paas-docs Documentation

Release 0.1

PaaS pizza team

January 11, 2018

This site describes the principles, implementations & how-to-use guides for **PaaS-Docker** initiative. If you do not find here your expected information, please contact the people referenced in *community* page.

Note: This documentation is a work in progress. Topics marked with a |stub-icon| are placeholders that have not been written yet. You can track the status of these topics through our public documentation issue tracker.

The PaaS offer relies on Docker Datacenter - DDC - solution. Docker Datacenter delivers an integrated platform for developers and IT operations to collaborate in the enterprise software supply chain. Bringing security, policy and controls to the application lifecycle without sacrificing any agility or application portability.

Docker Datacenter software consists of four two major components:

Universal Control Plane - UCP

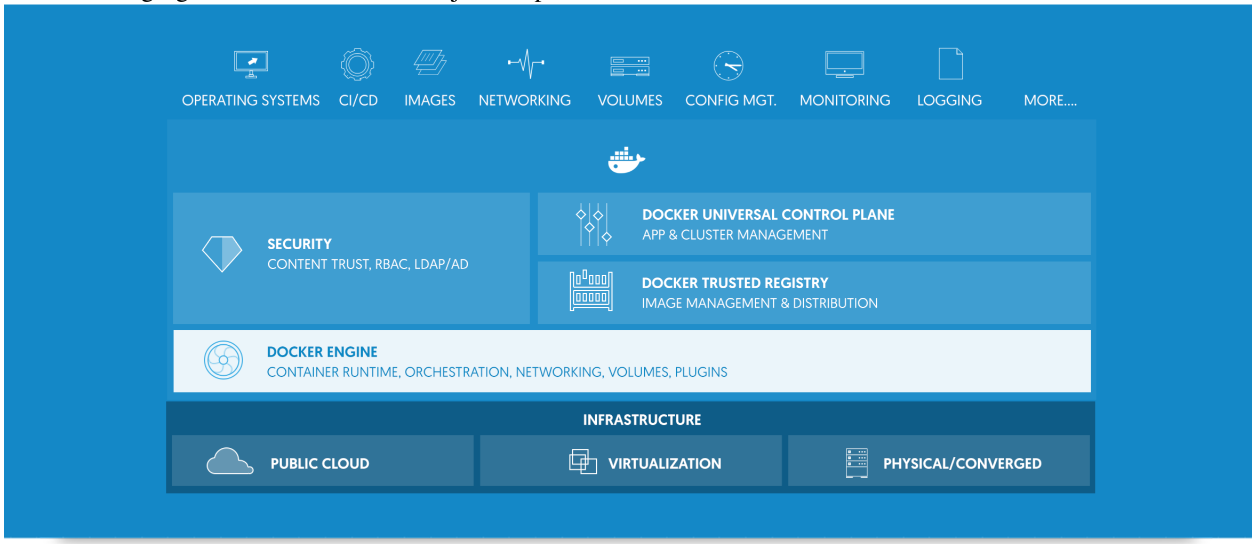
UCP enables teams to manage and deploy dockerized applications on top of our private SG cloud offer.

Docker Trusted Registry - DTR

Trusted Registry allows users to store and secure their images on-premises.

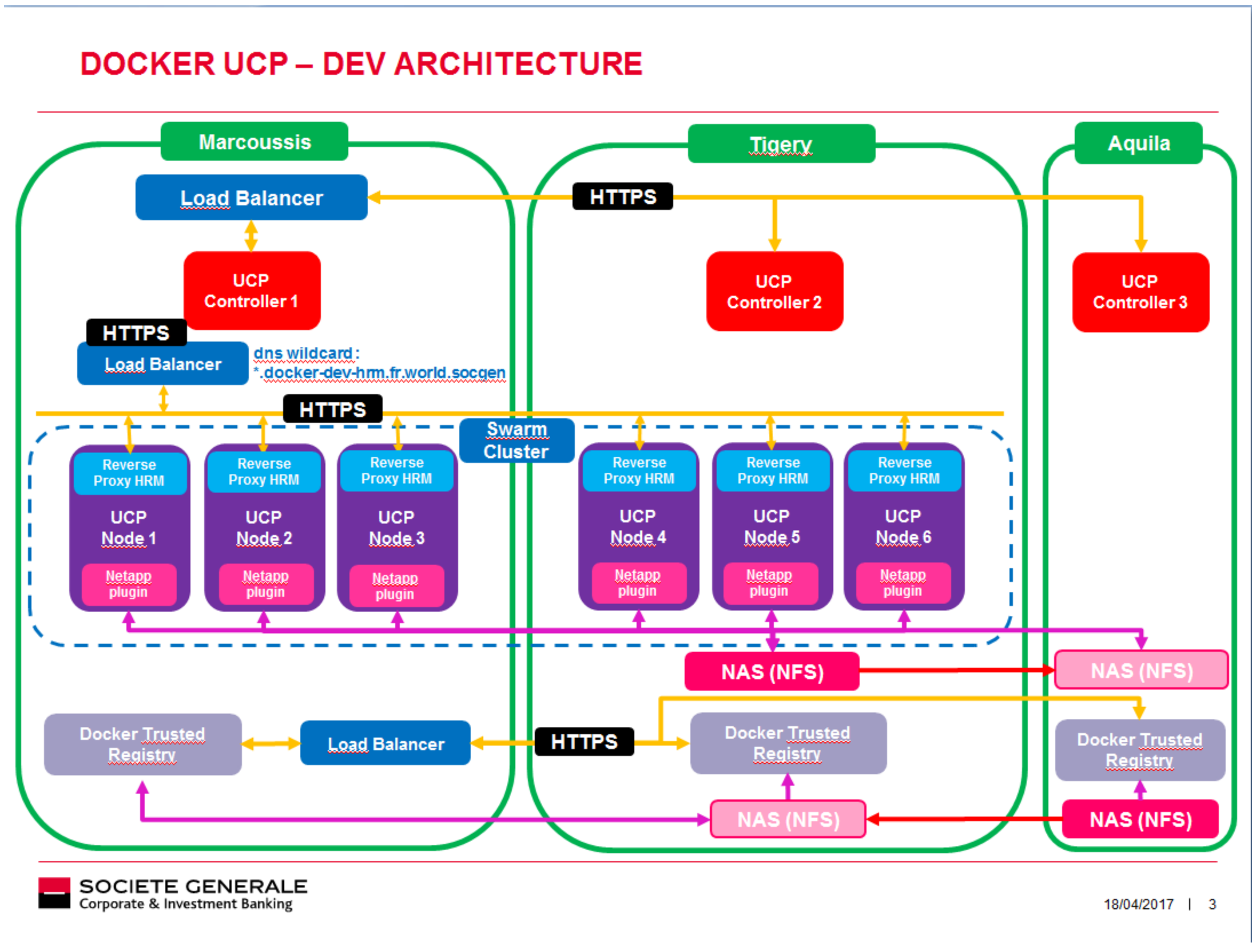
Note: Currently, one development environment is available and access is given to pilot applications validated by ITEC/ARC. The URLs are https://ucp-dev.fr.world.socgen & https://dtr-dev.fr.world.socgen.

The following figure illustrates DDC's major components and features.



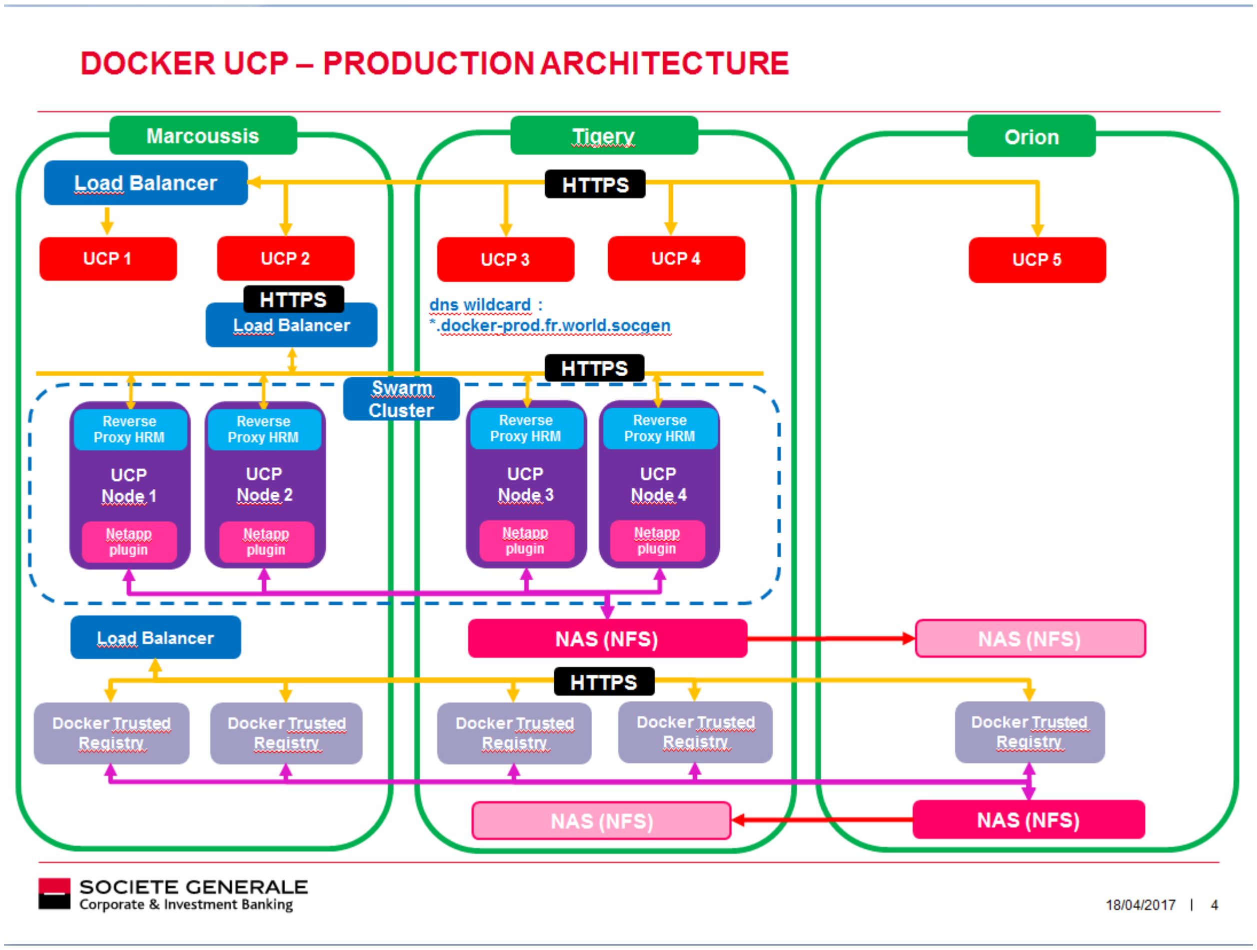
CHAPTER 1

Dev Architecture



CHAPTER 2

Prod Archictecture



CHAPTER 3

Docker on workstations

3.1 Purpose

Before deploying your application on the PaaS platform, you'll probably have to do some local developement especially to write and build your images. Since Docker is as of today only a Linux solution, the solution is to run a Linux VirtualBox VM on your Windows Desktop.

The repository with detailed implementation is here.

3.2 Installation

This section is about the client installation. In other words, when a end user wants a VM on its computer.

3.3 Prerequisites

Please, before installing any Virtual Machine, be sure to read the corresponding security standard [VM - Linux for developments].

If you wish to use Linux onto your desktop, first you need:

- A NDG desktop (Windows 7)
- AppsOnDemand
- To be member of one of the following groups:
  - For EUR
  - *EURFRSGCIB-FR-AOD-VIRTUALBOX-USR*
  - *EURFRSGCORP-FR-AOD-VIRTUALBOX-USR*
  - For ASI
  - *ASIHKSGCIB-HK-AOD-VIRTUALBOX-USR*
  - For AMER
  - *AMESGCIB-AME-AOD-VirtualBox-USR*

  As of summer 2017, **PAR-ITEC-ARC-PAAS-Expertise** manage these groups. You can send a mail to this group for access and validation of VirtualBox or PaaS platform

3.4 Install process

Once you satisfy the previous needs :

- Launch AppsOnDemand
- Install "VirtualBox"
- Click on the newly created shortcut *"Install a VM"* in your Windows start menu
- Follow the steps to customize your VM

3.5 How to access your VirtualBox VM webserver from your host machine

- Change the VirtualBox VM network setup from `NAT Network` to `NAT`
- In the `Advanced > Port Forwarding` section, add a route: - name: `http` - protocol: `TCP` - Host IP: *leave blank* - Host Port: choose a port to use on your host machine, for example `18080` - Guest IP: *leave blank* - Guest Port: the port to use inside your VM, for example' *8080*'
- Once that's done, you can run a webserver inside your VM on port `8080`, and access it from your host machine using `http://localhost:18080`

3.6 How to install Java on my Virtualbox VM

- Add the following into your `.bashrc` (don't forget to `source .bashrc` afterwards):

```
export JAVA_HOME=~/opt/java alias java=${JAVA_HOME}/bin/java
```

- Install Java in `~/opt/java` (since you are not `root` you can't install in `/opt/java`):

```
curl -k -sS -L "https://itbox-nexus-arc.fr.world.socgen/nexus-arc/content/repositories/sgc
| tar -C /opt/ -xz && mv /opt/jdk1.8.0_121 /opt/java && chmod
755 /opt/java/bin/* && chmod 755 /opt/java/jre/bin/* && ln -s
${JAVA_HOME}/bin/java /usr/bin/java
```

3.7 FAQ

1. I don't see "VirtualBox" in AppsOnDemand !

Please read again the prerequisites, you must be part of a special AD group to see it.

2. The *"Install a VM"* command fails

Try to unset your environment variables `HTTP_PROXY` and `HTTPS_PROXY`

CHAPTER 4

Processing your on-boarding

4.1 1. ITEC/ARC validation

Currently, one development environment is available and access is given to pilot applications validated by ITEC/ARC. Please contact ITEC-PaaS-Expertise team to ask for an access.

4.2 2. GTS/MDM/CTN ticket

Once ITEC has validated your application, please make a ticket in Service Now to PaaS team.

To do so, go to ServiceNow > Service Catalogs > Infrastructure Service Catalog > Middleware : PaaS > [MDM][DCK] Request access to PaaS Platform.

Make sure to have the following information to properly fill the ticket:

- your application trigram (**mandatory**)
- your iappli code (**mandatory**)
- your AD group (**mandatory**)
- 2 or 3 AD logins which will be admin of your DTR organization(**mandatory**)

CHAPTER 5

Self training plan

The internet is full of Docker ressources including the Official Docker documentation. In example :

- Introduction : https://docs.docker.com/engine/userguide/intro/
- Dockerfile : https://docs.docker.com/engine/reference/builder/
- Run reference : https://docs.docker.com/engine/reference/run/
- Stack : https://docs.docker.com/engine/reference/commandline/stack/
- Compose : https://docs.docker.com/compose/

5.1 Pluralsight

Pluralsight has very good video tutorials on Docker:

- Container Management using Docker training path
- Getting started with Docker Datacenter

5.2 Katacoda Docker courses

Katacoda describes itself as an Interactive Learning Platform for Software Engineers. It allows you to learn new technologies right in your browser. Some Labs you could find useful on your road to Docker:

1. Docker & Containers - 19 scenarios
2. Docker in Production - 8 scenarios

5.3 Docker Tutorials and Labs

Back in May, Docker itself launched the Docker Labs repo in an effort to provide the community with a central place to both learn from and contribute to Docker tutorials. There are now have 16 separate labs and tutorials, with 16 different contributors, both from Docker and from the community.

5.3.1 Docker tutorials - links to a different repository

- Docker for beginners
- Docker Swarm Mode
- Configuring developer tools
- Docker for ASP.NET and Windows containers
- Docker for Java Developers
- Dockerize a simple Node.js app
- Building a 12 Factor app with Docker

5.3.2 Community tutorials - links to a different repository

- Docker Tutorials from the Community
- Advanced Docker orchestration workshop

CHAPTER 6

Dockerfile Best Practices

6.1 Base images

Base images are made available by ARC/LAB here: https://sgithub.fr.world.socgen/DockerPOCClub/sgsharehub

You can also start from the GTS Red Hat base image, stored here: https://dtr-dev.fr.world.socgen/repositories/gtsmktssb/rhel7

6.2 Instructions

Below is a list of instruction usable in a Dockerfile. The one marked as "– **MANDATORY**" are not optionnal, you have to set them in your image or it won't be eligible to production.

6.2.1 ADD / COPY Instruction

The ADD instruction is supposed to be deprecated and COPY prefered. By the way, the ADD instruction could be useful as it uncompress/copy in the same time. So if your need is to simply copy files or directory in your image, you should use copy. And only use ADD for copy + uncompress operation.

6.2.2 CMD Instruction

You can use CMD instruction but only in addition to an ENTRYPOINT instruction (socgen). Multiple CMD is useless, only the last one is used.

6.2.3 ENTRYPOINT Instruction

You **MUST** use ENTRYPOINT and could use CMD in addition. Your entrypoint can be a script shell, which is pretty useful to set environnement variables or running conditions of you container. Entrypoint MUST NOT contains any "yum install" command.

6.2.4 FROM Instruction

Your base image should be an official socgen image. Your FROM instruction should address an "dtr-xxx" image.

6.2.5 HEALTHCHECK Instruction – MANDATORY

The HEALTHCHECK instruction tells Docker how to test that the container is still working. This can detect cases such as a web server that is stuck in an infinite loop and unable to handle new connections, even though the server process is still running. The HEALTHCHECK define at image level will be used by default is there is no HEALTHCHECK at the stack level. It will allow to display health at the container level only (docker ps will display healthy/unhealthy) and docker inspect will display the healthcheck parameter

The syntax is :

List of options :

- –interval=DURATION (default: 30s)
- –timeout=DURATION (default: 30s)
- –retries=N (default: 3)

6.2.6 LABEL Instruction – MANDATORY

The LABEL instruction replace the MAINTAINER one. It add metadata to an image, such as version maintainer, etc... You have to use it to pin the version of your image, and specify any maintainer informations.

```
LABEL "com.example.vendor"="ACME Incorporated"
LABEL version="1.0"
LABEL description="This text illustrates \
that label-values can span multiple lines."
```

As version pinning is frequently change, the LABEL instruction should be at the bottom of your Dockerfile.

6.2.7 MAINTAINER Instruction

MAINTAINER instruction is deprecated and you should use “LABEL” instruction instead. If you still want to use the MAINTAINER instruction, be aware : **DO NOT** use your specific name or phone. To avoid a full rebuild of the image if a new person manage your image later and want to replace your name. Prefer the name of your service.

6.2.8 PORT Instruction

DO NOT explicitly bind host port in Dockerfile.

6.2.9 RUN Instruction

Just an information about some cache issues. You can force a RUN command to be re-run (no cache used) by using the flag –no-cache in your “docker build” command.

6.2.10 SHELL Instruction

Docker provides a SHELL instruction which does not require overwriting /bin/sh in your container.

6.2.11 USER Instruction – MANDATORY

DO NOT run containers with root user. This could cause security breaks. You should use an application user to run your container, or create a "lambda user" in your Dockerfile, and run image with it.

6.2.12 WORKDIR Instruction

The "cd" should not be used if possible in a Dockerfile as it can create layer for nothing. Most commands can work using absolute paths and changing directory is in most of the case not necessary. And if you start using "cd X" "cd ../Y" you may need to think about all those changes if you change a path at any time for any reason.

If you need to change the workdir use the "WORKDIR" instruction, and use absolute path.

6.3 Practices

6.3.1 Basics

When you write a Dockerfile you want to keep 3 things is mind :

- **My image need to be as light as possible.** This way your image will run, pull, push faster. Use rm to suppress all temporary files. Suppress all unecessary packages. yum clean after a yum install. Etc.. Make your images as light as possible.
- **I want to use the cache as much as possible.** This way your build is faster. Static fields should be at the top of your Dockerfile (such as basic user, dir creation..). And the field will change frequently, or won't be able to use cache at the bottom.

That way you'll tune the usage of the cache and avoid layer creation on each build. - **I want as less layer as possible**. Don't stack 10 "RUN" instructions, and prefer create only one :

6.3.2 sudo vs gosu

The sudo binary should not be used in an image. Prefer "gosu" to enforce root. The difference is that gosu avoids sudo's "strange and often annoying TTY and signal-forwarding behavior".

6.3.3 yum

- update

If you have to use "yum update" don't leave that command alone in a single line in the Dockerfile. Use update instructions along with install instructions and version pinning for packages while installing them. This would bust the cache and force to extract the required versions.

- install

You have to pin your versions explicitly. This will force the build to retrieve a dedicated version regardless of what's in the cache. This will be useful for the "update" instruction too.

- clean

Always clean temporary files after install a package.

```
RUN yum update \
 && yum -y install tar \
 && COMMAND1 \
 && COMMAND2 \
 && COMMANDXXXXX \
 && yum clean all
```

6.3.4 image tag : latest vs pinned

You should not use tag “latest” as it can cause confusion or build problems. Always pin your image version.

6.3.5 .dockerignore

A file .dockerignore can be used (same principle than a .gitignore) to reduce the context load in memory while you build an image. Just list in there all file that aren’t necessary in the build process. The .dockerignore has to be placed in the Dockerfile’s directory.

6.3.6 secret

Your secrets should not be stored in the version control; but should be in the UCP docker secret or in the Vault. **Avoid** secret in environmental variables cause it can be seen by “docker inspect”, or in a error log stack.

6.3.7 Copy Dockerfile into /

Even if the “docker history” can give many information about the build of an image, copying your Dockerfile in the / of your image if a best practice.

6.4 Template

Here is an example of a “basic” Dockerfile

```
FROM dtr-xxx.fr.world.socgen/xxxxxx/image:tag

# Create a lambda user
RUN mkdir -p /home/webdck01 && groupadd -g 996 -r webadm && useradd -u 997 -r -g webadm -d /home/webd
       && chmod 755 /home/webdck01 && chown webdck01:webadm /home/webdck01

WORKDIR /home/webdck01

# A variable for current version
ARG version

# Don't run container as root
USER webdck01

# Copy your entrypoint.sh
COPY files/entrypoint.sh /entrypoint.sh

# Set entrypoint.sh
ENTRYPOINT ["/entrypoint.sh"]
```

```
PORT 8080

# Set an healthcheck
HEALTHCHECK --interval=30s --timeout=3s \
  CMD curl -f http://localhost:8000/ || exit 1

# Set your label
LABEL "Maintainer"="My Service"
LABEL version="$version"
```

CHAPTER 7

My_stack.yml Best Practices

7.1 Template

7.1.1 docker-compose.yml example

```
version: "3.1"
networks:
  my-hrm-network:
    external: true
  internal-network:
    driver: overlay
    labels:
      com.docker.ucp.access.label: "my-group-label"


services:

  myservice:
    image: "dtr-dev.fr.world.socgen/my-repo/my-image:TAG"
    deploy:
      replicas: 2
      labels:
        com.docker.ucp.access.label: "my-group-label"
        com.docker.ucp.access.owner: "stack-owner"
        com.docker.ucp.mesh.http.XX: "internal_port=XX,external_route=sni://my-service-dev-hrm.fr.wor
      resources:
        limits:
          cpus: '0.01'
          memory: 50M
      update_config:
        parallelism: 1
        delay: 10s
      restart_policy:
        condition: on-failure
        delay: 5s
        max_attempts: 3
        window: 120s
    healthcheck:
      test: ["CMD", "curl", "--fail", "--insecure", "https://localhost/health"]
      interval: 30s
      timeout: 2s
      retries: 3
    networks:
      - my-hrm-network
      - internal-network
    ports:
      - 80
      - 443
    environment:
      - ENVNAME=$ENVNAME
      - ENVTYPE=$ENVTYPE
    secrets:
      - source: my-secret-cert-in-ucp
        target: certificate.pem
        mode: 0444
        uid: '1000'
        gid: '1000'

secrets:
  my-secret-cert-in-ucp:
    external: true
```

7.2 Mandatory parameters

- **replicas** :

You have to set replicas at 2 minimum to ensure load balancing, and availability in production.

- **limits** :

You have to set cpu/mem limits on each of your containers, to avoid resource overload. cpu limits is the number of core. cpus: "1" means you'll take 1 cpu. Our VM got 8 cpu, so you SHOULD NOT reserve 1 cpu as many containers are running on VM you'll won't find any dedicated CPU.

- **parallelism** :

This should be set to 1 to ensure your stack will be updated by a rolling update policy.

- **reservations** :

You may need to set a resource reservation if you need a minimum number of resources to run your application (by example a java -Xms). This parameter has to be use wisely, and application who need a large amount of reserved resource to run, could be the subject of a study.

Warning: For now, as we can't dedicate a node to a specific stack, we don't want reservation to be used.

- **port** :

Do not bind internal port on host. This syntaxe is prohibited :

ports:

- 80:80
- 443:8443

- **label** :

You have to set com.docker.ucp.access.label to ensure only your group can operate with your stack. For now you have to set com.docker.ucp.access.owner, to make you able to deploy a stack that is already running (update)

- **max_attempts** :

You have to set a “max_attempts” for your stack. That way, if it fails, your stack won’t try to reload for ever and my impact the cluster.

- **healthcheck** :

Healthchecks have to be set in Dockerfile and in your yml stack file. If you don’t specify any healthcheck at the stack level the healthcheck from the image (define in Dockerfile) will be used. But it’s better to define it also in your stack file which allow you to override default value (from the image). Using healthcheck at the stack level will also display healthcheck at both levels (service using docker service inspect and container using docker inspect) and not only at the container level.

7.3 Practices

- Default envfile :

You can use a default envfile by creating a file named ”.env” in the same directory as the docker-compose.yml file. It will be automatically load by the docker-compose.

- Debug your image :

A good way to debug a container that can’t start is to overwrite the entrypoint using docker-compose.yml. Simply add :

```
entrypoint: tail -f /etc/hosts
```

This way, you’ll be able to connect to your running image, and test the failures scripts without having your container going down.

CHAPTER 8

Build Best Practices

8.1 TAG

You have to tag your image using your git commit ID.

This is MANDATORY. This will avoid any conflict with an existing image during the build, as you'll never known where your build container is going to be populated on the cluster.

8.2 CACHE USAGE

You should never used the –no-cache to build your image.

THIS IS MANDATORY. Your build environments are mutualized you need to be aware and responsible of your storage. If you encountered something weird during your build, you can temporary set the –no-cache and try. But it should never be the default configuration.

Warning: This is even more impacting currently, as the jenkins plugin builds on DTR and manager nodes.

8.3 AGNOSTIC IMAGES

Image should be agnostic of environments

This is MANDATORY. Your image should be build without any environment configuration inside. The environment configuration should be injected during the run. This will avoid any issue during the deployment process on production. The image tested on DEV are the same on PROD, there is no build process on the prod infrastructure.

The full process is :

- build agnostic image on DEV.
- run release image injecting your DEV configuration on the DEV cluster.
- test and validate this image.
- retag your image as "prd" and push it in the prod DTR. (done by PAAS team)
- deploy your image on production injecting prod configuration during the run.

CHAPTER 9

Remember of ITEC change management

Below are some documents how to manage production change:

- GBIS Change Management VISA for GTS, ITEC, BSC: https://sbc.safe.socgen/docs/DOC-27880
- Chg MGMT Process A3: https://sbc.safe.socgen/docs/DOC-68591
- Change Management in Impulse workflow: https://sbc.safe.socgen/docs/DOC-68592

Path to production and TOM

TOM is for Target Operating Model.

The main principles are:

1. No build on production platform: idempotence of image = same image everywhere.
2. As platform is mutualized, deployment in production platform can be done only with a specific GTS production Jenkins pipeline that will:
    (a) check docker-compose.yml file compliance
    (b) extract the needed image(s) for each service
    (c) check the compliance of each image:
      - root not used in container
      - by running privesc audit script in a container started with the image
      - by checking CVE
    (a) forbid the deployment if any compliance check fails
    (b) otherwise promote images and deploy the stack
4. Report every image push and start process to comply with SAFE requirements.

Although they are not finalized, you may find in https://sbc.safe.socgen/docs/DOC-83435 these guidelines and operating model.

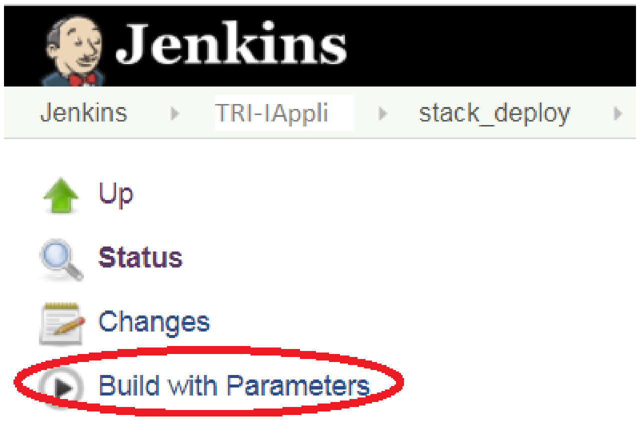
How to deploy mystack?

At first, all prerequisite must be fulfilled like:

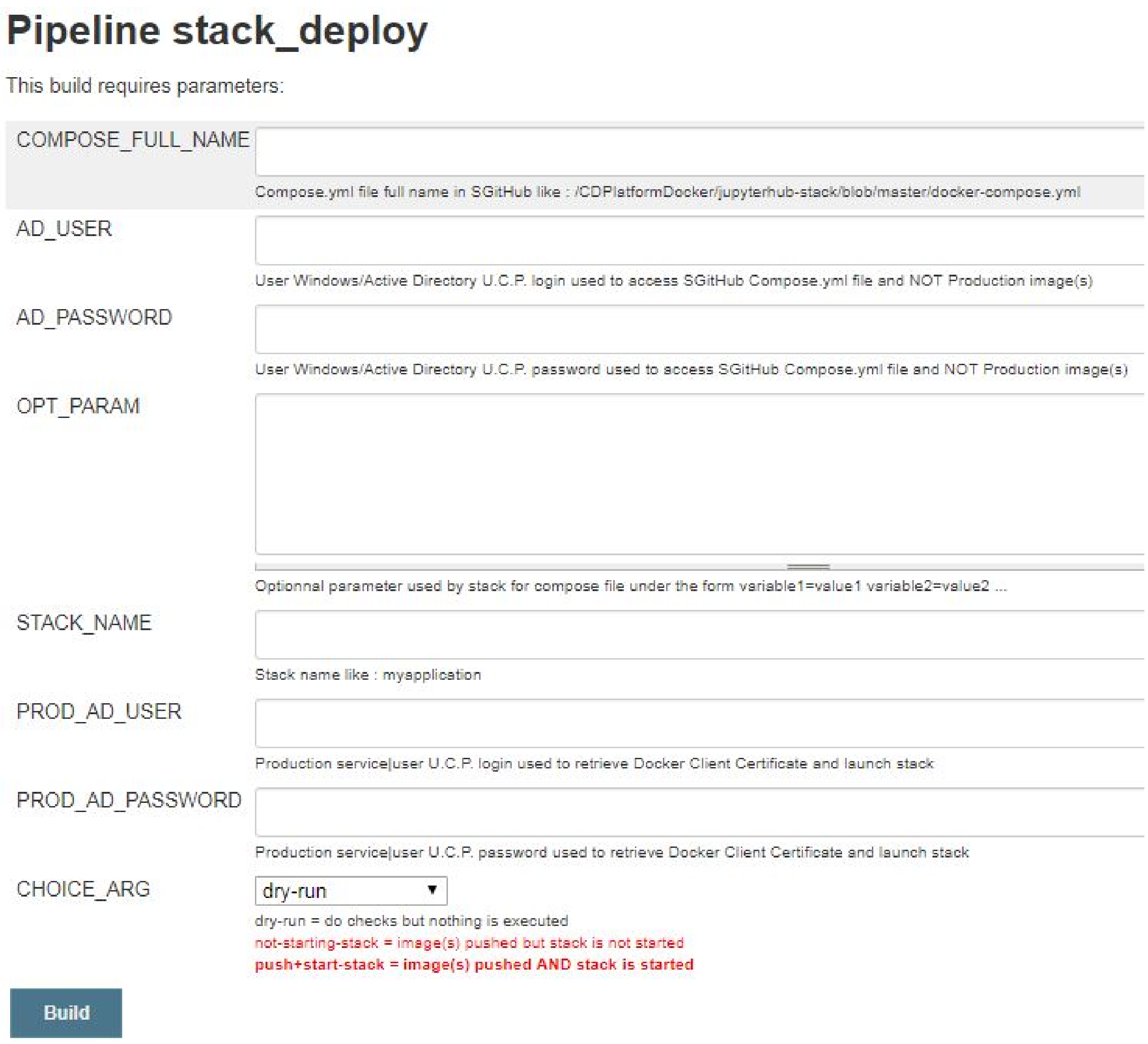
- Security Scanning of the NOT production image(s) detailled in http://rtd.fr.world.socgen/docs/paas-docker/en/latest/DDC/use-it/DTR/DSS.html
- compose-file of the stack must be available in https://sgithub.fr.world.socgen/... and read accessible with dck-prd-svc A.D. account known as dck_prd_svc in SGitHub
- production U.C.P. access, HRM network creation, ... : granted by doing an Impulse request as indicated in http://rtd.fr.world.socgen/docs/paas-docker/en/latest/DDC/on-Boarding/process.html, but this time choosing for Impacted environment: **Production** and Environment: **PROD**

After login in https://jenkins-dck.docker-prod.fr.world.socgen/ open your TRI-IAppli folder to be in https://jenkins-dck.docker-prod.fr.world.socgen/job/TRI-IAppli/

Then launch Build with Parameters the stack_deploy job:



And fill all mandatory parameters, before clicking Build button :



Pipeline stack_deploy

This build requires parameters:

COMPOSE_FULL_NAME	Compose.yml file full name in SGitHub like : /CDPlatformDocker/jupyterhub-stack/blob/master/docker-compose.yml
AD_USER	User Windows/Active Directory U.C.P. login used to access SGitHub Compose.yml file and NOT Production image(s)
AD_PASSWORD	User Windows/Active Directory U.C.P. password used to access SGitHub Compose.yml file and NOT Production image(s)
OPT_PARAM	Optionnal parameter used by stack for compose file under the form variable1=value1 variable2=value2 ...
STACK_NAME	Stack name like : myapplication
PROD_AD_USER	Production service|user U.C.P. login used to retrieve Docker Client Certificate and launch stack
PROD_AD_PASSWORD	Production service|user U.C.P. password used to retrieve Docker Client Certificate and launch stack
CHOICE_ARG	dry-run dry-run = do checks but nothing is executed not-starting-stack = image(s) pushed but stack is not started push+start-stack = image(s) pushed AND stack is started

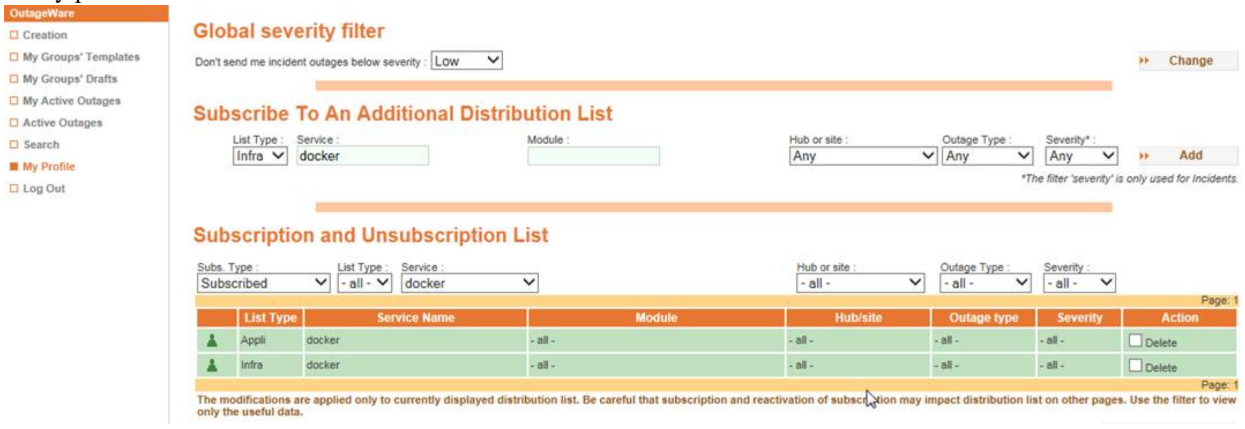
Warning: As indicated in http://rtd.fr.world.socgen/docs/paas-docker/en/latest/DDC/on-Boarding/Build%20best%20practices.html each new image must have a new tag. Thus compliance will fail if you try to deploy in production a different image with the same tag than an already existing image in production with this tag. If you still wish to keep the same tag, the single solution will be to stop the stack, ask in writing to PaaS team to remove the production image.

CHAPTER 12

How subscribe to Docker outages?

Docker team send an outage for both non-production and production platform when relevant (for example: releases, instabilities. . .).

You can directly subscribe to these outage via outageware https://outageware.fr.world.socgen/index.php?page=Profile in My profile section:



Global severity filter

Don't send me incident outages below severity : Low

Subscribe To An Additional Distribution List

List Type : Infra
Service : docker
Module :
Hub or site : Any
Outage Type : Any
Severity* : Any

*The filter 'severity' is only used for Incidents.

Subscription and Unsubscription List

Subs. Type : Subscribed
List Type : - all -
Service : docker
Hub or site : - all -
Outage Type : - all -
Severity : - all -

Page: 1

	List Type	Service Name	Module	Hub/site	Outage type	Severity	Action
	Appli	docker	- all -	- all -	- all -	- all -	☐ Delete
	Infra	docker	- all -	- all -	- all -	- all -	☐ Delete

Page: 1

The modifications are applied only to currently displayed distribution list. Be careful that subscription and reactivation of subscription may impact distribution list on other pages. Use the filter to view only the useful data.

CHAPTER 13

Connect to UCP UI

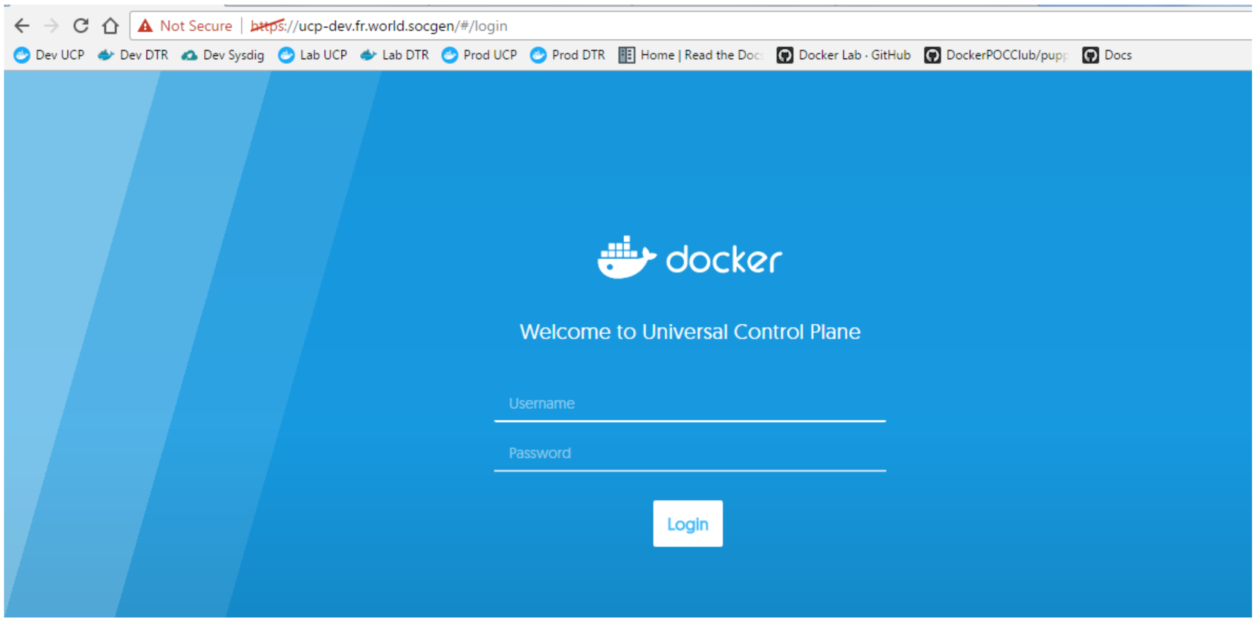
13.1 Connection

To be able to connect to UCP, your access need to be granted by the **GTS team**. Just send them your ldap account name and AD, and you should be able to launch a navigator and connect to UCP.

> Dev UCP URL : https://ucp-dev.fr.world.socgen/
>
> Prod UCP URL : https://ucp-prod.fr.world.socgen/

UCP is an interfaces that gives you ability to operate your containers or your services in a docker swarm cluster environnement.

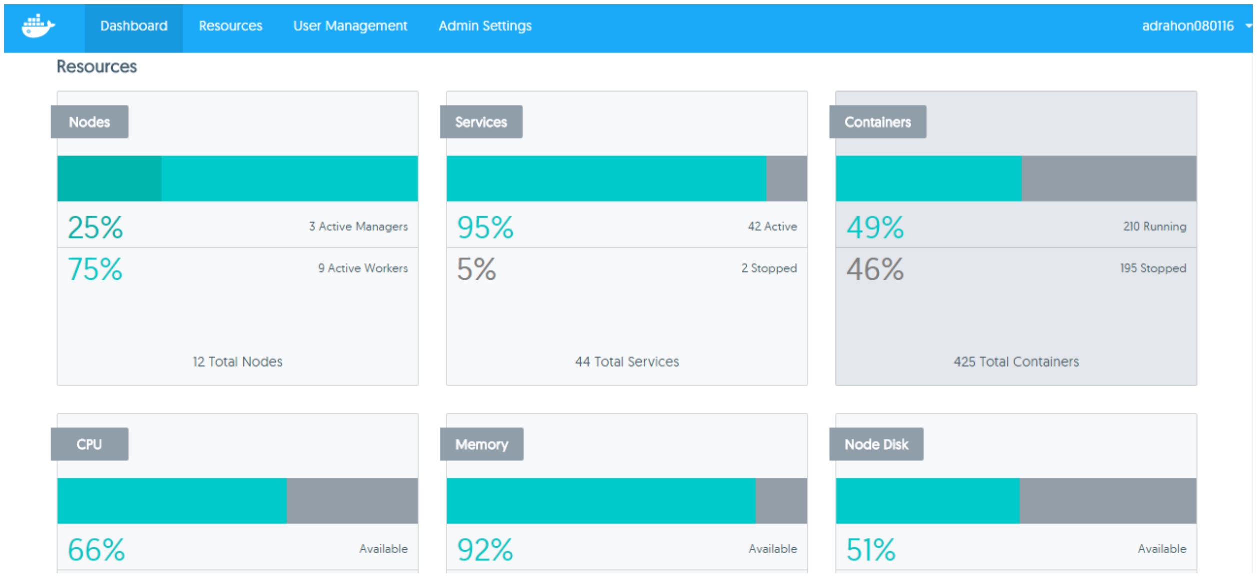
You'll find differents tabs at the top of the interface : Dashboard, Resources, User management(only admins), Admin setting (only admins), and in the top right end corner the classical menu for your profile management and logout.



13.2 Dashboard Tab

Dashboard will gives you an overview of your environment utilisation such as :

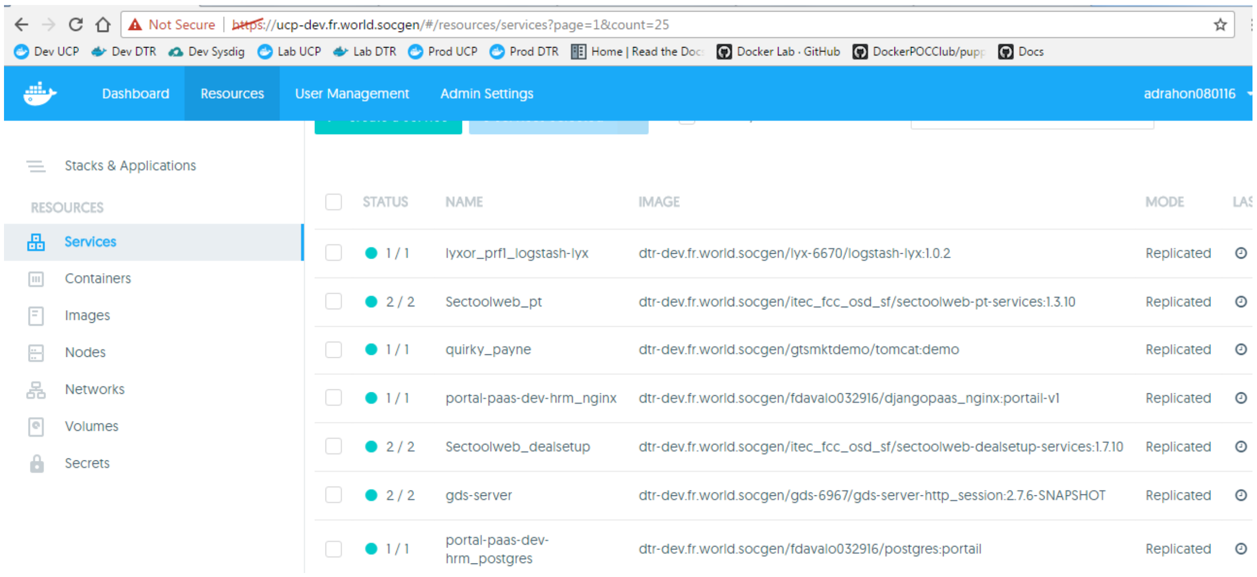
- actives nodes,
- services states,
- global cpu utilization,
- disk usage,
- etc.



13.3 Resource Tab

Resources will gives you many tools to operate your services and containers. Some possibilities :

- list all your applications and services related.
- list attributes of your services such as image, network, volumes.
- list all network, image, related to your application.
- Create new task, service, container, network, secret, etc...
- Scale your services up or down.
- State of your application, nodes.
- etc.



STATUS	NAME	IMAGE	MODE
1 / 1	lyxor_prf1_logstash-lyx	dtr-dev.fr.world.socgen/lyx-6670/logstash-lyx:1.0.2	Replicated
2 / 2	Sectoolweb_pt	dtr-dev.fr.world.socgen/itec_fcc_osd_sf/sectoolweb-pt-services:1.3.10	Replicated
1 / 1	quirky_payne	dtr-dev.fr.world.socgen/gtsmktdemo/tomcat:demo	Replicated
1 / 1	portal-paas-dev-hrm_nginx	dtr-dev.fr.world.socgen/fdavalo032916/djangopaas_nginx:portail-v1	Replicated
2 / 2	Sectoolweb_dealsetup	dtr-dev.fr.world.socgen/itec_fcc_osd_sf/sectoolweb-dealsetup-services:1.7.10	Replicated
2 / 2	gds-server	dtr-dev.fr.world.socgen/gds-6967/gds-server-http_session:2.7.6-SNAPSHOT	Replicated
1 / 1	portal-paas-dev-hrm_postgres	dtr-dev.fr.world.socgen/fdavalo032916/postgres:portail	Replicated

Simply clic on your stack/services to navigate into menu and see any usable options.

Warning: If you want your change to be persistent, you should do it manually via a docker-compose.yml file. Using the interface can chance the state of your services, but it won't be persistent.

CHAPTER 14

Deploy a Stack

14.1 docker-compose.yml

Your images are builded and pushed into the DTR. Now you want to deploy your images on the cluster as a stack of services. The easiest way to do it, is to use a file (called docker-compose.yml) that will describe your full stack of service, and load it into UCP.

After a deploy your containers start over the cluster and communicate.

Here are some informations you should set in the docker-compose file :

- docker-compose version (3 minimum to have deploy swarm options)
- name of your network.
- name of your services.
- label associated with your services.
- name of you containers.
- images to run.
- environnement variables.
- volumes.
- networks.
- secrets.
- deployment policy.
- etc.

Version 3 of the docker-compose.yml is required to be able to deploy a stack of services. If you need the version 2, you won't be able to deploy a service on the cluster, but only deploy containers on a single node.

Here is an example of a stack of services composed of an "api-gateway" and a "consul" services. Deploying those services will create 4 containers (replicas 2 for both services) on a fpl-hrm-network. Only team with the "fpl_dev_label" label associated to their group will be able to operate this services.

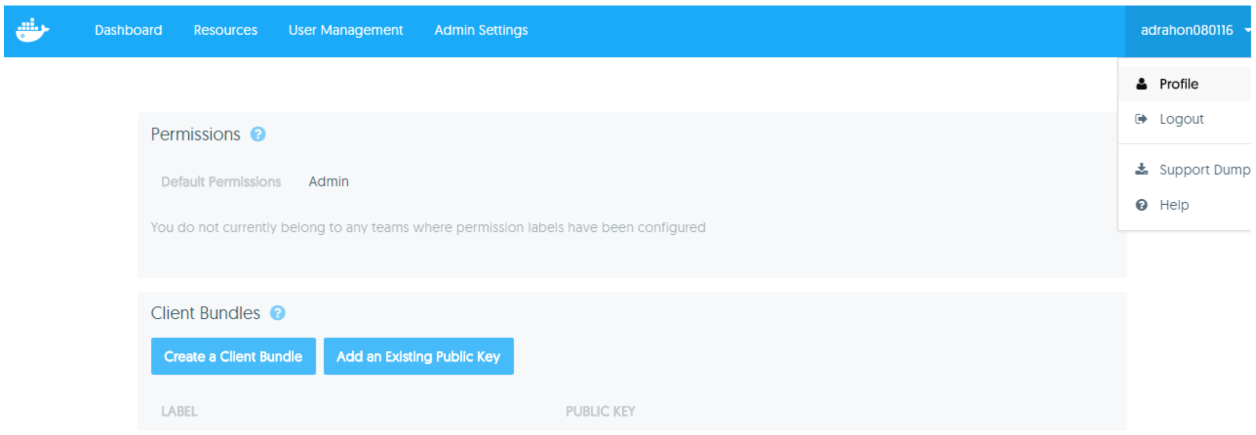
Once again, note the "version 3.1" at the top, to be able to use the "deploy" options in your docker-compose.

```
version: "3.1"
networks:
  app-hrm-network:
    external: true
  nw-app-v4:
    driver: overlay

services:
  app-gateway:
    image: "dtr-dev.fr.world.socgen/fpl-95595/app-gateway"
    networks:
      - app-hrm-network
      - nw-app-v4
    deploy:
      replicas: 2
      labels:
        com.docker.ucp.access.label: "app-devteam"
    labels:
      com.docker.ucp.mesh.http.443: internal_port=443,external_route=sni://app-dev.docker-dev-hrm.fr
    ports:
      - 443
    env_file:
      - ./credentials.env
      - ./environment.env
    environment:
      - DOCKER_HOST=$DOCKER_HOST
      - DOCKER_TLS_VERIFY=$DOCKER_TLS_VERIFY
      - ogn_proxy_server_url=https://app-gateway

consul:
  image: "dtr-dev.fr.world.socgen/xxx-95595/consul"
    deploy:
      replicas: 2
      labels:
        com.docker.ucp.mesh.http.443: internal_port=443,external_route=sni://app-dev.docker-dev-hrm.
    ports:
      - 8500
    networks:
      - nw-app-v4
    command: agent -server -ui -bind=127.0.0.1 -client=0.0.0.0 -data-dir=./data -node=consul -bootst
```

14.2 Bundle

As you should use your vitual Box VM to build your images, you should use it too to operate with UCP. You should never try to connect to UCP manager nodes neither on worker nodes using ssh.

To operate with UCP, a ucp bundle is downloadable from the UI. To download this bundle, top right clic on : your name -> profile and then clic on “Create a Client Bundle”.



Permissions

Default Permissions Admin

You do not currently belong to any teams where permission labels have been configured

Client Bundles

This will download a zip, which you should unzip in one of your VM directory. Just source the .env, and you should be able to operate with the target UCP as a client.

```
source env.sh
```

14.2.1 Label your service

When you deploy a stack using the UCP, the label of your user is automatically assign to your service. That way only you can "see" your services and containers on the cluster.

**Therefore, if you want your team to be able to operate your stack, you have to set a "com.docker.ucp.access.label" in your docker-compose.yml file. **

That label should reference at least the Trigram of your application, and the environnement where the service is running, or the team operating your service. By example it can be of the form "ApplicationTrigram_devteam", in our case :

```
labels:
    com.docker.ucp.access.label: "app-devteam"
```

A label should be associated to your team at the creation. If it's not the case, or if you have lost your label, ask to the GTS team (PAR-RESG-GTS-DOCKER).

You can also check the label associated to your service by going : Resources -> Services -> your_services -> Permission label.

14.3 Deploy your stack

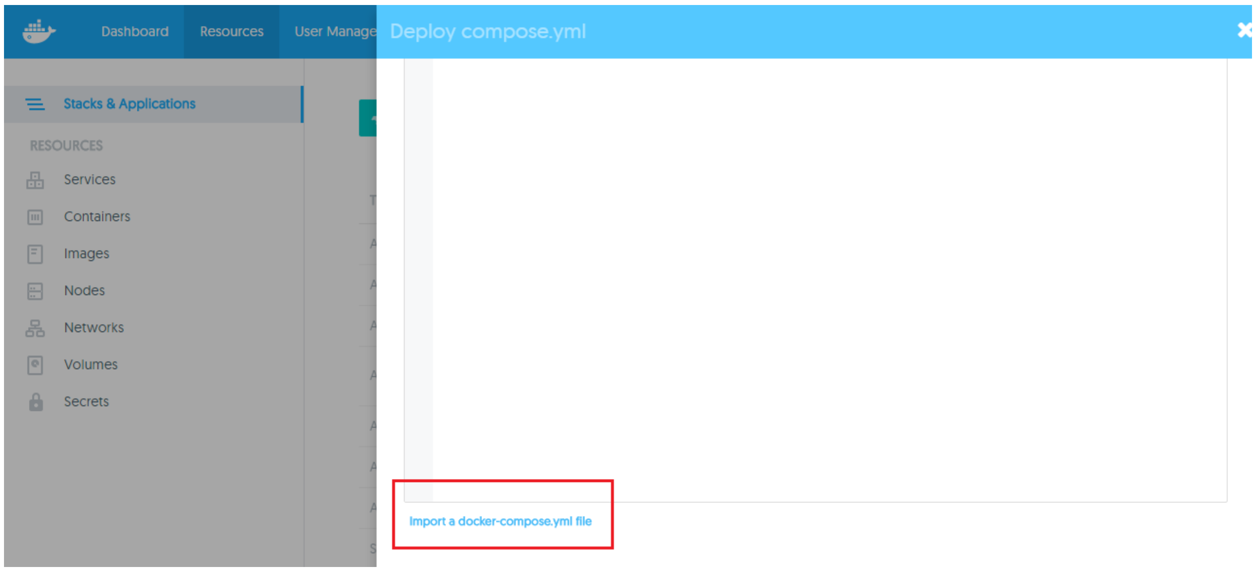
There is 2 differents way to deploy your stack from your docker-compose.yml. You can do it manually using the ucp bundle, or from the UCP interface.

If you want do it manually, you'll need to download your bundle on your Vbox/Docker4windows (see above) and set your environment by sourcing the env.sh file. Then you'll be connected to UCP with your own credential and able to launch your stack using a classical compose file.

Then place yourself in the docker-compose.yml directory and simply type :

```
docker stack deploy -c docker-compose.yml stack_name
```

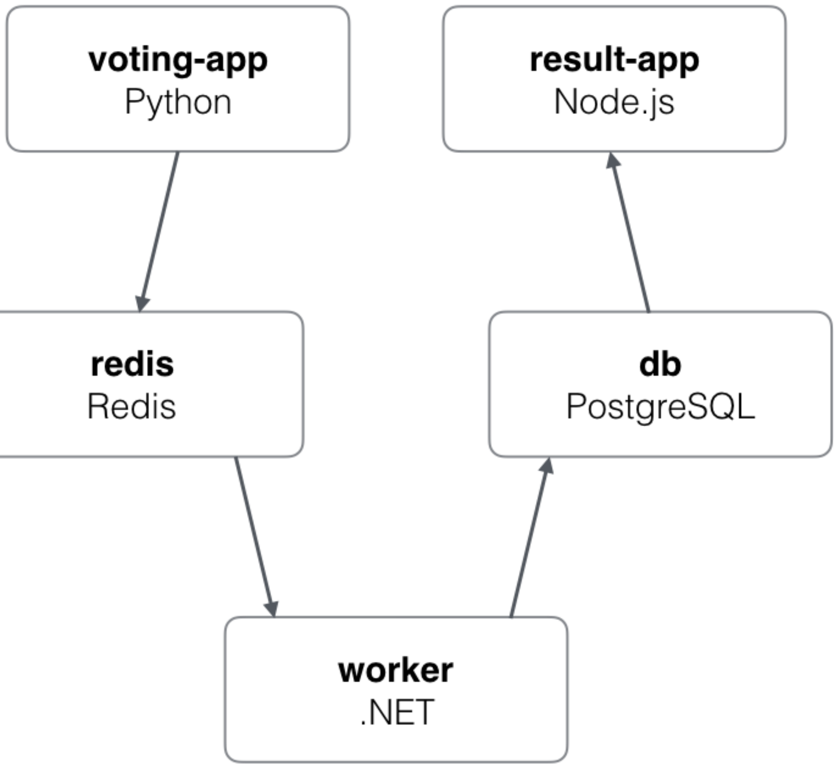
If you want to do it from the UCP interfaces, connect to UCP url and then resources -> deploy. You can now simply paste your docker-compose.yml or use the "import docker-compose.yml" button and select a docker-compose.yml on your computer.



By the way, using the UCP interface to deploy your stack is not a best practice, because the docker-compose is not persisted. Your docker-compose.yml files should be stored in github and versionned in there.

14.3.1 Deploy an example application

UCP natively provides you a load balancing facility, a discovery backend and a clustered pool of compute ressources. So now, you can deploy your first application: the voting application itself. You do this by starting a number of "Dockerized applications" running in containers. Here is the architecture of the application:

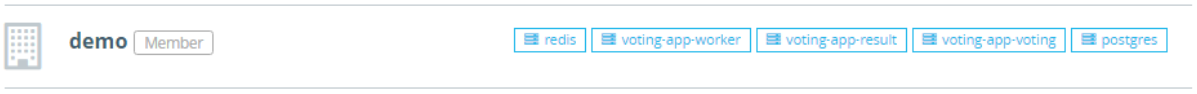


It is composed of:

- A Python webapp which lets you vote between two options
- A Redis queue which collects new votes
- A .NET worker which consumes votes and stores them in
- A Postgres database backed by a Docker volume
- A Node.js webapp which shows the results of the voting in real time

Besides, all containers will be connected using a docker overlay network: voteapp. The voteapp network is available to all Docker hosts using UCP discovery backend.

1. You can find the public repository of the voting-app on sgithub_voting_app
2. All the docker images required for this appliation are available in the DTR, in public repositories under the demo organization:



3. Get locally the sgithub_compose_file file & launch it aka run a docker-compose up in the same folder of the docker compose file after having set your bundle

CHAPTER 15

Stack Management

15.1 Manage a stack of services

First thing to know is there is always 2 ways to do something : manually using the UCP bundle from your Virtual Box VM or by using the UCP interface. “Manually” means using a docker-compose.yml to interact with UCP (see the “Launch a Stack topic for details).

If you want to change a parameter in your stack and want this change to be persistent (for example : scaling your service on 3 nodes instead of 2), you should always use the manual method and update your docker-compose.yml file.

Scaling a service by the interface won’t be persistent, and the next restart of your service from the docker-compose file could lead to lose your changes.

15.1.1 Help commands

Any command has an “help” section, simply add a –help to your command. For example :

```
docker --help
docker ls --help
docker stack --help
docker service --help
```

15.1.2 list/stop/deploy commands

Here are some useful commands, you need to be in your docker-compose.yml directory, and set the docker-compose.yml to version 3 to have full access to every deploy option :

- **List** your running stacks

```
docker stack ls
```

- **Deploy** your stack of services from your docker-compose.yml, this will start every services describe in the file.

```
docker stack deploy -c docker-compose.yml my_stack
```

- **Delete** your stack, this will remove every services described in your file.

```
docker stack rm my_stack
```

- **List services** in your stack (my_stack)

```
docker stack ps my_stack
```

- **Update a service** (any change to your docker-compose.yml file should be loaded)

  docker service update my_service (options)

- **Scale you service** to 4 containers (if you want it to be persistent, update your docker-compose.yml and reload your service)

```
docker service scale your_service=4
```

Warning:
**A best practice is to use the access.owner with a software account (which is trully exists) inside your docker-compose.yml fil

Add the following line to your compose file in addition to your access.label :

```
- "com.docker.ucp.access.label=app-dev-label"
- "com.docker.ucp.access.owner=your-software-account"
```

15.1.3 Inspect a service, container, network etc..

It could be useful to debug or get information to inspect your service or your container. By example if you want to know what port is exposed by swarm for your service? Or see what environmental variables are setted in your container?

```
docker inspect container_name
docker network inspect network_name
docker service inspect --pretty service_name
```

You can also use the container or network id, but this is ain't a best practice. We want to know what people do, and id aren't simple to read.

Note: the –pretty flag avoid the json format output.

15.2 Use UCP to manage your stack

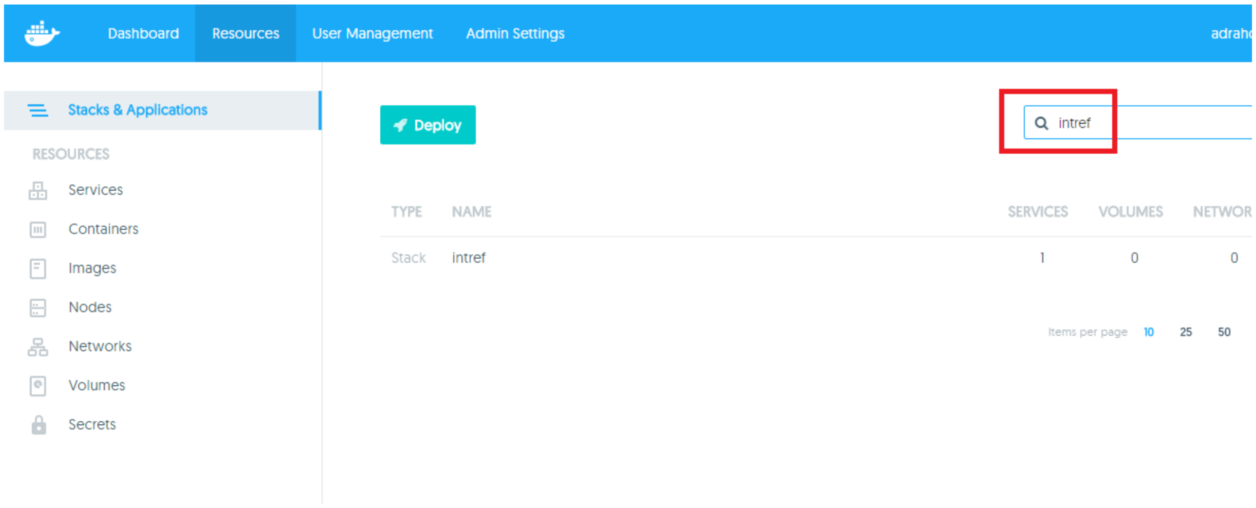
Again, any persistent change on your stack should be done by updating your docker-compose.yml file manually and using it. Any change by UCP won't be persistent.

You can still : stop, run, update, scale, monitore your stack and services by the UCP interface.

Connect to UCP and click on "Stack and application". Now you just have to click to your stack (or use the search menu top right to find your stack) and click on it to access to all services and options.

Here is an example of how to scale a service by using UCP interface :

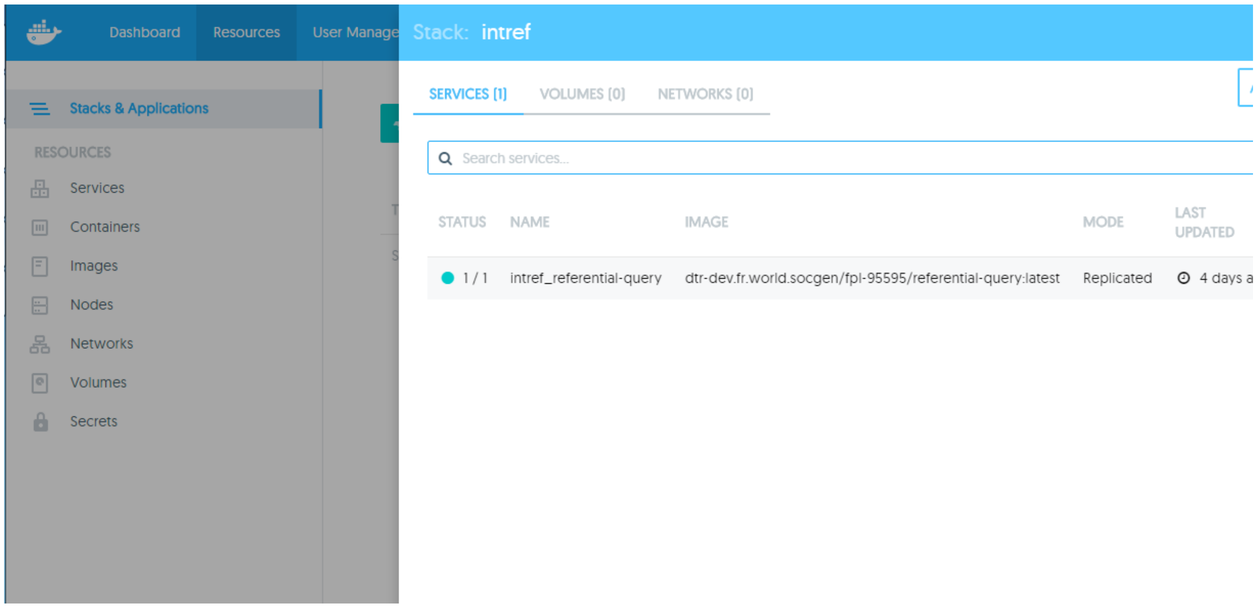
Connect to UCP and clic on "Stack & application" to search for your stack



TYPE	NAME	SERVICES	VOLUMES	NETWOR
Stack	intref	1	0	0

Items per page 10 25 50

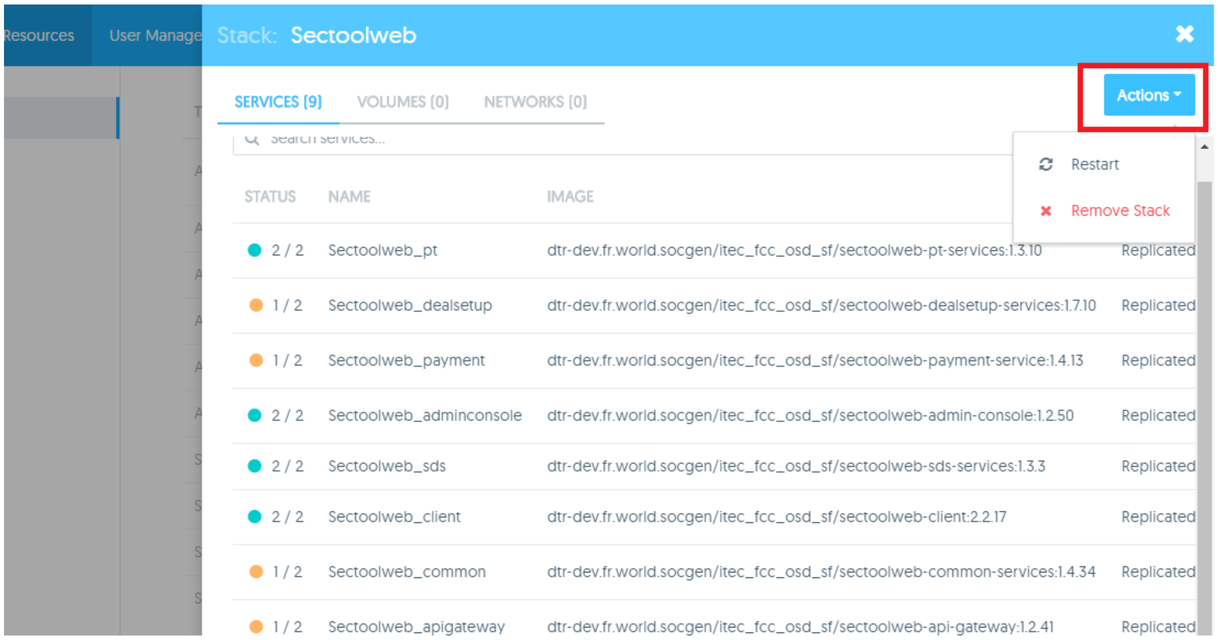
Clic on your stack to see services related to



STATUS	NAME	IMAGE	MODE	LAST UPDATED
1 / 1	intref_referential-query	dtr-dev.fr.world.socgen/fpl-95595/referential-query:latest	Replicated	4 days a

15.2.1 Restart, Remove a full stack of services

Select your stack in the “stack and application” menu, then clic on the top right “action” button.

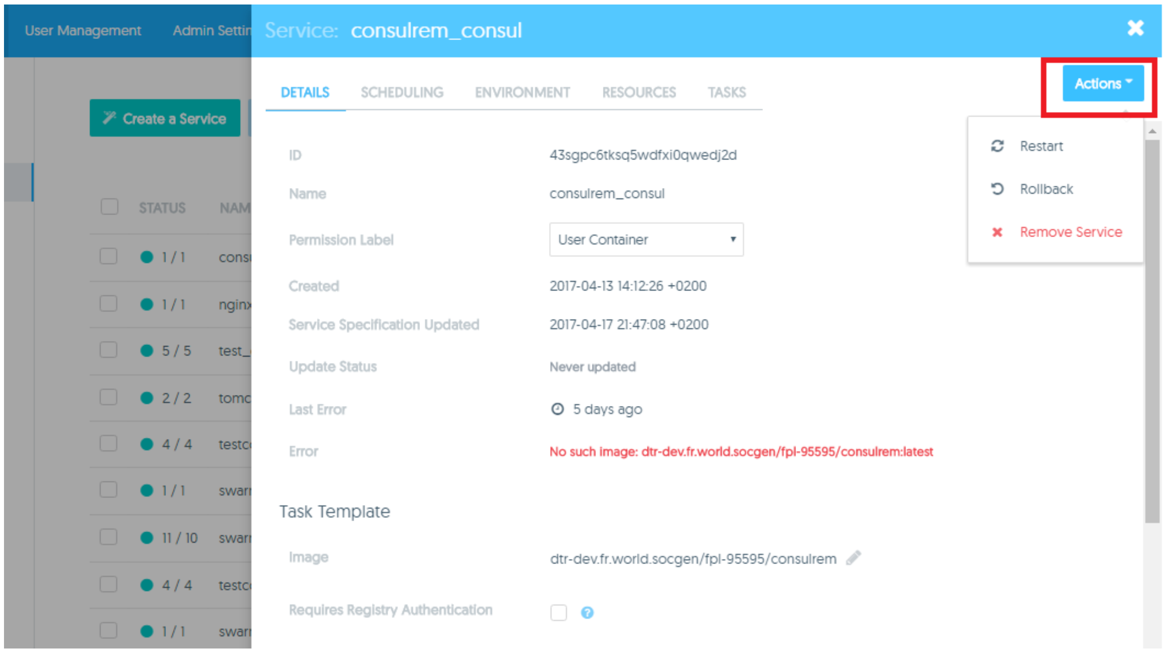


STATUS	NAME	IMAGE	
2 / 2	Sectoolweb_pt	dtr-dev.fr.world.socgen/itec_fcc_osd_sf/sectoolweb-pt-services:1.3.10	Replicated
1 / 2	Sectoolweb_dealsetup	dtr-dev.fr.world.socgen/itec_fcc_osd_sf/sectoolweb-dealsetup-services:1.7.10	Replicated
1 / 2	Sectoolweb_payment	dtr-dev.fr.world.socgen/itec_fcc_osd_sf/sectoolweb-payment-service:1.4.13	Replicated
2 / 2	Sectoolweb_adminconsole	dtr-dev.fr.world.socgen/itec_fcc_osd_sf/sectoolweb-admin-console:1.2.50	Replicated
2 / 2	Sectoolweb_sds	dtr-dev.fr.world.socgen/itec_fcc_osd_sf/sectoolweb-sds-services:1.3.3	Replicated
2 / 2	Sectoolweb_client	dtr-dev.fr.world.socgen/itec_fcc_osd_sf/sectoolweb-client:2.2.17	Replicated
1 / 2	Sectoolweb_common	dtr-dev.fr.world.socgen/itec_fcc_osd_sf/sectoolweb-common-services:1.4.34	Replicated
1 / 2	Sectoolweb_apigateway	dtr-dev.fr.world.socgen/itec_fcc_osd_sf/sectoolweb-api-gateway:1.2.41	Replicated

You can now remove or restart the full stack of services of your app.

15.2.2 Restart, Rollback, Remove a service

Select your service by clicking on it, and clic on the "action" button top right.



ID	43sgpc6tksq5wdfxi0qwedj2d
Name	consulrem_consul
Permission Label	User Container
Created	2017-04-13 14:12:26 +0200
Service Specification Updated	2017-04-17 21:47:08 +0200
Update Status	Never updated
Last Error	5 days ago
Error	No such image: dtr-dev.fr.world.socgen/fpl-95595/consulrem:latest

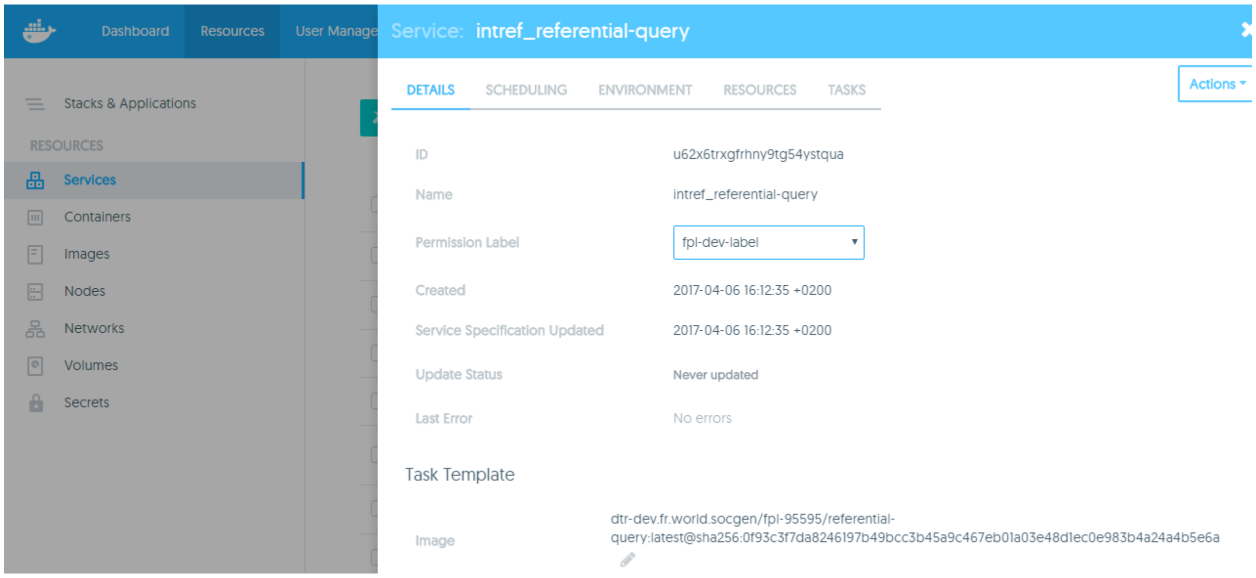
Task Template

Image	dtr-dev.fr.world.socgen/fpl-95595/consulrem
Requires Registry Authentication	☐

You can now restart your service. Or rollback (will restart your service with the precedent image used) or remove it.

15.2.3 Scale a service

Select your service by clicking on it, and go in the "scheduler" tab



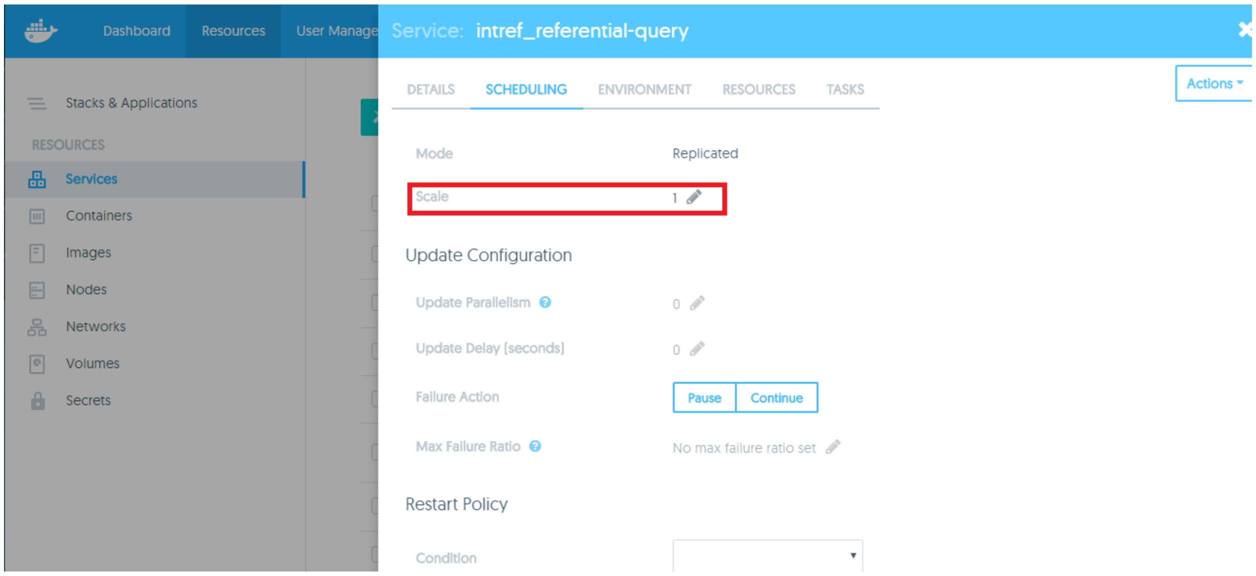
Service: intref_referential-query

ID	u62x6trxgfrhny9tg54ystqua
Name	intref_referential-query
Permission Label	fpl-dev-label
Created	2017-04-06 16:12:35 +0200
Service Specification Updated	2017-04-06 16:12:35 +0200
Update Status	Never updated
Last Error	No errors

Task Template

Image	dtr-dev.fr.world.socgen/fpl-95595/referential-query:latest@sha256:0f93c3f7da8246197b49bcc3b45a9c467eb01a03e48d1ec0e983b4a24a4b5e6a

Now you just have to edit the "scale" number and save changes



Service: intref_referential-query

Mode	Replicated
Scale	1

Update Configuration

Update Parallelism	0
Update Delay (seconds)	0
Failure Action	Pause Continue
Max Failure Ratio	No max failure ratio set

Restart Policy

Condition	

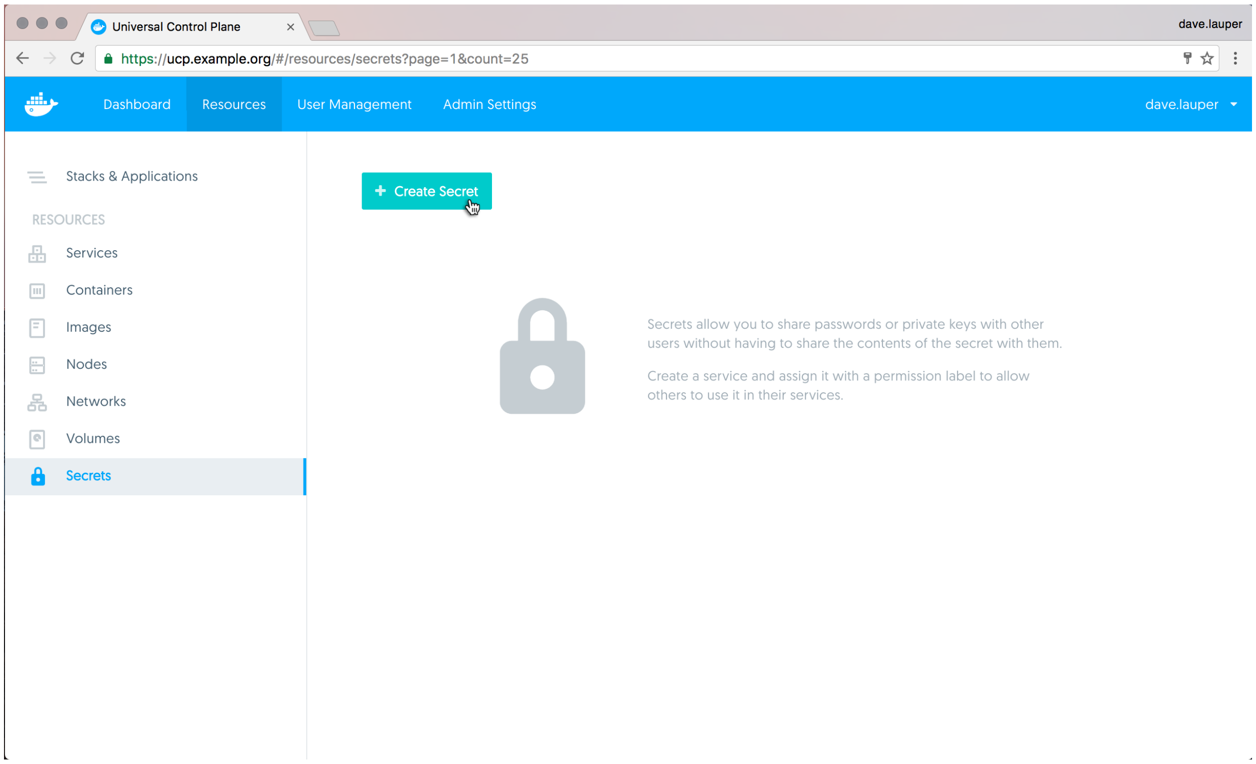
Manage secrets in UCP

When deploying and orchestrating services, you often need to configure those services with sensitive information like passwords, TLS certificates, or private keys.

Universal Control Plane allows you to store this sensitive information, also know as secrets, in a secure way. It also gives you role-based access control so that you can control which users can use a secret in their services and which ones can manage the secret.

16.1 Create your secret using the UI

In the UI go in resources -> secrets.



Secrets allow you to share passwords or private keys with other users without having to share the contents of the secret with them.

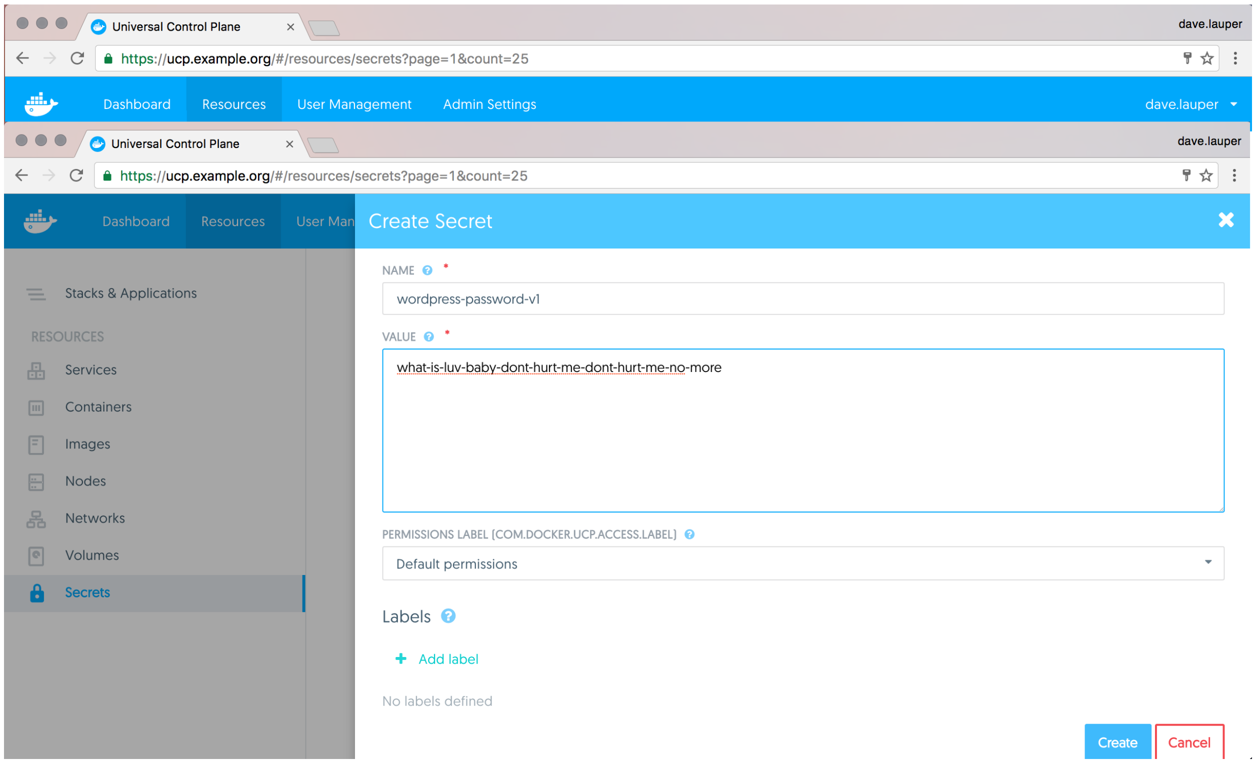
Create a service and assign it with a permission label to allow others to use it in their services.

Just click on create secret and :

- Assign the name of your secret in the NAME field.

- Value is your secret (password or certificate by example)
- Label : a secret and service must have the same label. So set your "team label" here, so that other users have permission to use this secret.

Warning: Once your create a secret, you won't be able to see or edit the secret data again.



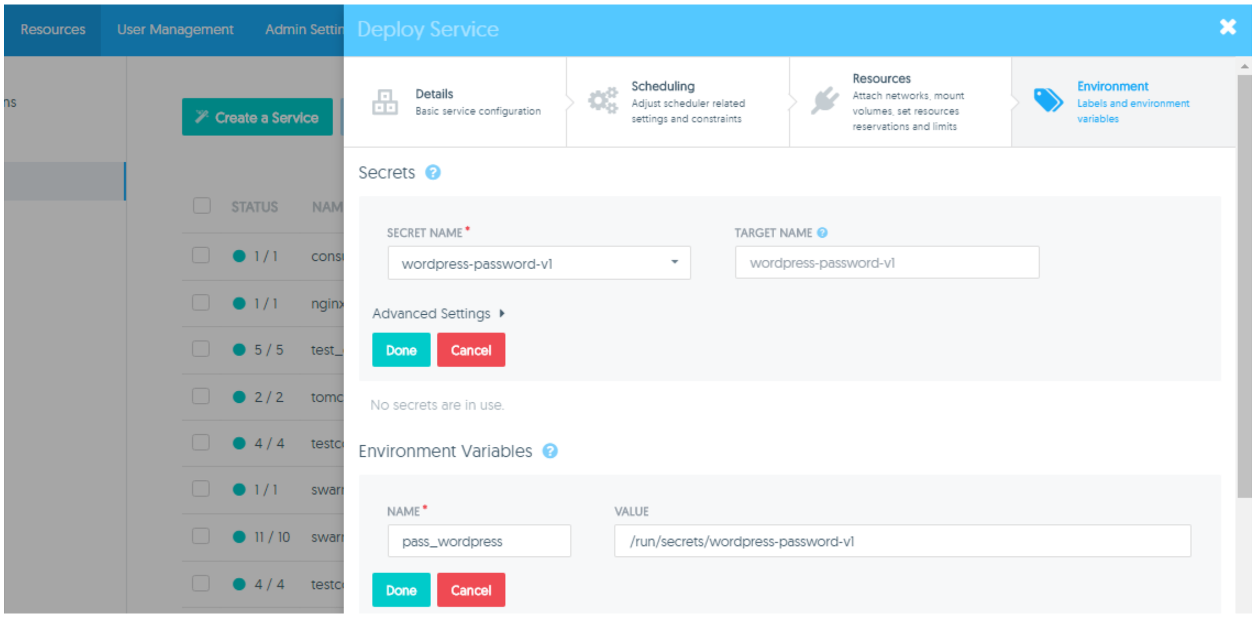
NAME *
wordpress-password-v1

VALUE *
what-is-luv-baby-dont-hurt-me-dont-hurt-me-no-more

PERMISSIONS LABEL [COM.DOCKER.UCP.ACCESS.LABEL]
Default permissions

16.2 Assign a variable to your secret

Create your service, and in the "environment" tab you'll find the secret part. (don't forget to assign the same label to your service and secret). Assign your secret to your service, and give it a name. This will create at launch a file under /run/secrets/wordpress-password-v1. You can also assign a environment variable to this file :


Deploy Service

Details: Basic service configuration

Scheduling: Adjust scheduler related settings and constraints

Resources: Attach networks, mount volumes, set resources reservations and limits

Environment: Labels and environment variables

Secrets

SECRET NAME*: wordpress-password-v1

TARGET NAME: wordpress-password-v1

Advanced Settings

Done Cancel

No secrets are in use.

Environment Variables

NAME*: pass_wordpress

VALUE: /run/secrets/wordpress-password-v1

Done Cancel

16.3 Secret in docker-compose.yml

You can also create your secret using a command line. Set your environment via the UCP bundle, and then use the "docker secret create" command :

```
docker secret create --label label_type=your_label secret_name your_secret
docker secret create --label com.docker.ucp.access.label=app-dev-label app-dev-credentials ./my_app_c
jtn7g6aukl5ky7nr9gvwafoxh
```

Here is an example of a docker-compose.yml using this secret.

```
 app-query:
  image: "dtr-dev.fr.world.socgen/fpl/app_image:tag"
  ports:
   - 80
  environment:
   - CREDENTIALS_FILE=/run/secrets/credentials.env
  secrets:
   - source: app-dev-credentials
     target: credentials.env

secrets:
 app-dev-credentials:
  external: true
```

Warning: You can also assign GUI, GID, and permissions to a secret.

16.4 Secret commands

List secrets :

```
docker secret ls
```

Create secret :

```
docker secret create --label label_type=your_label secret_name your_secret
```

remove a secret :

```
docker secret rm secret_id
```

or

```
docker secret rm secret_name
```

16.5 Update a secret

If the secret gets compromised you'll need to rotate it so that your services start using a new secret. In this case we need to change the password we're using and update the MySQL and WordPress services to use the new password.

Since secrets are immutable in the sense that you cannot change the data they store after they are created, we can use the following process to achieve this:

- Create a new service with a different password
- Update all the services that are using the old secret to use the new one instead
- Delete the old secret

CHAPTER 17

Load balancer

Docker UCP implements a managed load balancer using HAProxy that can be manually scaled to multiple hosts. A load balancer can be used to distribute network and application traffic to individual containers by directly adding them or "linked" to a basic service. A basic service that is "linked" will have all its underlying containers automatically registered as load balancer targets by UCP.

Learn more about how to use the load balancers in UCP and how we use Docker UCP HRM (Http Routing Mesh) features.

17.1 HRM

Warning: HRM works only with service and stack

Warning: The timeout of HRM can not be modified and is set to 30s

HRM is a process which detect creation/destruction of containers and configure the HTTP reverse proxy layer accordingly. To be able to expose a container (in HTTP(S)) to the outside world.

You can etiher start a container after setting your client by the bundle or using the web interface of UCP. Since we are using HRM to do make the service layer, you need to add a label for HTTP(S) services, expose a port, and connect to a dedicated HRM overlay network:

For HTTP : "com.docker.ucp.mesh.http.80=internal_port=443,external_route=http://myapplication.docker-dev-hrm.fr.world.socgen"

For HTTPS : "com.docker.ucp.mesh.http.443=internal_port=443,external_route=sni://myapplication.docker-dev-hrm.fr.world.socgen"

You need multiple informations to complete the setup:

- The internal port of your container (eg : 8080)
- The external route (FQDN) you want to use
- **A valid certificate for HTTPS**

Warning: Before using HRM, please ask PaaS Team to create a network and an access label for this network

Warning: For HTTPS, the container must decypher and host the certificates. HRM will only pass through SSL to a container.

Warning: Be aware that for non production you have to use **docker-dev-hrm.fr.world.socgen** and **docker-prod.fr.world.socgen** for production

Warning: You have to expose a port of your service. **Do not fix it, let docker choose one**

Warning: Take note that the external_route is **sni://** for https

Warning: **For the name of the FQDN/hostname to reach your application, please use this pattern [hostname].[trigram]** to be sure to not override some configurations in HAProxy

Note: As the network is created outside of the stack, it must be declared as **external** or be explicitely used in command line

Example with a docker-compose stack and a project name cmbs (docker stack deploy cmbs -c docker-compose.yaml) :

```
version: "3.1"
  services:
    creditihm:
      image: dtr-dev.fr.world.socgen/gtsmktdemo/creditihm:1.4
      networks:
        - cmbs-hrm-network
      ports:
        - "8443"
      deploy:
        labels:
          com.docker.ucp.mesh.http.8443:"internal_port=8443,external_route=sni://cmbs.docker-dev-hrm
          com.docker.ucp.access.label: "app-hrm-access-label"
  networks:
    cmbs-hrm-network:
      external:true
```

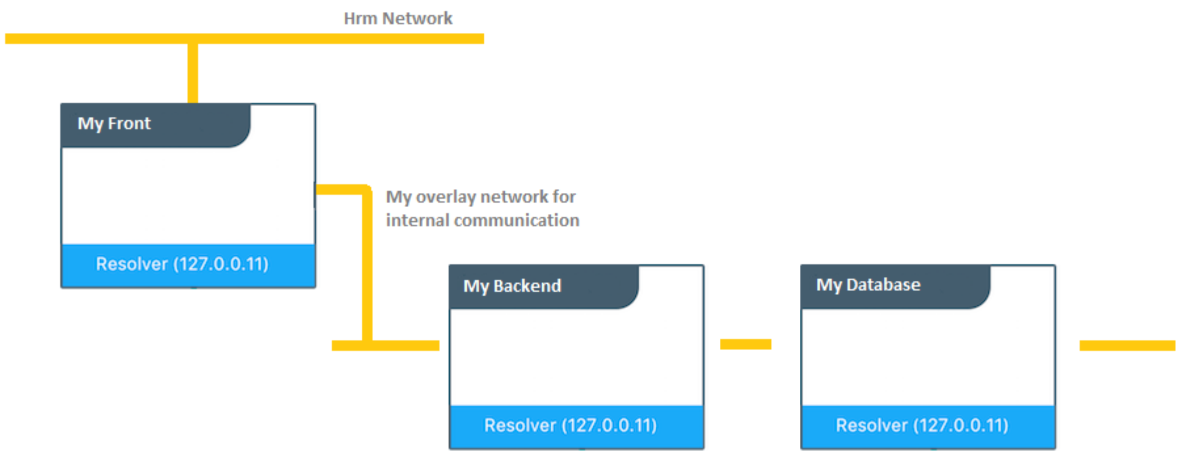
Warning: **Do not use uppercase letters**, prefer to use lowercase letters to define URLs

Or with a command line :

```
docker service create --name creditihm --network cmbs-hrm-network -p 8443 --label com.docker.ucp.mesh
```

You'll be able to access your application by following the URL : http://[your_application_name].docker-dev-hrm.fr.world.socgen

Warning: HRM is an overlay network dedicated to give http access to your application. It ***SHOULD NOT*** be used for communication between frontend and backend if you don't need to request your backend using http. If you don't need http request for your backend, simply create an overlay network using your stack file.yml
Network infra should look like that :


Hrm Network
My Front
Resolver (127.0.0.11)
My overlay network for internal communication
My Backend
Resolver (127.0.0.11)
My Database
Resolver (127.0.0.11)

CHAPTER 18

Persistent storage service

Storage service is provided by a netapp plugin for ONTAP NAS –> https://github.com/NetApp/netappdvp

18.1 Netapp plugin

Volumes inside the containers are mounted as a NFS share, and the plug-in is installed on each compute node. You can modify some option regarding the volume you create, but by default the plug-in offering is :

- Size of a volume is **1GB**
- Volume is own by **root** with rwxr-x-rx permissions (755)
- Space is not reserved on the NAS (Only the real space used by the volume is consumed on the NAS. Eg, if you set a 2GB volume and used only 10MB, only 10MB are consumed on the NAS)

Available options :

- Size of the volume
- Permissions on the volume

Warning: Before using volume, please ask PaaS Team to create an access label This access label will ensure that your volume can not be mounted by other team/applications. You have to use it when using stack or create service with command line

Warning: Inodes are related to the size of the volume. Please size correctly the volume. If you do not know, let the default value

Warning: **Do not put "-" in the naming of your volume, it will fail in creation with no clear error message

Note: **Do prefix your volume name with your application name (ex: **dck_9234_vol_**....)

Warning: Regarding the volume name, please use this pattern : applicationName_env_(serviceName || imageName || buildName)

Note: **Please use volume each time you need to store data in a volume, even for a short period of time.** It will avoid to fullfill the host file system of the compute node

Note: **Docker will not destroy the data after destroying container.** You have to do it explicitly with dedicated commands

Note: **Volume are shared by default.** So if 2 containers use the same volume name, the volume will be shared between those containers

18.1.1 Create volumes with docker-compose (stack deploy)

```
service:
  creditihm:
    image: dtr-dev.fr.world.socgen/gtsmktdemo/creditihm:1.4
      networks:
        - frontend
        - backend
      ports:
        - "8080"
      deploy:
        labels:
          com.docker.ucp.access.label: "app-storage-access-label"
      volumes:
        - ihmdata:/tmp/data
volumes:
  ihmdata:
    driver: netapp
    driver_opts:
      size: "3g"
      unixPermissions:"---rwxrwxrwx"
```

Warning: Volume and data are not destroyed when you remove your stack project

To delete the volume **(and all the datas)**, ressources created by the docker stack :

```
docker volume rm myvolume
```

Warning: If you scale a service that use a volume, **this volume will be shared with all the instances of this service**

18.1.2 Create volumes with swarm service in command line

To create (or mount) a volume with docker swarm service, use this command :

```
docker service create -name myservice --label com.docker.ucp.access.label="app-storage-access-label"
```

Available options are :

- volume-driver : Driver to use, currently **netapp**

- source : name of the volume (see the naming pattern)
- destination : folder to mount the volume in your container
- volume-opt :
- size : size of the volume
- unixPermissions : permissions to set to the folder

If the volume doesn't exist, it will be created. Otherwise it will be mounted.

Warning: The *docker service rm* command do not destroy the volume and the data, but only unmount it. To destroy the volume and the data, you have to remove the volume with appropriate command

```
docker volume rm  myvol
```

Warning: If you scale a service that use a volume, **this volume will be shared with all the tasks of this service**

18.2 CIFS plugin

First you'll need to create a volume "cifs" on UCP interface. This volume is a kind of "link" to your cifhost share. By clicking on the "create" button, UCP will create a volume on each node where cifs plugin is enabled.

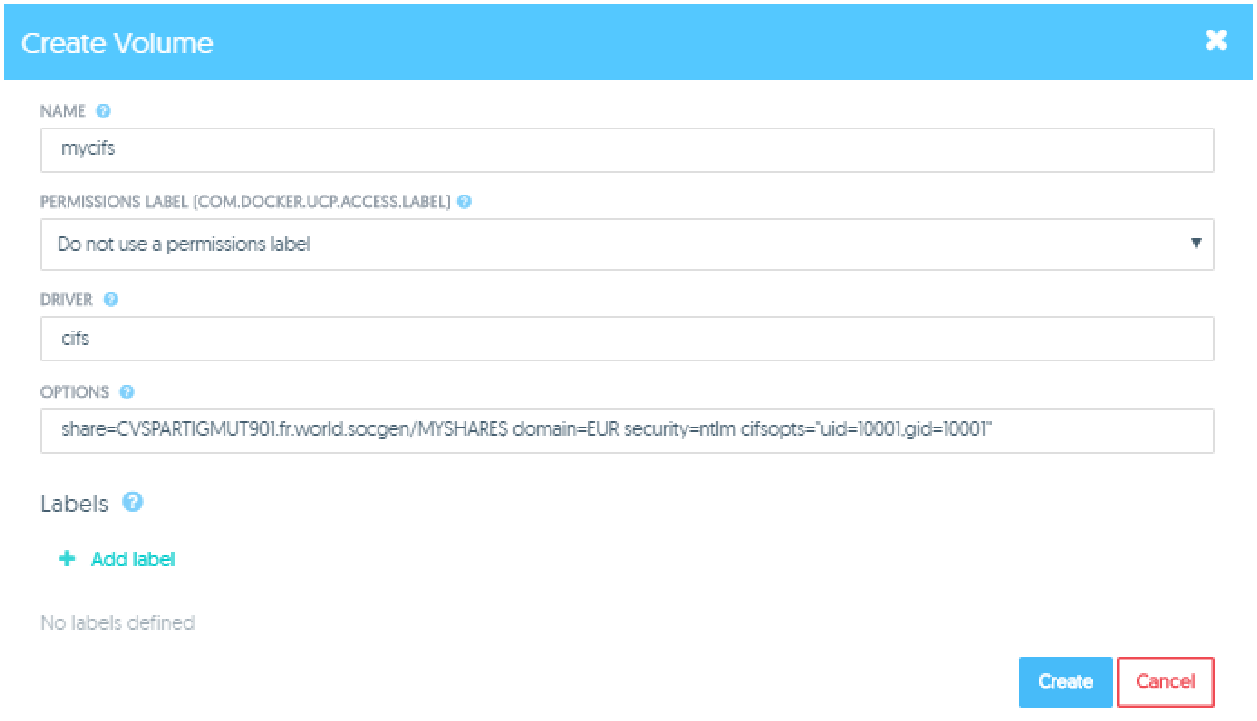
The username and password options SHOULD NOT be set in UCP.

Warning: Credentials will be store in Vault and the CIFS plugin will lookup in Vault to get the credentials. That's why you need to provide the exact volume name you created in UCP which should match with the Vault's entry. Volume name should be in lower case and each word should be seprate by comma (IE : my-wonderful-cifs-volume)

You'll need to create a generic impulse case to Paas TEAM, with the "name" of you volume and the "username". Then you'll provide the password using **C3** to the assignee.

Please use this Impulse template to provide information to the FT PaaS for storage CIFS credentials : Impulse => Infrastructure Service Catalog => Market Data Middleware => PaaS => [MDM][DCK][CIFS] Credentials setup on PaaS platform

Under UCP you will just provide the others options, **it's a comma between uid and gid. Please also carreful with your environment variables on your image such as $HOME, your mount point on your Compose-file and some commands like chown** :


Create Volume
NAME
mycifs
PERMISSIONS LABEL (COM.DOCKER.UCP.ACCESS.LABEL)
Do not use a permissions label
DRIVER
cifs
OPTIONS
share=CVSPARTIGMUT901.fr.world.socgen/MYSHARES domain=EUR security=ntlm cifsopts="uid=10001,gid=10001"
Labels
+ Add label
No labels defined
Create
Cancel

Example for OPTIONS : share=CVSPARTIGMUT901.fr.world.socgen/MYSHARE$ domain=EUR security=ntlm cifsopts=”uid=10001,gid=10001”

uid and gid should be Numeric Value (no explicit name)

uid=10001 -> OK // uid=monUser -> KO

Warning: As explained above, for security reasons, DO NOT use UCP to store your cifs username/password.

18.2.1 Launch a stack from docker-compose.yml

Here is docker-compose.yml **example**, you just need then to run “docker stack deploy -c docker-compose.yml mystack”

```
version: "3.1"

services:

  alpine:
    image: "dtr-dev.fr.world.socgen/gtsmktssb/rhel7:7.2"
    deploy:
      replicas: 1
    ports:
      - 8500
    volumes:
      - mycifs:/data
    entrypoint: tail -f /etc/hosts

volumes:
  mycifs:
    external: true
```

Known Issues :

- scaling the service could start a container on a node without plugin activated. A case has been created on support.docker.com : 21082.

CHAPTER 19

Access

The DTR provides a repository to store your docker images. The service provides high availability, access control and security scanning.

Dev DTR URL : https://dtr-dev.fr.world.socgen/repositories Prod DTR URL : https://dtr-prod.fr.world.socgen/repositories

19.1 Organizations and Teams

You need to be authentified to be able to push and pull image from the DTR. A repository is assign to a team, and members are assign to teams. That way, members of a team can all push and modify image in their own repository.

If you want to create a team, its name should be : applicationTrigram_codeIappli

Warning: You need to be an "Orga member" to be able to create teams.

19.2 Push an image to the DTR

You need to set your bundle to push image to your DTR. If you don't know what is the bundle, have a look at the "use UCP!" documentation.

Warning: When the source.env is settled, you can't see your local images.

1. [Optional] Unset the three variables of env.sh

2. Login as your user

```
docker login dtr-dev.fr.world.socgen
```

3. [Optional] If your repository doesn't exist, create it either by the DTR GUI or API

```
curl -s -XPOST -H 'Content-Type:application/json' -H 'Accept:application/json'  -d '{"name":" image_
```

4. Tag your image

```
docker tag [image_id] dtr-dev.fr.world.socgen/[your_repo]/[image_name]:[image_version]
```

5. Push your image

Warning: The name of your image is very important! To be able to push to the dev DTR the name has to be dtr-dev.fr.world.socgen/[your_repo]/[image_name]:[image_version]

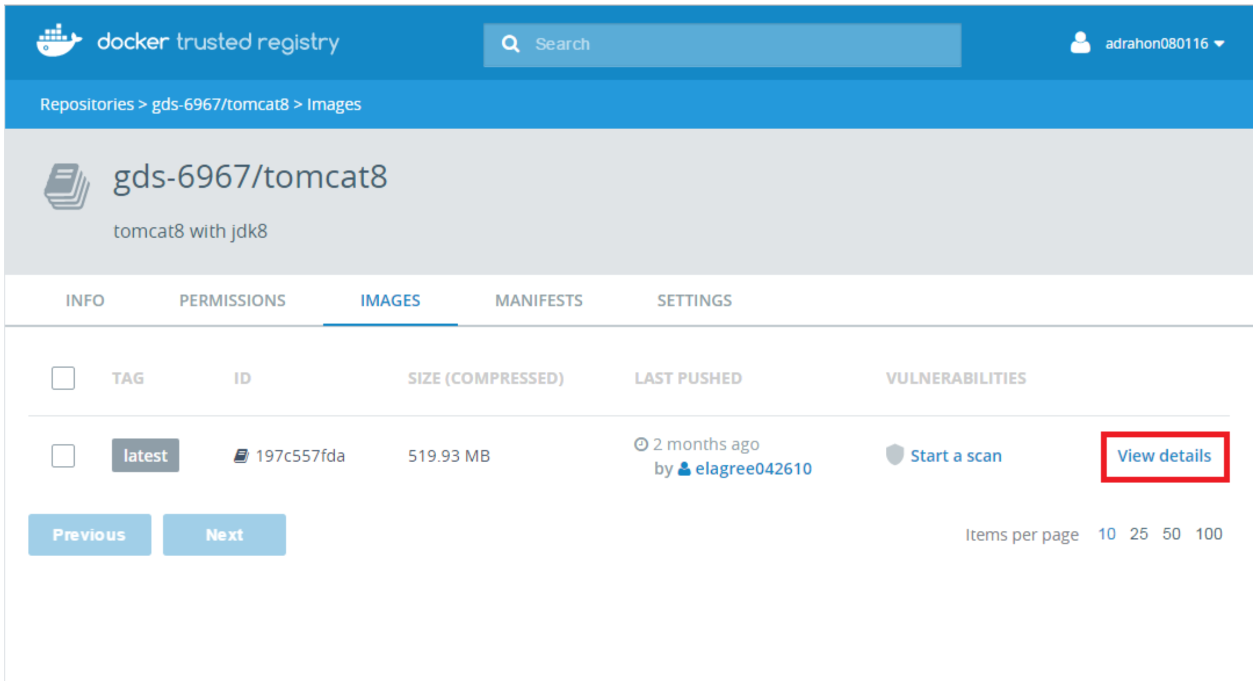
19.3 DTR API

The reference: https://docs.docker.com/v1.8/docker-trusted-registry/api/.

Security Scanning (DSS)

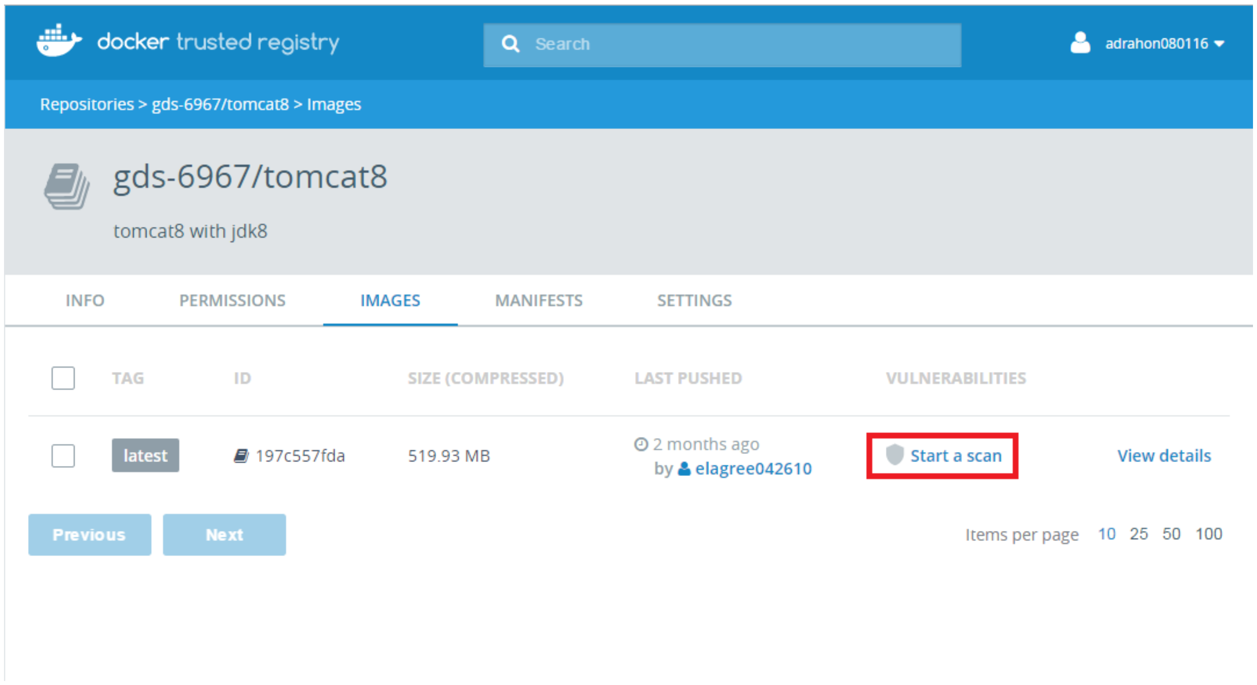
DTR has a built in security scanner that can be used to discover what versions of software are used in your images. It scans each layer and aggregates the results to give you a complete picture of what you are shipping as a part of your stack. Most importantly, it co-relates this information with a vulnerability database that is kept up to date through periodic updates.

When you push an image, it is automatically scanned and the result can be seen there : repositories -> view details on your image -> "images" tab -> view details



TAG	ID	SIZE (COMPRESSED)	LAST PUSHED	VULNERABILITIES
latest	197c557fda	519.93 MB	2 months ago by elagree042610	Start a scan

You can also start a manual scan by clicking on the "start a scan" button :



TAG	ID	SIZE (COMPRESSED)	LAST PUSHED	VULNERABILITIES	
latest	197c557fda	519.93 MB	2 months ago by elagree042610	Start a scan	View details

Software Factory services

ITEC/ARC/COC Jenkins team has actively worked on easing Docker DDC integration in the continuous integration pipeline. You can see the PlatformCDForPAAS in https://sbc.safe.socgen/docs/DOC-173828

Their internal contributions, **only for their own use** are here:

1. Jenkins & Docker
2. Bring your Jenkins master in UCP

CHAPTER 22

Intro

22.1 Vault or Docker ?

Docker secret management enables you to store secrets in Docker with UCP adding a layer of Rule Based Access Controls (see chapter Docker Secret Management).

Vault from Hashicorp is a solution that enables you to store secrets with many more functionalities (tokens, audit, policies, ...) and can be used for legacy applications outside the Docker platform (see chapter Vault Secret Management).

But to access Vault you need to secure an initial secret which you cannot store in Vault. That is why we highly recommand you to store your Vault initial secret as a Docker secret (approle secret) and all your application secrets in Vault. The initial secret secured by docker will enable you, inside your conatiners to access the secret of your application in the Vault.

Keep in mind that when you use a docker secret, if you need to update it, you have to re-create it. And to remove the secret before re-creating it, you need to remove your stack also. With Vault you can update your secret and just update your stack.

22.2 Sample of code inside a container

secrets stored in UCP are available, with correct rights, in /run/secrets mount point. This exemple use Token ID stored in UCP to connect on Vault and then recover for exemple à DB password.

```
export VAULT_URL=https://vault-$ENV.fr.world.socgen
export VAULT_CA=/run/secrets/vault-ca-$ENV.pem

_VAULT_SECRET=`cat /run/secrets/paas-vault-access-$ENV`
_VAULT_TOKEN=`curl -k -X POST -d '{"role_id":"dck-0001-login-readdev","secret_id":"'$_VAULT_SECRET'"

DB_PASS=`curl -k -X GET -H "X-Vault-Token: $_VAULT_TOKEN" $VAULT_URL/v1/secret/dck-0001/dev/portalpa
```

Vault

Vault from Hashicorp is a solution that enables you to store secrets with many more functionalities (tokens, audit, policies, ...) and can be used for legacy applications outside the Docker platform.

Vault can be used with api request.

We have wrote our API for our use. All information can be find in our portal

https://portalpaas.docker-dev-hrm.fr.world.socgen

23.1 Use our API

Vault infra is available here for non-prod (no GUI or Web Site, url only redirect to the different Vault API).

First choose a name for your application, as the API is not yet on ITAAS you can choose whatever.

It must be of the form : tri-XXXXXX, so you can use by example rag-1234 and change the numbers for each test.

You will use on the onboarding API, a login from AD on the eur branch for the moment (technical account are on the eur branch)

First create the context for your application in the vault.

```
curl -v -k -X POST -u <login> https://portalpaas.docker-dev-hrm.fr.world.socgen/vault/api/v1/applicat
```

Then create a policy inside your context.

```
curl -v -k -X POST -u <login> -d '{"path":{"secret/rag-1234/test/*":{"policy":"write"}}}' https://por
```

Then create credentials (technical account) in the vault with policy for your application.

```
curl -v -k -X POST -u <login> -d '{"policy":"test-write"}' https://portalpaas.docker-dev-hrm.fr.worl
```

Warning: Don't forget to save secret_id returned by the request (ex: "c2018e37-6d4f-z284-5eb1-50t2v9w6bec3") You will use it with role id to login to Vault API.

You can now log in to the Vault.

```
curl -k -X POST -d '{"role_id":"rag-1234-login-test","secret_id":"My-Secret-id"}' https://vault-dev.
```

With the token you can now read or write secrets in your application context.

```
curl -k -X POST -H "X-Vault-Token: <your token>" -H "Content-Type: application/json" -d '{"value":"ba
```

Then read your secret.

```
curl -k -X GET -H "X-Vault-Token: <your token>" https://vault-dev.fr.world.socgen/v1/secret/rag-1234/
```

You can also use the python hvac library which is really simple for direct calls to the Vault.

urls showed in this Doc are **DEV** urls.

- VAULT ONBOARDING API: https://portalpaas.docker-dev-hrm.fr.world.socgen
- VAULT API: https://vault-dev.fr.world.socgen
- UI: https://vault-ui.docker-dev-hrm.fr.world.socgen

PROD urls are

- VAULT ONBOARDING API: https://portalpaas.docker-prod.fr.world.socgen
- VAULT API: https://vault-prod.fr.world.socgen
- UI: https://vault-ui.docker-prod.fr.world.socgen

UCP

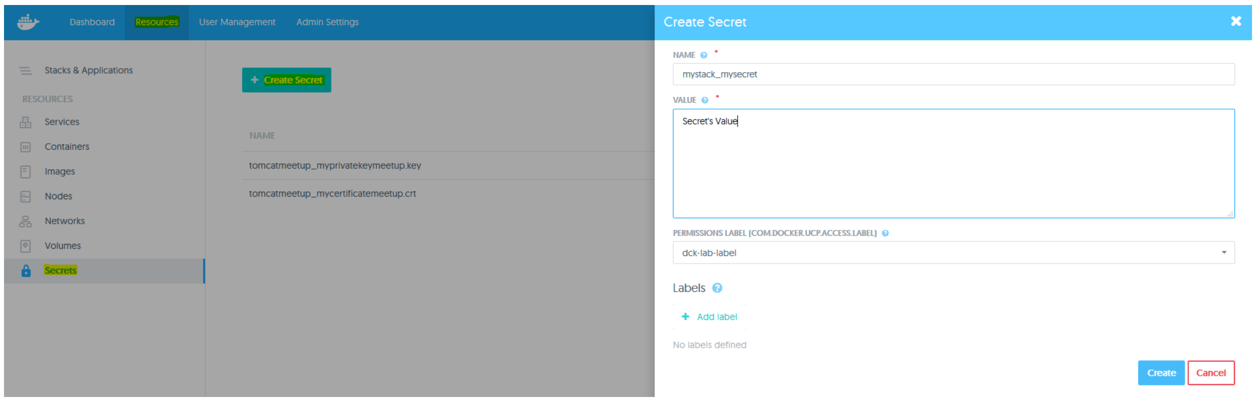
In UCP Resources, secrets can be stored. These secrets are key/value store, protected with a label.

Note that these secrets can only be used in the DDC plateform. Another item outside the Cluster will not be able to see the secret. So if all your infra is not on the PAAS platform, secrets will not be shared.

That's why we recommand to store your secret in VAULT (cf Vault part of the doc) and put in UCP secret your Vault Token ID, so your containers will be able to access to your Vault entries.

24.1 Create your secret

A secret is a key/value item, stored and encrypted in UCP. The secret is available only by users with good label.



NAME

tomcatmeetup_myprivatekeymeetup.key

tomcatmeetup_mycertificatemeetup.crt

NAME *

mystack_mysecret

VALUE *

Secret's Value

PERMISSIONS LABEL (COM.DOCKER.UCP.ACCESS.LABEL)

dck-lab-label

Labels

No labels defined

Warning: Your secret name should be prefixed by your stack Name, in exemple **mystack**

24.2 Invoke your secret

Your UCP secret is now available for your Container/Stack/Service In your compose file

```
secrets:
  - source: mysecret
        target: mytargetfile.txt
        uid: '997'
        gid: '996'
        mode: 0440
```

- source: secret name in UCP without the "**stackname_**"
- target: file where your secret will be stored. This file will be used in your Dockerfile
- uid/gid/mode: Unix rights wich will be set on the file(Owner Group Rights). not necessary

Note: user launching Compose should have the right on the label defined in the secret.

Secrets stored in UCP are available, with correct rights, in /run/secrets mount point. you can use it in your Dockefile, for exemple:

```
UCP_SECRET=`cat /run/secrets/mytargetfile.txt`
```

CHAPTER 25

Intro

25.1 What is Consul ?

Consul is a software from Hasicorp compagny (like Vault or Terraform). It's a tool for service discovery and configuration management. It's providing some cool features like :

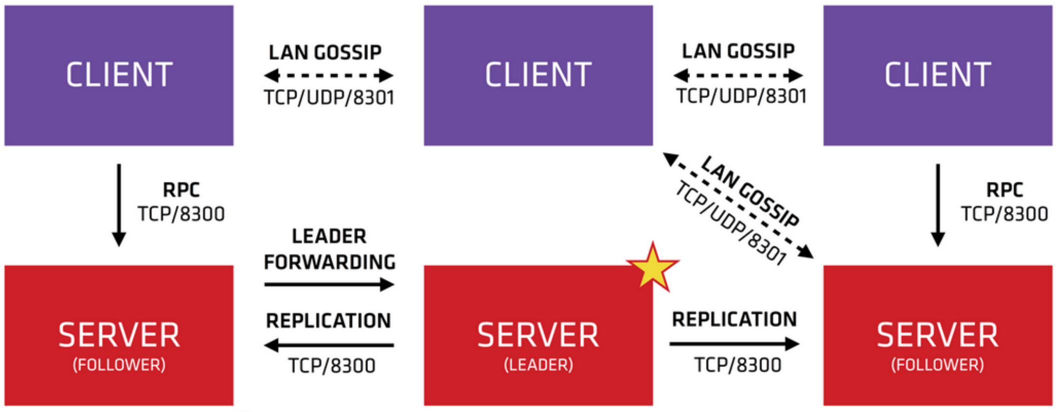
- Service Discovery - Consul makes it simple for services to register themselves and to discover other services via a DNS or HTTP.
- Health Checking - Health Checking enables Consul to quickly alert operators about any issues in a cluster. The integration with service discovery prevents routing traffic to unhealthy hosts and enables service level circuit breakers.
- Key/Value Storage - A flexible key/value store enables storing dynamic configuration, feature flagging, coordination, leader election and more. The simple HTTP API makes it easy to use anywhere.
- Multi-Datacenter - Consul is built to be datacenter aware, and can support any number of regions without complex configuration.

25.2 Consul in SG

We are providing 2 Mutualized Consul Clusters (NON-PROD & PROD) for you. Each cluster are split between Marccousis, Tigery and Orion datacenter to provide you high availability and low latency. Currently we are using Consul 0.9.3.

Binaries for your Consul agent client can be downloaded from Consul website : https://www.consul.io/downloads.html Consul is working in Linux or Windows.

Recommanded usage of Consul is to **NOT interact directly with the Consul cluster** but use a Consul agent client on your local applicative server (where your service are running).



You need to register your applicative service on your own Consul agent client locally on your applicative server.

Multi-tenancy is guarantee using Consul ACLs which enforce security and application segregation using specific patterns and rules.

Each applications will receive 4 ACLs :

- Technical ACLS:
- [trigram]-[env]-acl-token : the default ACLs
- [trigram]-[env]-acl-agent-token : to perform internal operations
- Applicative ACLS:
- [trigram]-[env]-read-ACL : The applicative read ACL which allow to read your service, K/V store, events and agents
- [trigram]-[env]-write-ACL : The applicative write ACL which allow to read/write your service, K/V store, events and agents

You can find Consul documentation here : https://www.consul.io/docs

25.3 Development best-practice

It is not recommended to register your desktop/laptop on the Consul cluster. To avoid doing so, you can run a local Consul Agent in dev mode:

```
$> consul agent -dev
```

For more information, please refer to the official documentation

CHAPTER 26

How to request access to the Mutualized Consul cluster

26.1 Using Impulse

You can request your own access to the Mutualized Consul cluster using Impulse => https://go.fr.world.socgen/impulse Once you are logged you can go to Service Catalogs => Infrastructure Service Catalog => Middleware => PaaS and use template [MDM][CSL] Mutualized Consul cluster access or use directly [this link]

You will have to provide your Trigram and on which cluster you want to be onboarded (NON-PROD or PROD).

Once the ACLs will be created we will sent them by mail in C3.

26.2 Using API

To be available soon

CHAPTER 27

Use Consul

27.1 How to configure your own Consul agent client

Note: As we explain previously you MUST use a Consul agent client to interract with the Consul cluster

To connect to a Consul Cluster you will need some specific stuff like :

- Encrypt Key used for encrypted communication between all nodes
- Datacenter tell to Consul on which Datacenter it's running
- Domain used by Consul for DNS queries
- DNS Name of the servers used by the Consul cluster
- Acl_Agent_Token use to perform internal operations
- Acl_Token use as default Acl
- Node_Name used to setup the name of your Consul agent client

Here are the common parameters for Mutualized Consul Cluster:

Region	Env	Datacenter	Domain	Consul cluster servers	Node Name
Paris	DEV	eu-fr-paris	csldevparmut	csldevparmut01/02/03/04/05	[yourtrigram]cslclipardev_[yourservername]
	PROD		cslprdparmut	cslprdparmut01/02/03/04/05	[yourtrigram]cslcliparprd_[yourservername]
HK	DEV	hongkong	consulcsldev	cslprdhkweb003/004/001/005/01	[yourtrigram]cslclihkgdev_[yourservername]
	PROD		consulcslprd	cslprdhkweb006/007/008/009/02	[yourtrigram]cslclihkdprd_[yourservername]

Warning: If you don't follow naming convention for your NodeName, it will not be able to register

Warning: Please do not register your laptop/desktop in consul. Only register your servers

Here is an example of the config.json (Consul config file) in DEV :

```
{
     "server": false,
     "datacenter": "eu-fr-paris",
     "domain": "csldevparmut",
     "data_dir": "/home/webdev01/consul_data/tmp",
     "node_name": "dckcslclipardev_csldevweb054",
     "ui": true,
     "encrypt": "TV223LDgZx232375QhiGu3c-=",
     "retry_join": ["csldevparmut01.fr.world.socgen","csldevparmut02.fr.world.socgen","csldevparmut0
     "bind_addr": "192.168.0.1",
     "client_addr": "192.168.0.1",
     "ports" : { "http": -1, "https": 8543},
     "protocol": 3,
     "acl_agent_token" : "d1xd2238d-1cbd-01dc-98c8-2323xd23321Q",
     "acl_token" : "829f75ad-f1cz-fgy6-33d2-xd281JnbT56Y",
     "acl_datacenter": "eu-fr-paris",
     "ca_file": "/home/webdev01/consul_data/ssl/CA.pem",
     "cert_file": "/home/webdev01/consul_data/ssl/server.crt",
     "key_file": "/home/webdev01/consul_data/ssl/server.key",
     "acl_default_policy": "deny",
     "acl_down_policy": "deny",
     "disable_remote_exec": true,
     "skip_leave_on_interrupt": true,
     "verify_outgoing": true,
     "disable_update_check": true,
     "enable_script_checks": true
}
```

27.2 ACLs definition

Applicatives ACLs are created with this defintion

Name : [trigram]-[env]-ACL-read Rules :

Regions is one of:

- Paris: `par`
- Asia: `hkg`

# Applicative ACL read in [env] for [trigram] application

# —– Node part —–

node "[trigram]cslcli[region][env]_" { policy= "read"

}

node "dckcslcli[region][env]_" { policy= "read"

}

agent "[trigram]cslcli[region][env]_" { policy= "read"

}

agent "dckcslcli[region][env]_" { policy= "read"

}

session "[trigram]cslcli[region][env]_" { policy= "read"

}

session "dckcslcli[region][env]_" { policy= "read"

}

# —– Key/Value part —–

# Read only [trigram]/[env]/ key

key "[trigram]/[env]/" { policy = "read"

}

# —– Service part —–

# Read only [trigram]-[env]- services

service "[trigram]-[env]-" { policy = "read"

}

# —– Events part —–

# Read only [trigram]-[env]- events

event "[trigram]-[env]-" { policy="read"

}

Name : [trigram]-[env]-ACL-write Rules :

# Applicative ACL write in [env] for [trigram] application

# —– Node part —–

node "[trigram]cslcli[region][env]_" { policy= "write"

}

node "dckcslcli[region][env]_" { policy= "write"

}

agent "[trigram]cslcli[region][env]_" { policy= "write"

}

agent "dckcslcli[region][env]_" { policy= "read"

}

session "[trigram]cslcli[region][env]_" { policy= "write"

}

session "dckcslcli[region][env]_" { policy= "write"

}

# —– Key/Value part —–

# Read/Write [trigram]/[env]/ key

key "[trigram]/[env]/" { policy = "write"

}

# —– Service part —–

# Read/Write [trigram]-[env]- services

service "[trigram]-[env]-" { policy = "write"

}

# —– Events part —–

# Read/Write [trigram]-[env]- events

event "[trigram]-[env]-" { policy="write"

}

As you can see on each applicative ACLs we are also providing some specific access to dck node. These nodes are used on the PaaS platform by Docker workers (see below)

27.3 Consul on Paas platform

Each Docker workers (where your applicatives containers are running) are hosting a local Consul client agent. This agent can be reach from your applicative container by using DNS alias :

- https://consul-agent.paas.local:8543

So you don't need to use a Consul agent client in a Container for your application, you have to use directly the local agent on the host to register or reach your applicative services.

Warning: The local agent is only reachable via https, hence you need to have the Unipass CA in your application (DEV CA for Dev, PROD CA for Prod)

Warning: Please do not use remotely the consul agents of the PaaS Plaform

27.4 Healthchecks

Warning: All of your services registered in Consul must be declared with a Health Check. Please also configure your healthcheck to return a static contain and not volatile datas like timestamp, cpu load, free memory, whereas it's spoiling raft protocol and Consul logs

- Usual parameter:
    - Name
    - *Name of your service in Consul*
    - Tags
    - *Your tags*
    - Address
    - *Address of your service*
    - ID
    - *Must be Unique. Put the ServiceName (HostName or ContainerID) + Port in it*
    - Port
    - *Port where your service answer*

- Check (**Mandatory**)
  - Name
  - *Must be Unique like ID. Put the ServiceName + (HostName or ContainerID) + Port in it*
  - Interval
  - *Interval of your check*
  - Timeout
  - *Timeout of your check*
  - DeregisterCriticalServiceAfter (**Mandatory**)
  - Deregister automatically your failing service after a time out. Must be superior to 1m (ripping process is launched every 30s)

Here is an example of a json file to register a service with a HTTP(S) healthcheck :

```
{
   "name": "dck-dev-http-check",
   "tags": ["my tags"],
   "address": "mywonderfulserver",
   "ID": "mywonderfulserver-dck-dev-http-8080",
   "port": 8080,
       "check": {
             "name": "check-dck-dev-http-8080",
             "http": "http://mywonderfulserver.fr.world.socgen:8080/",
             "interval": "10s",
             "timeout": "1s",
             "DeregisterCriticalServiceAfter":"10m"
             }
}
```

If you service is a **Container** use the specific check "docker_container". Here is an example :

```
{
   "name": "dck-dev-docker-check",
   "tags": ["my tags"],
   "address": "IPOfMyContainerInOverlayNetwork or ContainerID",
   "ID": "ContainerID-dck-dev-docker-9443",
   "port": 9443,
      "check": {
             "name": "check-dck-dev-docker-9443",
             "shell": "/bin/sh",
             "docker_container_id":"ContainerID",
             "script":"curl -f -k https://localhost:9443/health",
             "timeout": "1s"
             "interval": "10s",
             "DeregisterCriticalServiceAfter":"10m"
             }
}
```

Warning: Please use the DeregisterCriticalServiceAfter parameter to avoid zombies services in Consul due to restart/kill/stop/scale services

Warning: Communication between Fabio and Containers must be done in HTTPS. You have to set tlsskipverify=true to avoid TLS verification issues

If you are using Fabio you need to use specific tags “urlprefix-yourfqdn” and must end with a “/”. Here is an example :

```
{
   "name": "dck-dev-docker-check",
   "tags": ["urlprefix-dck-dev-docker.fr.world.socgen/ proto=https tlsskipverify=true"],
   "address": "IPOfMyContainerInOverlayNetwork",
   "ID": "ContainerID-dck-dev-docker-8080",
   "port": 8080,
      "check": {
             "name": "check-dck-dev-docker-8080",
             "shell": "/bin/sh",
             "docker_container_id":"149d9ba3573c",
             "script":"curl -f -k https://localhost:9443/health",
             "timeout": "1s"
             "interval": "10s",
             "DeregisterCriticalServiceAfter":"10m"
             }
}
```

Warning: On the PaaS platform, health check must be a script and not a URL or network related. This script will be executed by the consul agent directly in the container

You can find more information on Consul website : https://www.consul.io/docs/agent/checks.html

CHAPTER 28

FAQ

1. Force API version to work with UCP

If you ever find yourself with a Docker version greater than 1.10, you won't be able to interact with UCP-poc. You have to force the API version 1.22. To do so, you can run this command on your terminal

```
export DOCKER_API_VERSION=1.22
```

CHAPTER 29

ITEC links

1. Sgithub Training
2. Jenkins Training
3. Configuration management

Community

This list all the people as main contributors of the PaaS Docker Community (project & technical). Please also use the Jive SG-PaaS space.

Contact	Role
LAGREE Emmanuel	Owner of the PaaS platform - GTS
BOUSSARDON Thomas	Owner of the PaaS platform - GTS
PAR-RESG-GTS-Docker	Mailing list of GTS PaaS team
BIZET Guillaume	PaaS expert- ITEC
PAR-GTS-ITEC-Docker	Mailing list of PaaS Docker team
KLAT Christian	Producter owner of the PaaS platform - ITEC
DECHOUX Stephan	Proxy PO of the PaaS platform - ITEC
MARTIN Cédric	Project manager of PaaS stream within GTS
COQUELIN Thomas	Security expert - GTS
SEBIHI Yacine	Unix expert - GTS
NOMOREDJO Jean-Christophe	Unix expert - GTS
FOUCHET François	Storage expert - GTS