire
python
GREENPLUM DATABASE
ANACONDA
Dev
python
kibana
hadoop
TIBCO Spotfire
elastic


High Performance Computing Systems
Modeling and Simulation
• Handles large scale scientific problems
• Powerful compute-intensive processing
• Involves state-of-the-art hardware, software, programming models
High Performance Data Analytics (HPDA)
Big Data Analytics Frameworks
Advanced Analytics
• Handles large scale analytical problems
• Powerful data-intensive analytics
• Involves state-of-the-art analytics based on machine learning, deep learning, knowledge discovery


The overall MapReduce word count process
Input
Splitting
Mapping
Shuffling
Reducing
Final result
Deer Bear River
Car Car River
Deer Car Bear
Deer Bear River
Car Car River
Deer Car Bear
Dear, 1
Bear, 1
River, 1
Car, 1
Car, 1
River, 1
Dear, 1
Car, 1
Bear, 1
Bear, 1
Bear, 1
Car, 1
Car, 1
Car, 1
Deer, 1
Deer, 1
River, 1
River, 1
Bear, 2
Car, 3
Deer, 2
River, 2
Bear, 2
Car, 3
Deer, 2
River, 2
www.educba.com

map, filter, and reduce explained with emoji 😂

```
map([🐮, 🥓, 🐔, 🥒], cook)
=> [🍔, 🍟, 🍗, 🥤]

filter([🍔, 🍟, 🍗, 🥤], isVegetarian)
=> [🍟, 🥤]

reduce([🍔, 🍟, 🍗, 🥤], eat)
=> 💩
```



(a). Compute-intensive processing paradigm

(b). Data-intensive processing paradigm

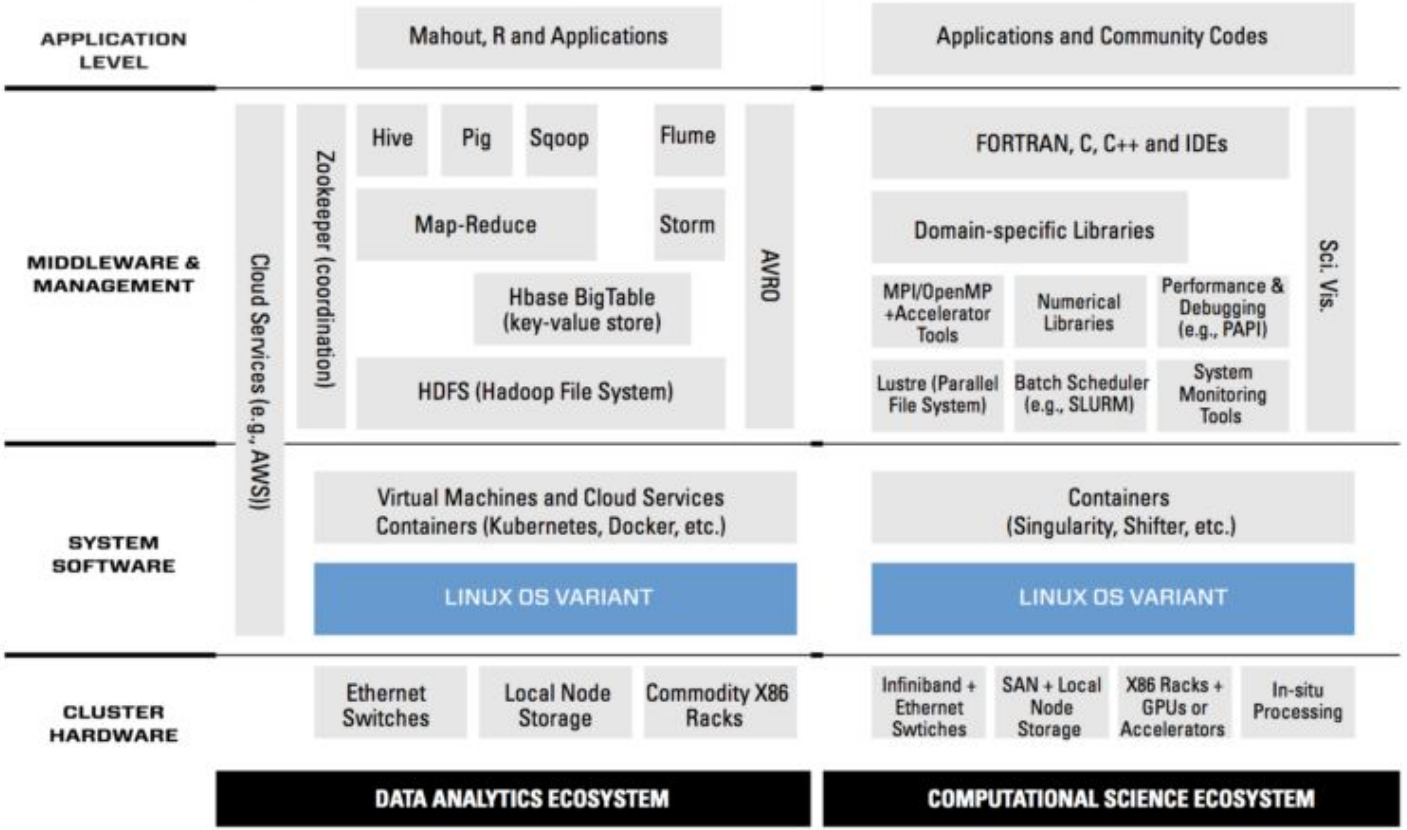
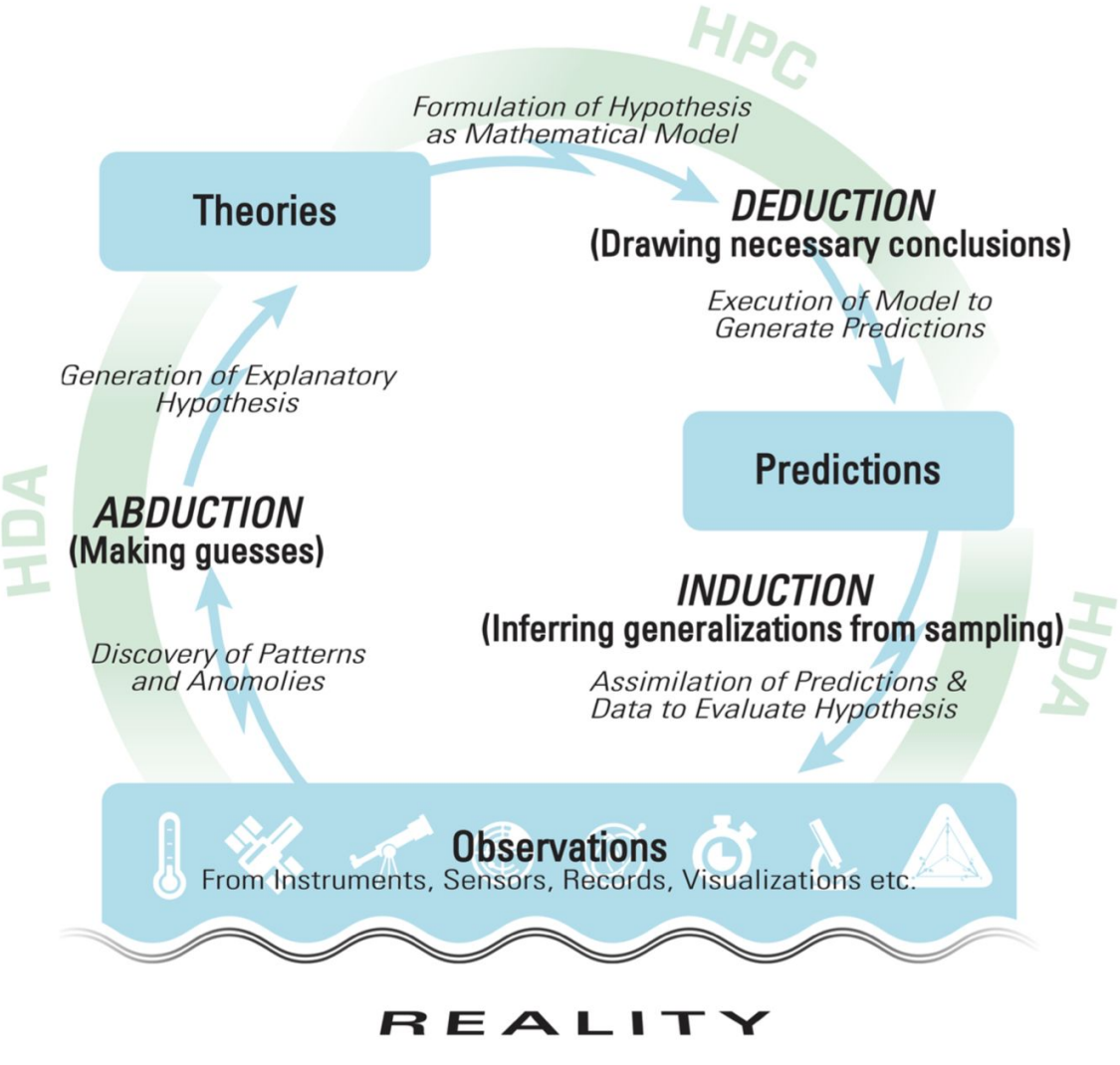


Figure 1: Different software ecosystems for high-end Data Analytics and for traditional Computational Science. [Credit: Reed and Dongarra [66]]


HPC
Formulation of Hypothesis as Mathematical Model
Theories
DEDUCTION
(Drawing necessary conclusions)
Execution of Model to Generate Predictions
Predictions
INDUCTION
(Inferring generalizations from sampling)
Assimilation of Predictions & Data to Evaluate Hypothesis
HDA
HDA
Discovery of Patterns and Anomolies
ABDUCTION
(Making guesses)
Generation of Explanatory Hypothesis
Observations
From Instruments, Sensors, Records, Visualizations etc.
REALITY

Kubernetes

- Complex installation
- More complex with a high learning curve, but more powerful
- Supports auto-scaling
- Built in monitoring
- Manual setup of load balancer
- Need for a separate CLI tool

Docker Swarm

- Easier installation
- More lightweight and easier to use, but limited functionality
- Manual scaling
- Needs third party tools for monitoring
- Auto load balancing
- Integrates docker CLI