Service Orchestration

BigData from testing to production

PhD. Ronal Muresano

Course Roadmap

Docker Swarm

Orchestration in a real private cluster (Deploying and Creating services)

Kubernetes

Production Orchestration. Automating deployment, scaling, and management of containerized applications

Orchestration for developing

Docker and Docker-Compose.

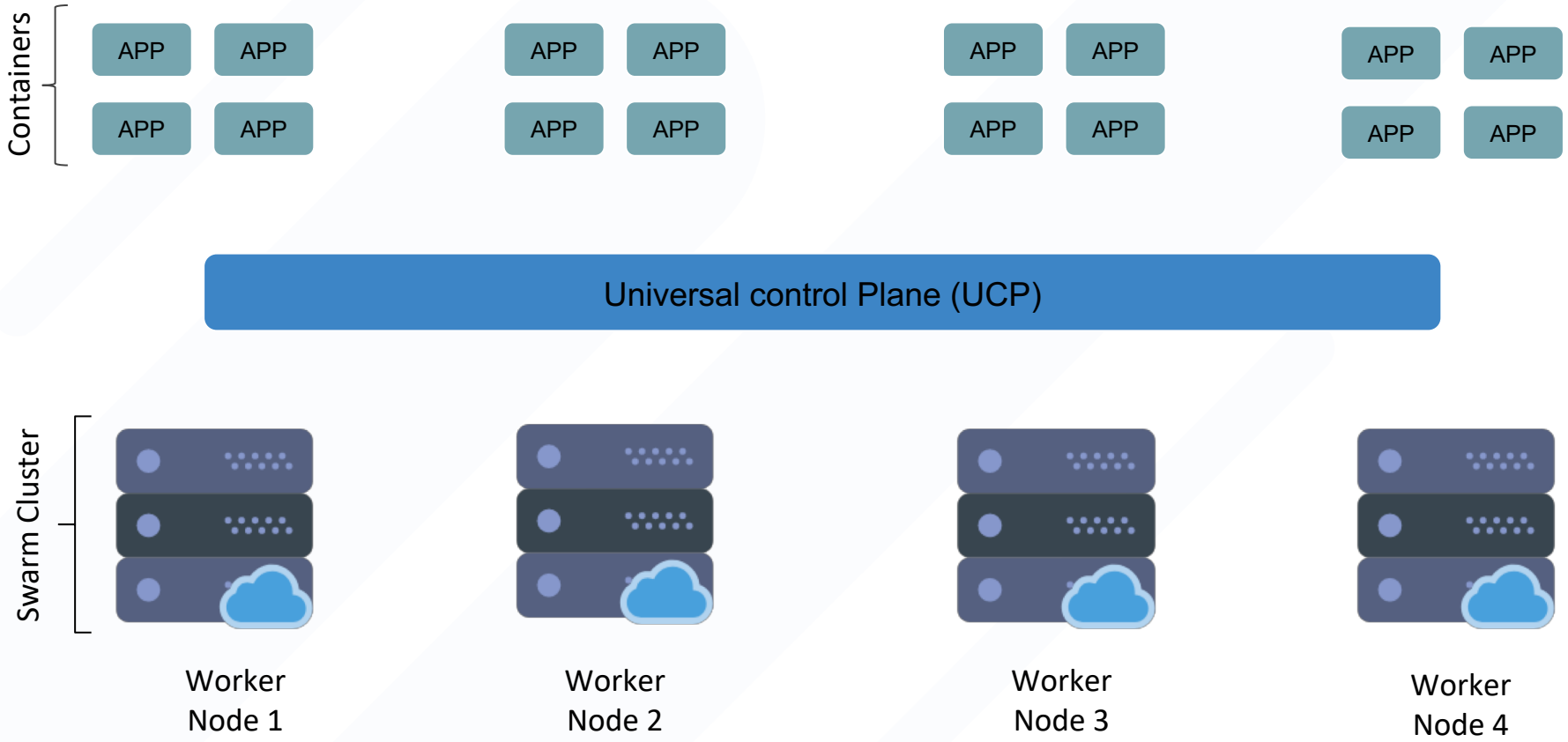
Big Data Challenges

An overview applied to the SMEs. Orchestration Philosophy

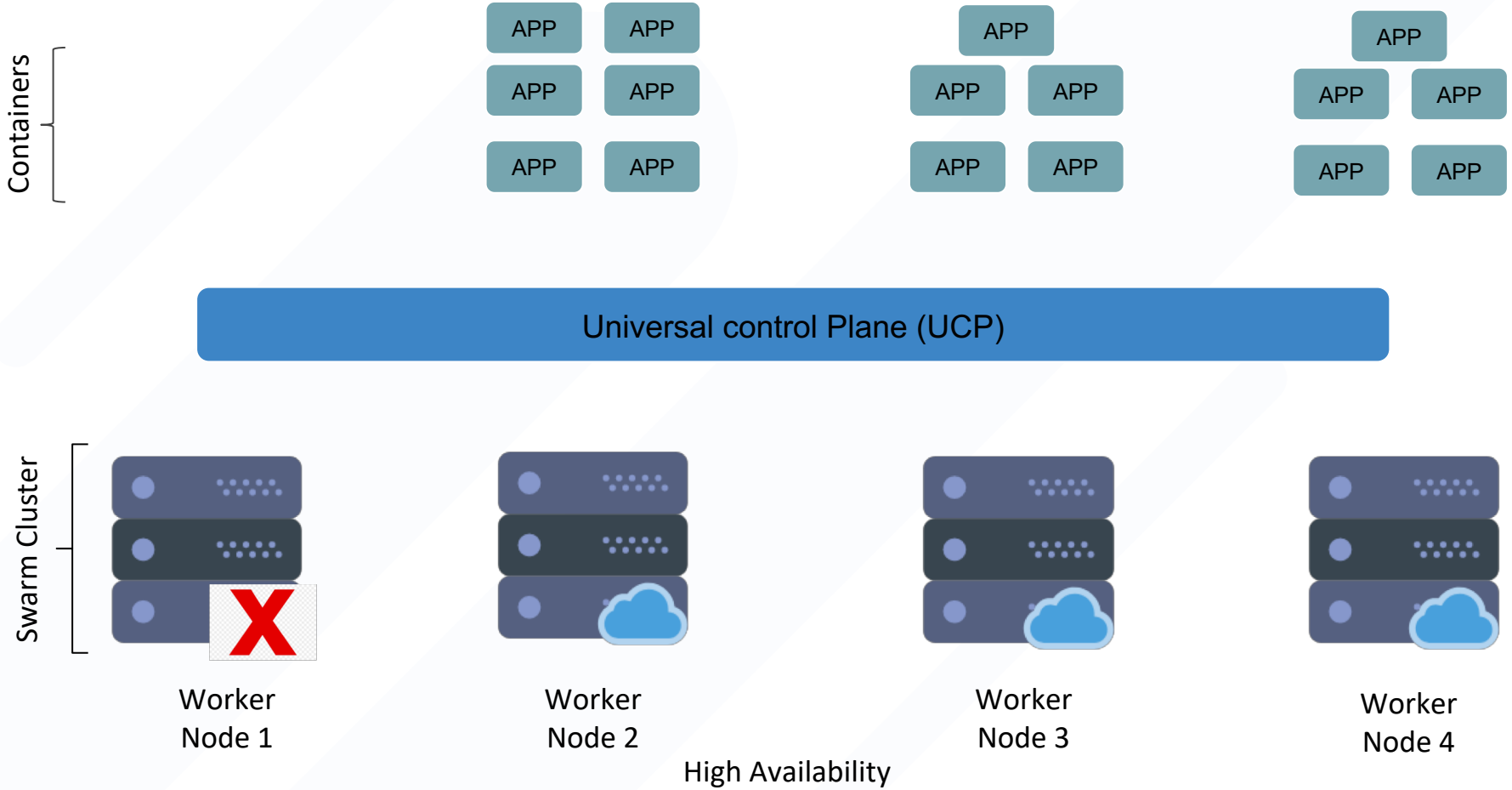
Cloud Infraestructure

Google Cloud and AWS deployments

Service Orchestration in a Cluster (Swarm)



Service Orchestration in a Cluster



Docker-Swarm (Multi-container Services)

It allows to developers to establish and manage a cluster of Docker nodes as a single virtual system.

Clustering is an important feature for container technology, because it creates a cooperative group of systems that can provide redundancy, enabling Docker Swarm failover if one or more nodes experience an outage.

Swarm assigns containers to underlying nodes and optimizes resources by automatically scheduling container workloads to run on the most appropriate host.

IT administrator controls Swarm through a swarm manager, which orchestrates and schedules containers.

Source: https://searchitoperations.techtarget.com/definition/Docker-Swarm

Docker-Swarm (Multi-container Services)

Feature HighLights

- **Cluster management integrated with Docker Engine:** Use the Docker Engine CLI to create a swarm of Docker Engines where you can deploy application services.
- **Declarative service model:** Docker Engine uses a declarative approach to let you define the desired state of the various services in your application stack.
- **Scaling:** For each service, you can declare the number of tasks you want to run.
- **Service discovery:** Swarm manager nodes assign each service in the swarm a unique DNS name and load balances running containers.
- **Load balancing:** You can expose the ports for services to an external load balancer.
- **Secure by default:** Each node in the swarm enforces TLS mutual authentication and encryption to secure communications between itself and all other nodes.
- **Rolling updates:** At rollout time you can apply service updates to nodes incrementally.

Api: Interact with service objects

Orchestrator: Creates tasks for service objects.

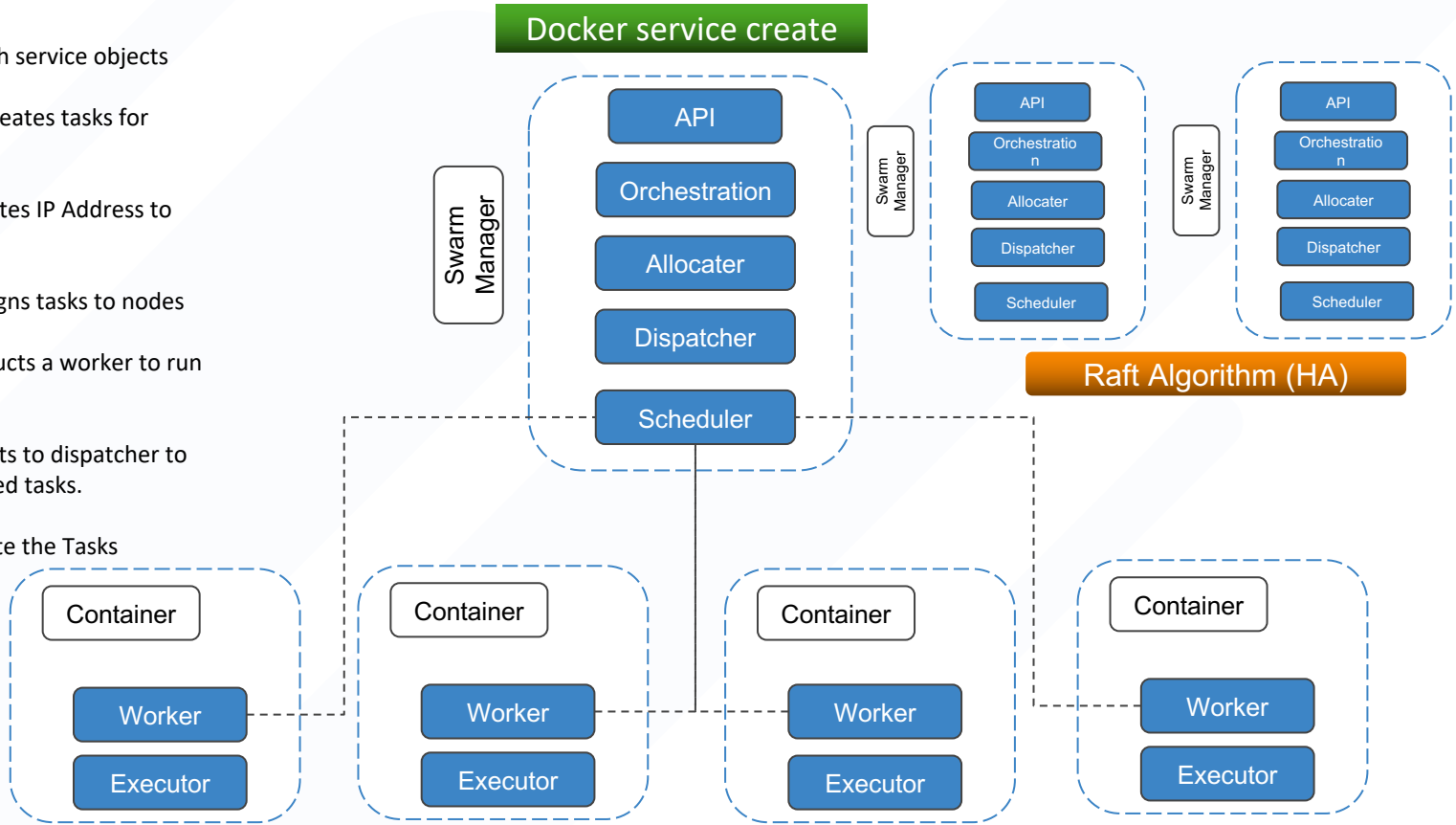
Allocater: Allocates IP Address to tasks.

Dispatcher: Assigns tasks to nodes

Scheduler: Instructs a worker to run a task

Worker: Connects to dispatcher to check for assigned tasks.

Executor: Execute the Tasks



Docker-Swarm (Multi-container Services)

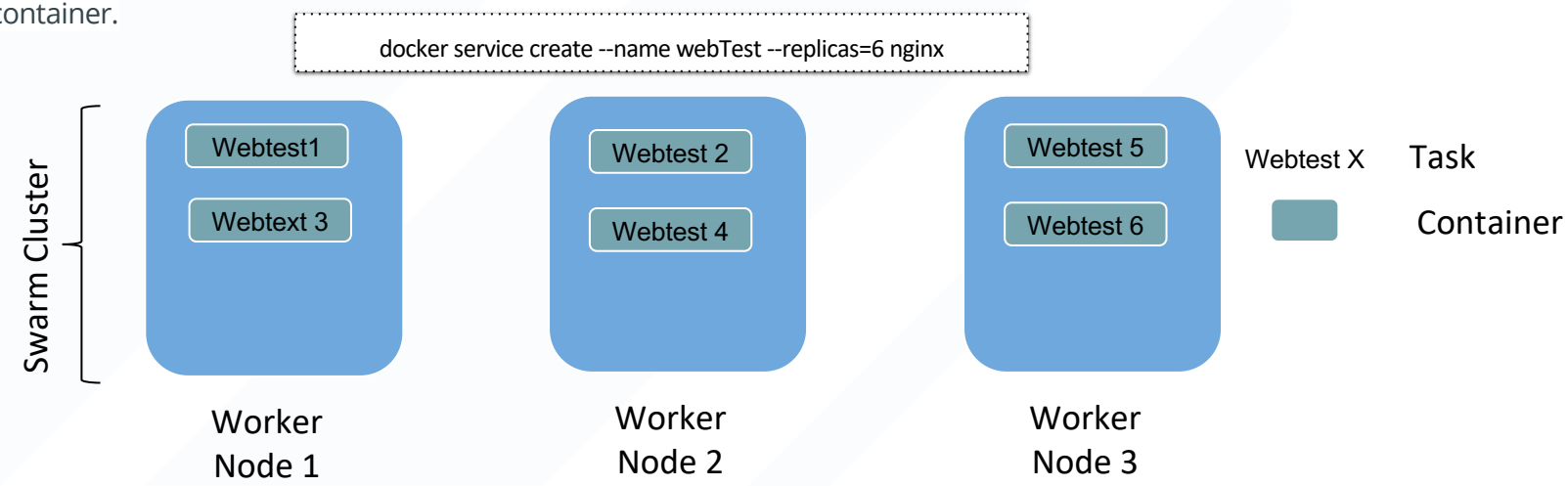
Key terms to know in Swarm :

Node: A node is a physical or virtual machine running an instance of the Docker Engine in swarm mode.

Services: A service is the definition of how you want to run your application containers in a swarm. At the most basic level a service defines which container image to run in the swarm and which commands to run in the container.

Containers: It is a runtime instance of a docker image.

Task: A task is the atomic unit of scheduling within a swarm. A task carries a Docker container and the commands to run inside the container.



Docker-Swarm (Multi-container Services)

Example 1: Starting a docker swarm example

```
docker@manager-bdaas:~$ docker service create --name ronaltest --replicas=6 nginx
xvt9m4hpf6m92o9shisffdrzx
overall progress: 6 out of 6 tasks
1/6: running   [==================================================>]
2/6: running   [==================================================>]
3/6: running   [==================================================>]
4/6: running   [==================================================>]
5/6: running   [==================================================>]
6/6: running   [==================================================>]
verify: Service converged
```

```
docker@manager-bdaas:~$ docker service ps ronaltest
ID                  NAME                IMAGE               NODE                DESIRED STATE       CURRENT STATE                ERROR               PORTS
irqk1m2m43dk        ronaltest.1         nginx:latest        manager-bdaas       Running             Running about a minute ago
rmhktzdgb9os        ronaltest.2         nginx:latest        worker-bdaas-2      Running             Running about a minute ago
dlwwydi7su7v        ronaltest.3         nginx:latest        manager-bdaas       Running             Running about a minute ago
cmpxajc1z9ih        ronaltest.4         nginx:latest        worker-bdaas-1      Running             Running about a minute ago
p4dx10j4gdg2        ronaltest.5         nginx:latest        worker-bdaas-3      Running             Running about a minute ago
kevxj952q2ma        ronaltest.6         nginx:latest        worker-bdaas-2      Running             Running about a minute ago
docker@manager-bdaas:~$
```

Docker-Swarm (Multi-container Services)

Container

- Runs on one host
- Must share network and storage per container.
- Tough to scale and make highly available

Service

- Automatically runs across the cluster
- High Availability is built in.
- Scalability of the app is easy.
- Exposing network and storage is easy

Docker-Swarm (Cluster Creation example cluster on Local)

Exercise 1: Creating a Swarm Cluster Approximate Time 15-20 minutes

Creating Virtual Machines

docker-machine create --driver virtualbox --virtualbox-memory <SIZE> <Name-manager>

```
edu241-224:~ rmuresano$ docker-machine ls
NAME             ACTIVE   DRIVER       STATE     URL                         SWARM   DOCKER     ERRORS
manager-bdaas    -        virtualbox   Running   tcp://192.168.99.103:2376           v18.09.6
worker-bdaas-1   -        virtualbox   Running   tcp://192.168.99.104:2376           v18.09.6
worker-bdaas-2   -        virtualbox   Running   tcp://192.168.99.105:2376           v18.09.6
worker-bdaas-3   -        virtualbox   Running   tcp://192.168.99.106:2376           v18.09.6
edu241-224:~ rmuresano$
```

```
$ base=https://github.com/docker/machine/releases/download/v0.16.0 &&
  curl -L $base/docker-machine-$(uname -s)-$(uname -m) >/tmp/docker-machine &&
  sudo install /tmp/docker-machine /usr/local/bin/docker-machine
```

Getting the Manager Info

docker-machine env manager-bdaas

```
edu241-224:~ rmuresano$ docker-machine env manager-bdaas
export DOCKER_TLS_VERIFY="1"
export DOCKER_HOST="tcp://192.168.99.103:2376"
export DOCKER_CERT_PATH="/Users/rmuresano/.docker/machine/machines/manager-bdaas"
export DOCKER_MACHINE_NAME="manager-bdaas"
# Run this command to configure your shell:
# eval $(docker-machine env manager-bdaas)
edu241-224:~ rmuresano$
```

Docker-Swarm (Cluster Creation example on Local)

Init the Swarm Cluster

```
docker-machine ssh manager-bdaas “docker swarm init --advertise-addr  W.X.Y.Z”
```

This command return the token which has to be included to the workers in order to join the cluster

Joining the Worker to the Swarm Cluster

```
docker-machine ssh worker-bdaas-N "docker swarm join --token $SWARM_TOKENSWARM_MANAGER_IP:2377"
```

Adding a new worker

```
docker-machine ssh manager-bdaas “docker swarm join-token worker -q”
```

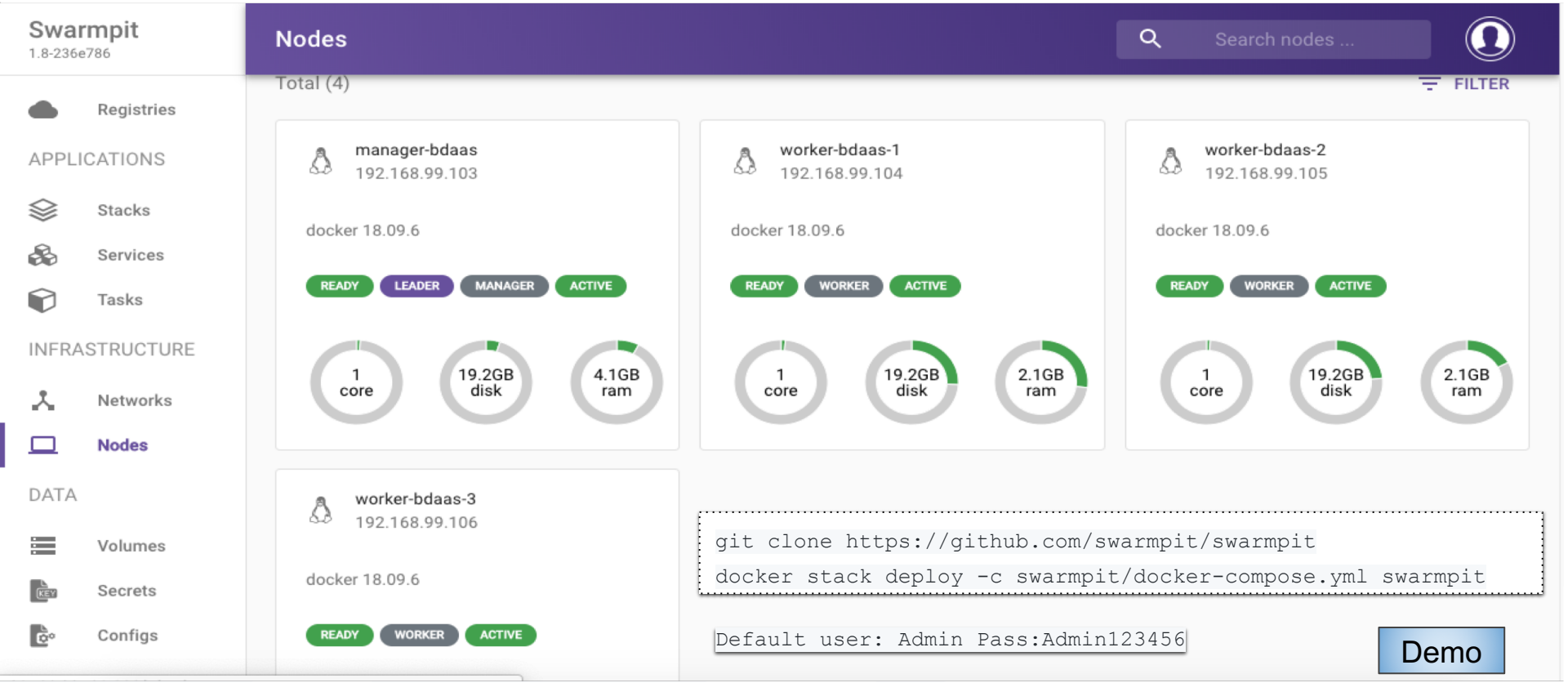
Docker-Swarm (Cluster Creation example on Local)

Getting the swarm Cluster Info

```
edu241-224:~ rmuresano$ docker-machine ls
NAME             ACTIVE   DRIVER       STATE     URL                         SWARM   DOCKER     ERRORS
manager-bdaas    -        virtualbox   Running   tcp://192.168.99.103:2376           v18.09.6
worker-bdaas-1   -        virtualbox   Running   tcp://192.168.99.104:2376           v18.09.6
worker-bdaas-2   -        virtualbox   Running   tcp://192.168.99.105:2376           v18.09.6
worker-bdaas-3   -        virtualbox   Running   tcp://192.168.99.106:2376           v18.09.6
edu241-224:~ rmuresano$ docker-machine ssh manager-bdaas
   ( '>')
  /) TC (\   Core is distributed with ABSOLUTELY NO WARRANTY.
 (/-_--_-\)           www.tinycorelinux.net

docker@manager-bdaas:~$ docker node ls
ID                            HOSTNAME            STATUS              AVAILABILITY        MANAGER STATUS      ENGINE VERSION
yldimz8l6mbnnmyf29p8xvqal *   manager-bdaas       Ready               Active              Leader              18.09.6
zqkjk21cq126ht73vu33ixk80     worker-bdaas-1      Ready               Active                                  18.09.6
j9ppbjkwuhemddybqodaffue3     worker-bdaas-2      Ready               Active                                  18.09.6
f6ylgd42ffxwf6zgwcr5ml7lj     worker-bdaas-3      Ready               Active                                  18.09.6
docker@manager-bdaas:~$
```

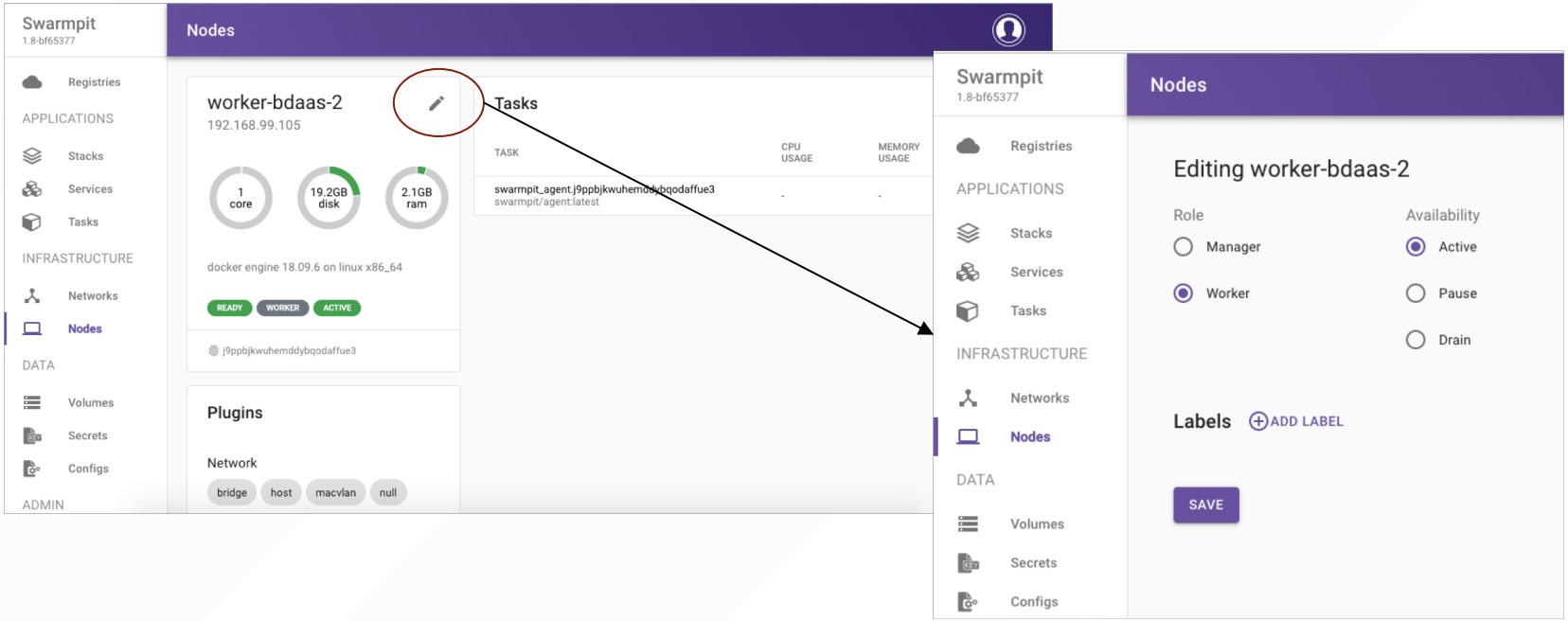
Docker-Swarm (Visualizer)*



Docker-Swarm (Visualizer)*

Exercise 2: Knowing and Playing with the SwarmPit interface 10-15 min.

Important: Try to stop a node and see what happens with the processes.



Docker Swarm Stack File

It is used to define a complete, highly available, highly scalable application to a swarm cluster.

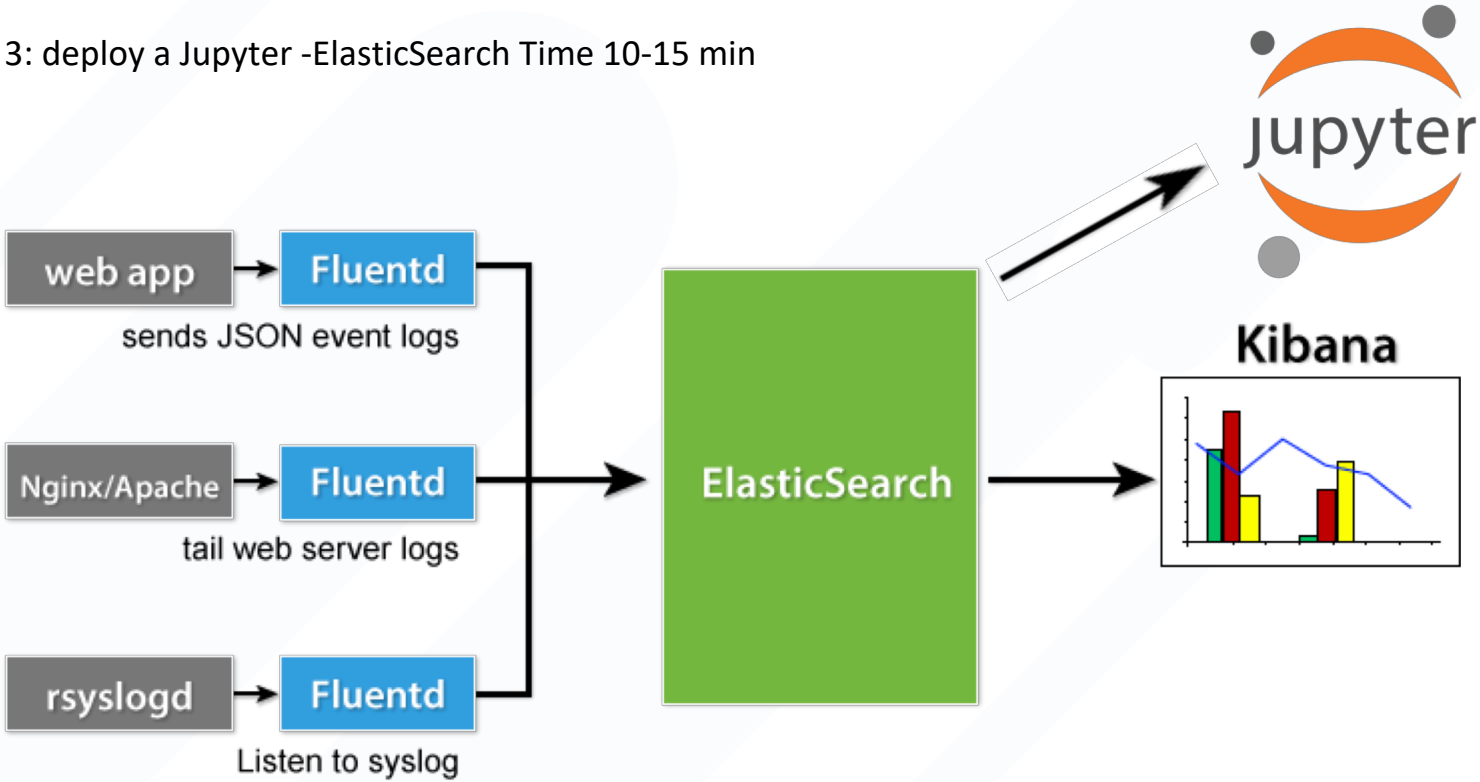
A text file create in Yaml Formal

Portable from desktop to swarm cluster

```yaml
services:
  minio1:
    image: minio/minio:RELEASE.2019-07-05T21-20-21Z
    hostname: minio1
    volumes:
      - minio1-data:/export
    ports:
      - "9001:9000"
    networks:
      - minio_distributed
    environment:
      MINIO_ACCESS_KEY: AKIAIOSFODNN7EXAMPLE
      MINIO_SECRET_KEY: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
    deploy:
      restart_policy:
        delay: 10s
        max_attempts: 10
        window: 60s
      resources:
        limits:
          cpus: "0.5"
          memory: 1024M
    command: server http://minio1/export http://minio2/export http://minio3/export
```

Docker Swarm Stack File

Exercise 3: deploy a Jupyter -ElasticSearch Time 10-15 min



Docker Swarm Orchestration

Exercise 4: A data Lake in production
minio.ipynb