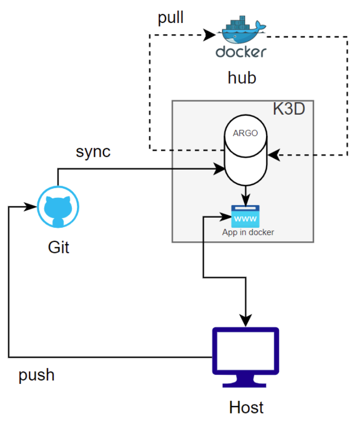
IV.3 Part 3: K3d and Argo CD

You now master a minimalist version of K3S! Time to set up everything you have just learnt (and much more!) but without Vagrant this time. To begin, install K3D on your virtual machine.

You will need Docker for K3d to work, and probably some other softwares too. Thus, you have to write a **script** to install every necessary packages and tools during your defense.

First of all, you must understand the difference between K3S and K3D.

Once your configuration works as expected, you can start to create your first **continuous integration**! To do so, you have to set up a small infrastructure following the logic illustrated by the diagram below:



You have to create two namespaces:

- The first one will be dedicated to Argo CD.
- The second one will be named *dev* and will contain an application. This application will be automatically deployed by Argo CD using your online Github repository.

Yes, indeed. You will have to create a public repository on Github where you will push your configuration files.
You are free to organize it the way you like. The only mandatory requirement is to put the login of a member of the group in the name of your repository.

The application that will be deployed must have **two different versions** (read about tagging if you don't know about it).

You have two options:

- You can use the ready-made application created by Wil. It's available on Dockerhub.
- Or you can code and use your own application. Create a public Dockerhub repository to push a Docker image of your application. Also, tag its two versions this way: **v1** and **v2**.

`You can find Wil's application on Dockerhub here:`
`https://hub.docker.com/r/wil42/playground.`
`The application uses the port 8888.`
`Find the two versions in the` *`TAG`* `section.`

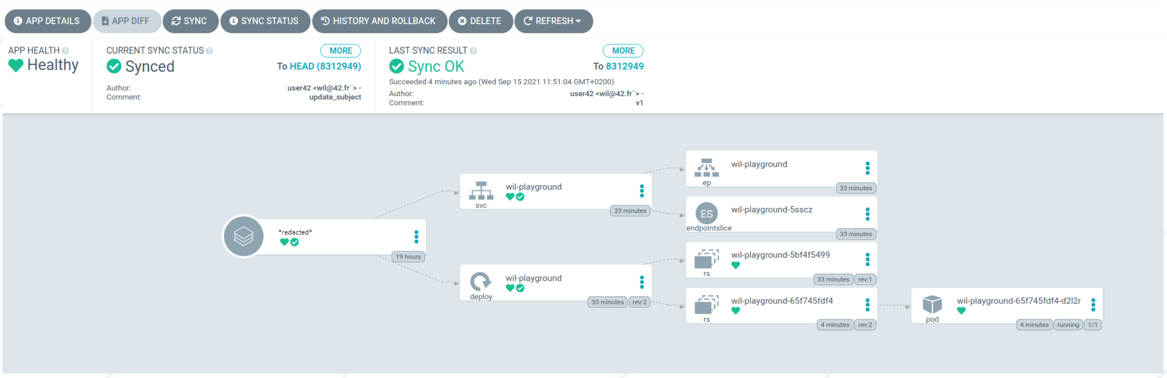
`If you decide to create your own application, it must be made available thanks to a public Docker image pushed into a Dockerhub repository. The two versions of your application must also have a few differences.`

You must be able to change the version from your public Github repository, then check that the application has been correctly updated.

Here is an example showing the two `namespaces` and the *POD* located in the *dev* `namespace`:

```
$> k get ns
NAME            STATUS   AGE
[..]
argocd          Active   19h
dev             Active   19h
$> k get pods -n dev
NAME                            READY   STATUS    RESTARTS   AGE
wil-playground-65f745fdf4-d2l2r 1/1     Running   0          8m9s
$>
```

Here is an example of launching `Argo CD` that was configured:



APP HEALTH
Healthy

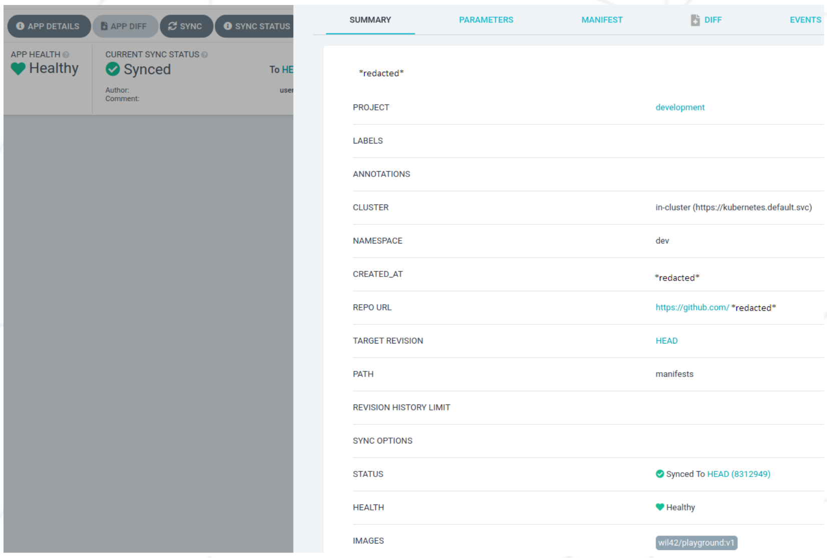
CURRENT SYNC STATUS
Synced
To HEAD (8312949)
Author: user42 <wil@42.fr¨> -
Comment: update_subject

LAST SYNC RESULT
Sync OK
To 8312949
Succeeded 4 minutes ago (Wed Sep 15 2021 11:51:04 GMT+0200)
Author: user42 <wil@42.fr¨> -
Comment: v1

We can check that our application uses the version we expect (in this case, the **v1**):

```
$> cat deployment.yaml | grep v1
      - image: wil42/playground:v1
$> curl http://localhost:8888/
{"status":"ok", "message": "v1"}
```

Here is a screenshot of `Argo CD` with our **v1** application using Github:



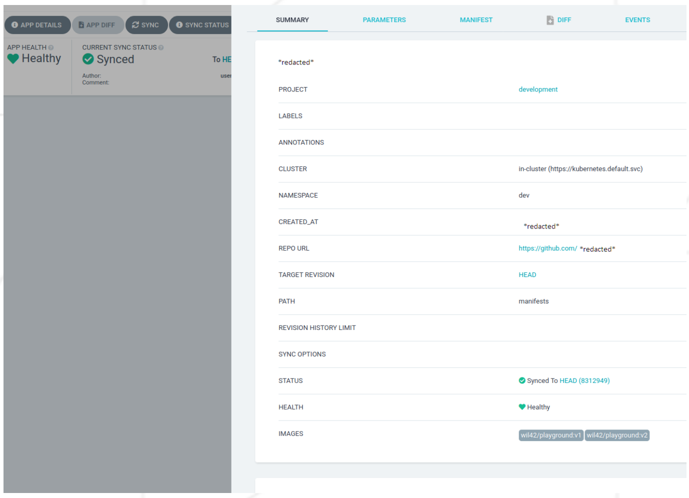
redacted

PROJECT	development
LABELS	
ANNOTATIONS	
CLUSTER	in-cluster (https://kubernetes.default.svc)
NAMESPACE	dev
CREATED_AT	*redacted*
REPO URL	https://github.com/ *redacted*
TARGET REVISION	HEAD
PATH	manifests
REVISION HISTORY LIMIT	
SYNC OPTIONS	
STATUS	Synced To HEAD (8312949)
HEALTH	Healthy
IMAGES	wil42/playground:v1

Below, we update our Github repository by changing the version of our application:

```
$>sed -i 's/wil42\/playground\:v1/wil42\/playground\:v2/g' deploy.yaml
$>g up "v2" # git add+commit+push
[..]
   a773f39..999b9fe master -> master
$> cat deployment.yaml | grep v2
  - image: wil42/playground:v2
```

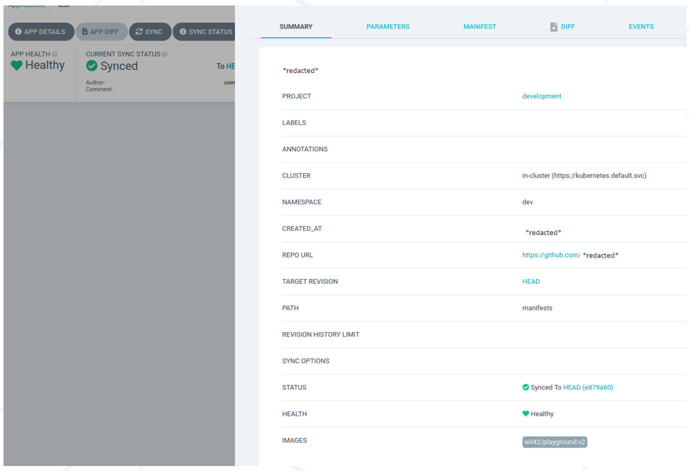
You can see thanks to `Argo CD` that the application is synchronized:



redacted

PROJECT	development
LABELS	
ANNOTATIONS	
CLUSTER	in-cluster (https://kubernetes.default.svc)
NAMESPACE	dev
CREATED_AT	*redacted*
REPO URL	https://github.com/ *redacted*
TARGET REVISION	HEAD
PATH	manifests
REVISION HISTORY LIMIT	
SYNC OPTIONS	
STATUS	Synced To HEAD (8312949)
HEALTH	Healthy
IMAGES	wil42/playground:v1 wil42/playground:v2

The application was updated with success:



redacted

PROJECT	development
LABELS	
ANNOTATIONS	
CLUSTER	in-cluster (https://kubernetes.default.svc)
NAMESPACE	dev
CREATED_AT	*redacted*
REPO URL	https://github.com/ *redacted*
TARGET REVISION	HEAD
PATH	manifests
REVISION HISTORY LIMIT	
SYNC OPTIONS	
STATUS	Synced To HEAD (e879a60)
HEALTH	Healthy
IMAGES	wil42/playground:v2

We check that the new version is available:

```
$> curl http://localhost:8888/
{"status":"ok", "message": "v2"}
```

During the evaluation process, you will have to do this operation with the app you chose: Wil's or yours.