



# Inception-of-Things ( IoT )

*Summary: This document is a System Administration related exercise.*

*Version: 4.0*

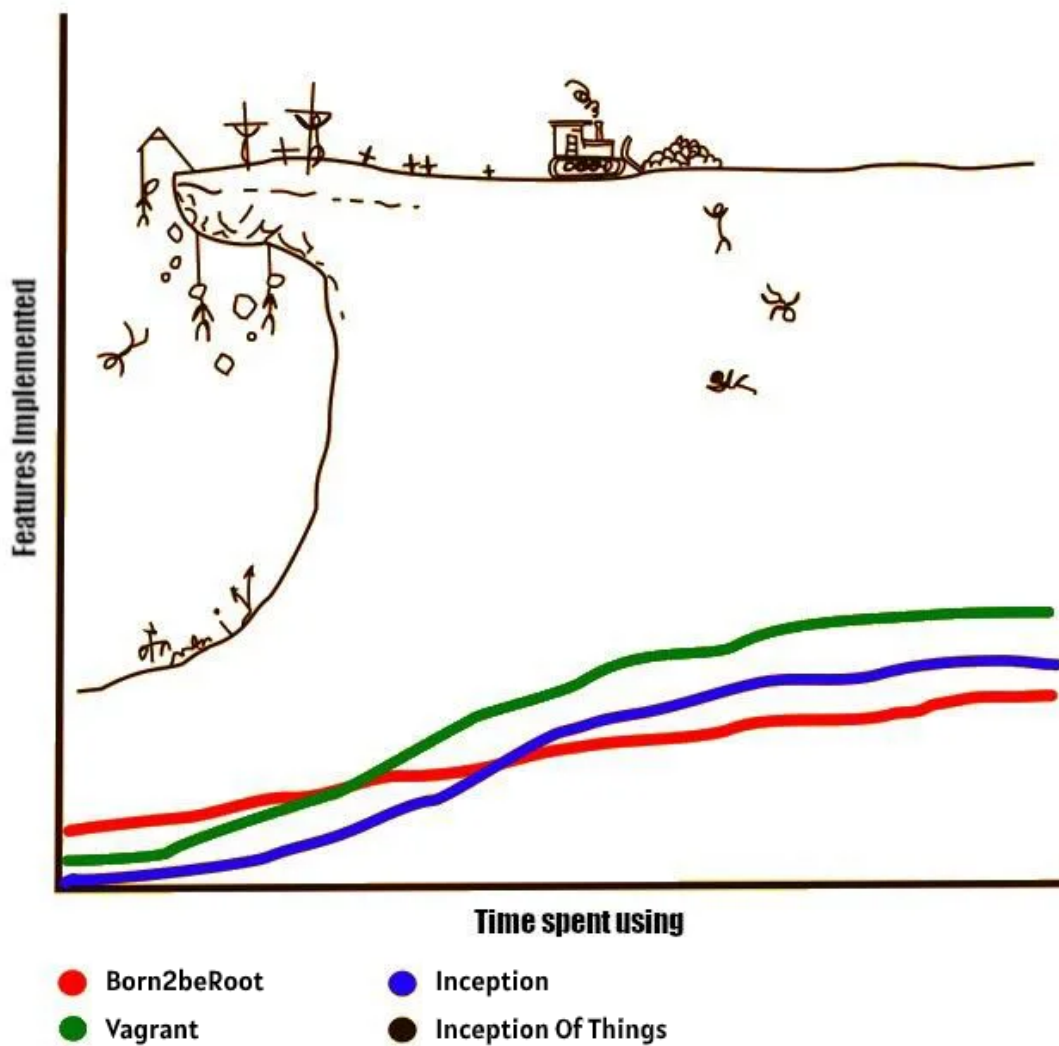
# Contents

<b>I</b>	<b>Preamble</b>	<b>2</b>
<b>II</b>	<b>Introduction</b>	<b>3</b>
<b>III</b>	<b>General guidelines</b>	<b>4</b>
<b>IV</b>	<b>Mandatory part</b>	<b>5</b>
IV.1	Part 1: K3s and Vagrant . . . . .	6
IV.2	Part 2: K3s and three simple applications . . . . .	9
IV.3	Part 3: K3d and Argo CD . . . . .	12
<b>V</b>	<b>Bonus part</b>	<b>16</b>
<b>VI</b>	<b>Submission and peer-evaluation</b>	<b>17</b>

# Chapter I

## Preamble

### Learning curves



# Chapter II

## Introduction

This project aims to deepen your knowledge by making you use K3d and K3s with Vagrant.

You will learn how to set up a personal virtual machine with Vagrant and the distribution of your choice. Then, you will learn how to use K3s and its Ingress. Last but not least, you will discover K3d that will simplify your life.

These steps will get you started with Kubernetes.



This project is a minimal introduction to Kubernetes. Indeed, this tool is too complex to be mastered in a single subject.

# Chapter III

## General guidelines

- The whole project has to be done in a **virtual machine**.
- You have to put all the configuration files of your project in folders located at the root of your repository (go to Submission and peer-evaluation for more information). The folders of the mandatory part will be named: **p1**, **p2** and **p3**, and the bonus one: **bonus**.
- This topic requires you to apply concepts that, depending on your background, you may not have covered yet. We therefore advise you not to be afraid to read a lot of documentation to learn how to use **K8s** with **K3s**, as well as **K3d**.



You can use any tools you want to set up your host virtual machine as well as the provider used in Vagrant.

# Chapter IV

## Mandatory part

This project will consist of setting up several environments under specific rules.

It is divided into three parts you have to do in the following order:

- Part 1: K3s and Vagrant
- Part 2: K3s and three simple applications
- Part 3: K3d and Argo CD

## IV.1 Part 1: K3s and Vagrant

To begin, you have to set up 2 **machines**.

Write your first **Vagrantfile** using the **latest stable version** of the distribution of your choice as your operating system. It is **STRONGLY** advised to allow only the bare minimum in terms of resources: 1 CPU and 512 MB of RAM (or 1024). The machines must be run using **Vagrant**.

Here are the expected specifications:

- The machine names must be the login of someone from your team. The hostname of the first machine must be followed by the capital letter S (like *Server*). The hostname of the second machine must be followed by SW (like *ServerWorker*).
- Have a dedicated IP on the primary network interface. The IP of the first machine (*Server*) will be 192.168.56.110, and the IP of the second machine (*ServerWorker*) will be 192.168.56.111.
- Be able to connect with SSH on both machines with no password.



You will set up your Vagrantfile according to modern practices.

You must install **K3s** on both machines:

- In the first one (*Server*), it will be installed in controller mode.
- In the second one (*ServerWorker*), in agent mode.



You will have to use **kubectl** (and therefore install it as well).

## Inception-of-Things ( IoT )

---

Here is a **basic** example of a Vagrantfile:

```
$> cat Vagrantfile
Vagrant.configure(2) do |config|
  [...]
  config.vm.box = REDACTED
  config.vm.box_url = REDACTED

  config.vm.define "wils" do |control|
    control.vm.hostname = "wils"
    control.vm.network REDACTED, ip: "192.168.56.110"
    control.vm.provider REDACTED do |v|
      v.customize ["modifyvm", :id, "--name", "wils"]
      [...]
    end
    config.vm.provision :shell, :inline => SHELL
    [...]
    SHELL
    control.vm.provision "shell", path: REDACTED
  end
  config.vm.define "wilsW" do |control|
    control.vm.hostname = "wilsW"
    control.vm.network REDACTED, ip: "192.168.56.111"
    control.vm.provider REDACTED do |v|
      v.customize ["modifyvm", :id, "--name", "wilsW"]
      [...]
    end
    config.vm.provision "shell", inline: <<-SHELL
    [...]
    SHELL
    control.vm.provision "shell", path: REDACTED
  end
end
end
```

## Inception-of-Things (IoT)

---

Here is an example when the virtual machines are launched:

```
→ p1 vagrant up
Bringing machine 'wils' up with 'virtualbox' provider...
Bringing machine 'wilsW' up with 'virtualbox' provider...
[...]
→ p1 vagrant ssh wils           → p1 vagrant ssh wilsW
[vagrant@wils ~]$              [vagrant@wilsW ~]$
```

Here is an example when the configuration is not complete:

```
[vagrant@wils ~]$ k get nodes -o wide
NAME STATUS ROLES AGE VERSION INTERNAL-IP EXTERNAL-IP OS-IMAGE KERNEL-VERSION CONTAINER-RUNTIME
wils Ready control-plane,master 4m37s v1.21.4+k3s1 192.168.56.110 <none> CentOS Linux 8 4.18.0-240.1.1.el8_3.x86_64 containerd://1.4.9-k3s1
[vagrant@wils ~]$ ifconfig eth1
eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
inet 192.168.42.110 netmask 255.255.255.0 broadcast 192.168.56.255
inet6 fe80::a00:27ff:fe79:56d8 prefixlen 64 scopeid 0x20<link>
ether 08:00:27:79:56:d8 txqueuelen 1000 (Ethernet)
RX packets 10 bytes 2427 (2.3 KiB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 28 bytes 3702 (3.6 KiB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Here is an example when the machines are correctly configured:

```
[vagrant@wils ~]$ k get nodes -o wide
NAME STATUS ROLES AGE VERSION INTERNAL-IP EXTERNAL-IP OS-IMAGE KERNEL-VERSION CONTAINER-RUNTIME
wils Ready control-plane,master 16m v1.21.4+k3s1 192.168.56.110 <none> CentOS Linux 8 4.18.0-240.1.1.el8_3.x86_64 containerd://1.4.9-k3s1
wilsW Ready <none> 78s v1.21.4+k3s1 192.168.56.111 <none> CentOS Linux 8 4.18.0-240.1.1.el8_3.x86_64 containerd://1.4.9-k3s1
[vagrant@wils ~]$
[vagrant@wilsW ~]$ ifconfig eth1
eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
inet 192.168.42.111 netmask 255.255.255.0 broadcast 192.168.56.255
inet6 fe80::a00:27ff:fea8:bc4 prefixlen 64 scopeid 0x20<link>
ether 08:00:27:a8:bc:b4 txqueuelen 1000 (Ethernet)
RX packets 446 bytes 322199 (314.6 KiB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 472 bytes 101181 (98.8 KiB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
[vagrant@wilsW ~]$
```



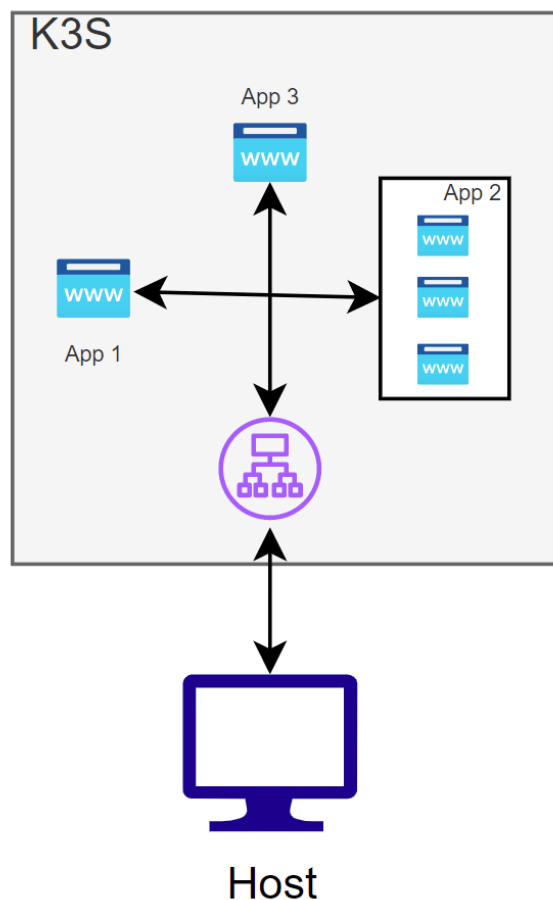
The screenshots above are examples only. Modern Linux distributions use predictable network interface names (e.g., enp0s8, enp0s9) instead of eth0/eth1. To check your network configuration, use: `ip a` to list all interfaces, or `ip a show <interface_name>` for a specific interface. On macOS, use `ifconfig`. Adapt the commands according to your system's actual interface names.

## IV.2 Part 2: K3s and three simple applications

You now understand the basics of K3s. Time to go further! To complete this part, you will need only one virtual machine with the distribution of your choice (**latest stable version**) and K3s in server mode installed.

You will set up 3 web applications of your choice that will run in your K3s instance. You will have to be able to access them depending on the **HOST** used when making a request to the IP address 192.168.56.110. The name of this machine will be your login followed by S (e.g., *wilS* if your login is *wil*).

Here is a simple example diagram:



When a client inputs the IP address 192.168.56.110 in their web browser with the **HOST** *app1.com*, the server must display app1. When the **HOST** *app2.com* is used, the server must display app2. Otherwise, app3 will be selected by default.



As you can see, application number 2 has 3 replicas. Adapt your configuration to create the replicas.

# Inception-of-Things ( IoT )

---

First, here is an expected result when the virtual machine is not configured:

```
[vagrant@wils ~]$ k get nodes -o wide
NAME      STATUS    ROLES    AGE   VERSION    INTERNAL-IP    EXTERNAL-IP    OS-IMAGE    KERNEL-VERSION    CONTAINER-RUNTIME
wils     Ready    control-plane,master  14m   v1.21.4+k3s1  192.168.56.110 <none>        CentOS Linux 8  4.18.0-240.1.1.el8_3.x86_64  containerd://1.4.9-k3s1
[vagrant@wils ~]$ k get all -n kube-system
NAME      READY   STATUS    RESTARTS   AGE
pod/metrics-server-86cbb8457f-69zx4  0/1     ContainerCreating  0           14m
pod/local-path-provisioner-5ff76fc89d-p7g5b  0/1     ContainerCreating  0           14m
pod/coredns-7448499f4d-jwlpt  0/1     ContainerCreating  0           14m
pod/helm-install-traefik-crd-wkn88  0/1     ContainerCreating  0           14m
pod/helm-install-traefik-82sqz  0/1     ContainerCreating  0           14m

NAME      TYPE          CLUSTER-IP    EXTERNAL-IP    PORT(S)          AGE
service/kube-dns  ClusterIP     10.43.0.10    <none>         53/UDP,53/TCP,9153/TCP  14m
service/metrics-server  ClusterIP     10.43.89.169 <none>         443/TCP           14m

NAME      READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/local-path-provisioner  0/1     1             0           14m
deployment.apps/coredns  0/1     1             0           14m
deployment.apps/metrics-server  0/1     1             0           14m

NAME      DESIRED   CURRENT   READY   AGE
replicaset.apps/metrics-server-86cbb8457f  1         1         0       14m
replicaset.apps/local-path-provisioner-5ff76fc89d  1         1         0       14m
replicaset.apps/coredns-7448499f4d  1         1         0       14m

NAME      COMPLETIONS   DURATION   AGE
job.batch/helm-install-traefik  0/1          14m       14m
job.batch/helm-install-traefik-crd  0/1          14m       14m
[vagrant@wils ~]$
```

## Inception-of-Things (IoT)

Here is an expected result when the virtual machine is correctly configured:

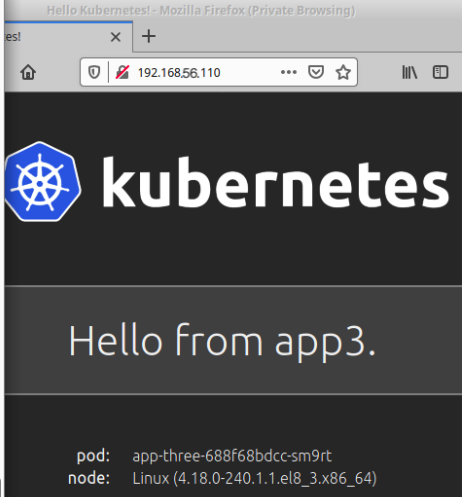
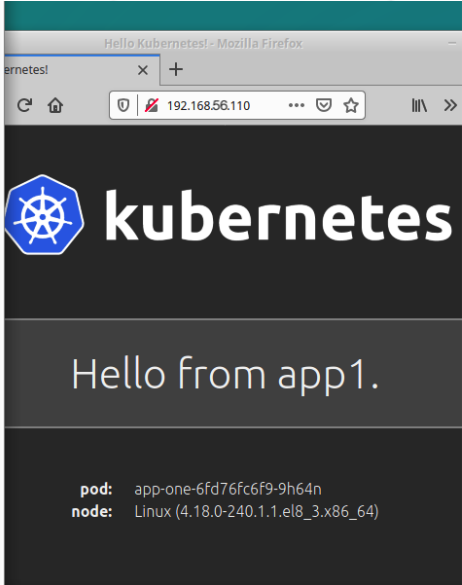
```
[vagrant@wils de]$ k get all
NAME                                READY   STATUS    RESTARTS   AGE
pod/app-two-6bc974bc98-qtjj7        1/1     Running  0           15m
pod/app-one-6fd76fc6f9-9h64n        1/1     Running  0           15m
pod/app-three-688f68bdcc-sm9rt      1/1     Running  0           15m
pod/app-two-6bc974bc98-nzwtb        1/1     Running  0           15m
pod/app-two-6bc974bc98-qhp6p        1/1     Running  0           15m

NAME                                TYPE          CLUSTER-IP   EXTERNAL-IP   PORT(S)    AGE
service/kubernetes                  ClusterIP     10.43.0.1    <none>        443/TCP    16m
service/app-three                   ClusterIP     10.43.229.156 <none>        80/TCP     15m
service/app-two                     ClusterIP     10.43.193.160 <none>        80/TCP     5m2s
service/app-one                     ClusterIP     10.43.171.213 <none>        80/TCP     4m45s

NAME                                READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/app-two              3/3     3             3           15m
deployment.apps/app-three            1/1     1             1           15m
deployment.apps/app-one              1/1     1             1           15m

NAME                                DESIRED   CURRENT   READY   AGE
replicaset.apps/app-one-6fd76fc6f9  1         1         1       15m
replicaset.apps/app-three-688f68bdcc 1         1         1       15m
replicaset.apps/app-two-6bc974bc98    3         3         3       15m
[vagrant@wils de]$ curl -H "Host:app2.com" 192.168.56.110
<!DOCTYPE html>
<html>
<head>
<title>Hello Kubernetes!</title>
<link rel="stylesheet" type="text/css" href="/css/main.css">
<link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Ubuntu:300">
</head>
<body>
<div class="main">

<div class="content">
<div id="message">
Hello from app2.
</div>
<div id="info">
<table>
<tr>
<th>pod:</th>
<td>app-two-6bc974bc98-qtjj7</td>
</tr>
<tr>
<th>node:</th>
<td>Linux (4.18.0-240.1.1.el8_3.x86_64)</td>
</tr>
</table>
</div>
</div>
</div>
</body>
</html>[vagrant@wils de]$
```



The Ingress is not displayed here on purpose. You will have to show it to your evaluators during your defense.

## IV.3 Part 3: K3d and Argo CD

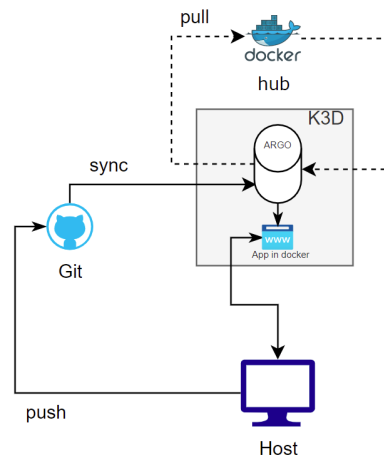
You now master a minimalist version of K3s! Time to set up everything you have just learnt (and much more!) but without Vagrant this time. To begin, install K3d on your virtual machine.



You will need Docker for K3d to work, and probably some other software as well. Therefore, you must write a script to install all the 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.

## Inception-of-Things ( IoT )

---

The application to be deployed must have **two different versions** (read about tagging if you are unfamiliar with it).

You have two options:

- You can use the pre-made application created by Wil, which is available on Docker Hub.
- Or you can code and use your own application. Create a public Docker Hub 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 Docker Hub here:  
<https://hub.docker.com/r/wil42/playground>.  
The application uses 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 Docker Hub 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
$>
```

## Inception-of-Things ( IoT )

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



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"}
```

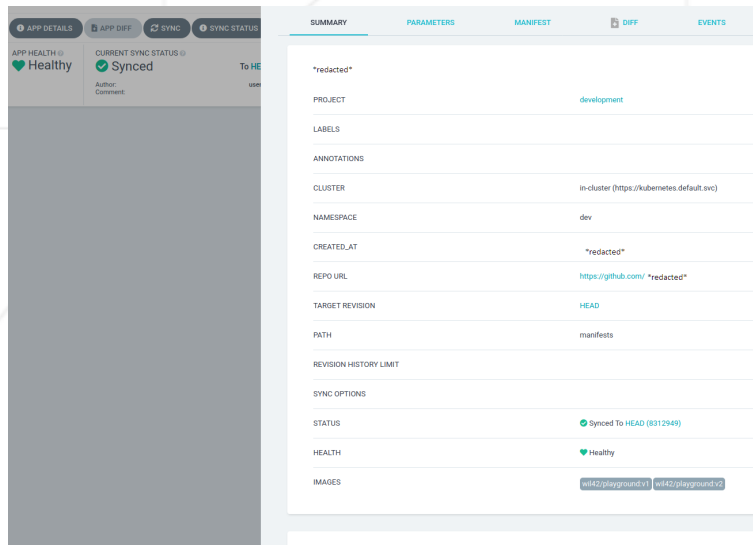
Here is a screenshot of Argo CD with our **v1** application using GitHub:

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

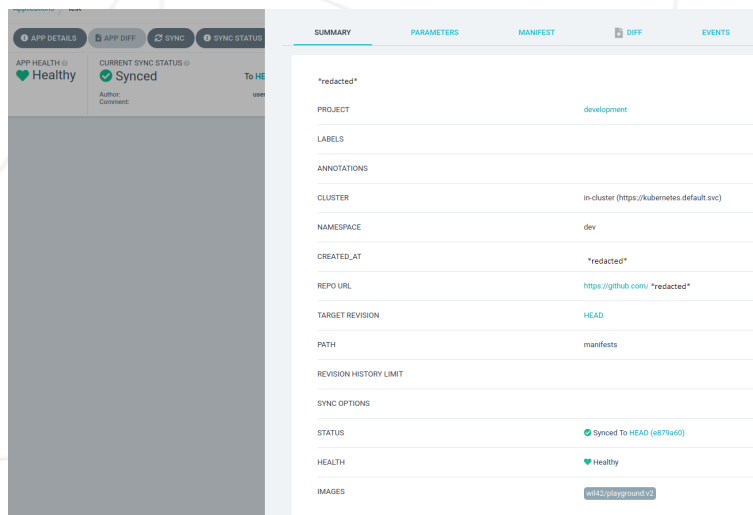
```
$> sed -i 's/wil42/playground\v1/wil42/playground\v2/g' deployment.yaml
$> git up "v2" # git add+commit+push
[.]
a773f39..999b9fe master -> master
$> cat deployment.yaml | grep v2
- image: wil42/playground:v2
```

## Inception-of-Things ( IoT )

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



The application was successfully updated:



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.

# Chapter V

## Bonus part

The following bonus task is intended to be useful: add **Gitlab** to the lab you completed in Part 3.



Beware this bonus is complex. The latest version available of Gitlab from the official website is expected.

You are allowed to use whatever you need to achieve this extra. For example, `helm` could be useful here.

- Your Gitlab instance must run locally.
- Configure Gitlab to make it work with your cluster.
- Create a dedicated `namespace` named `gitlab`.
- Everything you did in Part 3 must work with your local Gitlab.

Turn this extra work in a new folder named `bonus` and located at the root of your repository. You can add everything needed so your entire cluster works.



The bonus part will only be assessed if the mandatory part is flawless. Flawless means the mandatory part has been fully completed and functions without issues. If you have not passed ALL the mandatory requirements, your bonus part will not be evaluated at all.

# Chapter VI

## Submission and peer-evaluation

Turn in your assignment in your `Git` repository as usual. Only the work inside your repository will be evaluated during the defense. Don't hesitate to double-check the names of your folders and files to ensure they are correct.

### Reminder:

- Turn the mandatory part in three folders located at the root of your repository: `p1`, `p2` and `p3`.
- Optional: Turn the bonus part in a folder located at the root of your repository: `bonus`.

Below is an example of the expected directory structure:

```
$> find -maxdepth 2 -ls
424242    4 drwxr-xr-x  6 wandre wil42    4096 sept. 17 23:42 .
424242    4 drwxr-xr-x  3 wandre wil42    4096 sept. 17 23:42 ./p1
424242    4 -rw-r--r--  1 wandre wil42    XXXX sept. 17 23:42 ./p1/Vagrantfile
424242    4 drwxr-xr-x  2 wandre wil42    4096 sept. 17 23:42 ./p1/scripts
424242    4 drwxr-xr-x  2 wandre wil42    4096 sept. 17 23:42 ./p1/confs
424242    4 drwxr-xr-x  3 wandre wil42    4096 sept. 17 23:42 ./p2
424242    4 -rw-r--r--  1 wandre wil42    XXXX sept. 17 23:42 ./p2/Vagrantfile
424242    4 drwxr-xr-x  2 wandre wil42    4096 sept. 17 23:42 ./p2/scripts
424242    4 drwxr-xr-x  2 wandre wil42    4096 sept. 17 23:42 ./p1/confs
424242    4 drwxr-xr-x  3 wandre wil42    4096 sept. 17 23:42 ./p3
424242    4 drwxr-xr-x  2 wandre wil42    4096 sept. 17 23:42 ./p3/scripts
424242    4 drwxr-xr-x  2 wandre wil42    4096 sept. 17 23:42 ./p3/confs
424242    4 drwxr-xr-x  3 wandre wil42    4096 sept. 17 23:42 ./bonus
424242    4 -rw-r--r--  1 wandre wil42    XXXX sept. 17 23:42 ./bonus/Vagrantfile
424242    4 drwxr-xr-x  2 wandre wil42    4096 sept. 17 23:42 ./bonus/scripts
424242    4 drwxr-xr-x  2 wandre wil42    4096 sept. 17 23:42 ./bonus/confs
```



Any scripts you need will be added in a `scripts` folder. The configuration files will be in a `confs` folder.



The evaluation process will happen on the computer of the evaluated group.