Continuous Delivery using Jenkins + Docker + Consul + Fabio

Goal: To deploy a Continuous Integration(CI) & Continuous Delivery(CD) environment t**o** automatically deploy, test and release our web application on a docker cluster. There are many CI and CD tools to launch your own application. In our example, we use Jenkins + Docker + Shipyard + Git + Consul + Fabio to build our CI & CD environment. We will talk about why we use these tools and how do we use them in the following article.

What is CI & CD? Why we need to use it?

Simply speaking, it makes use of some automated tools and helps you deploy the environment of your app to let you focus on your code itself. More details refer to

What is continuous delivery:

https://www.atlassian.com/continuous-delivery

https://aws.amazon.com/devops/continuous-delivery/

CI Vs CD: https://www.atlassian.com/continuous-delivery/ci-vs-ci-vs-cd.

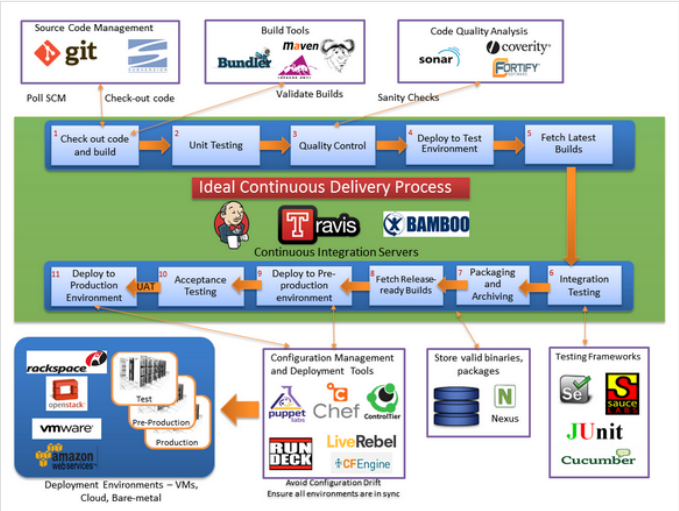
What are the popular choices for CI & CD?

https://stackify.com/top-continuous-integration-tools/

I will try AWS's pipeline and Atlassian's bamboo the next time.

Why do we choose Jenkins?

Free and many plugins.



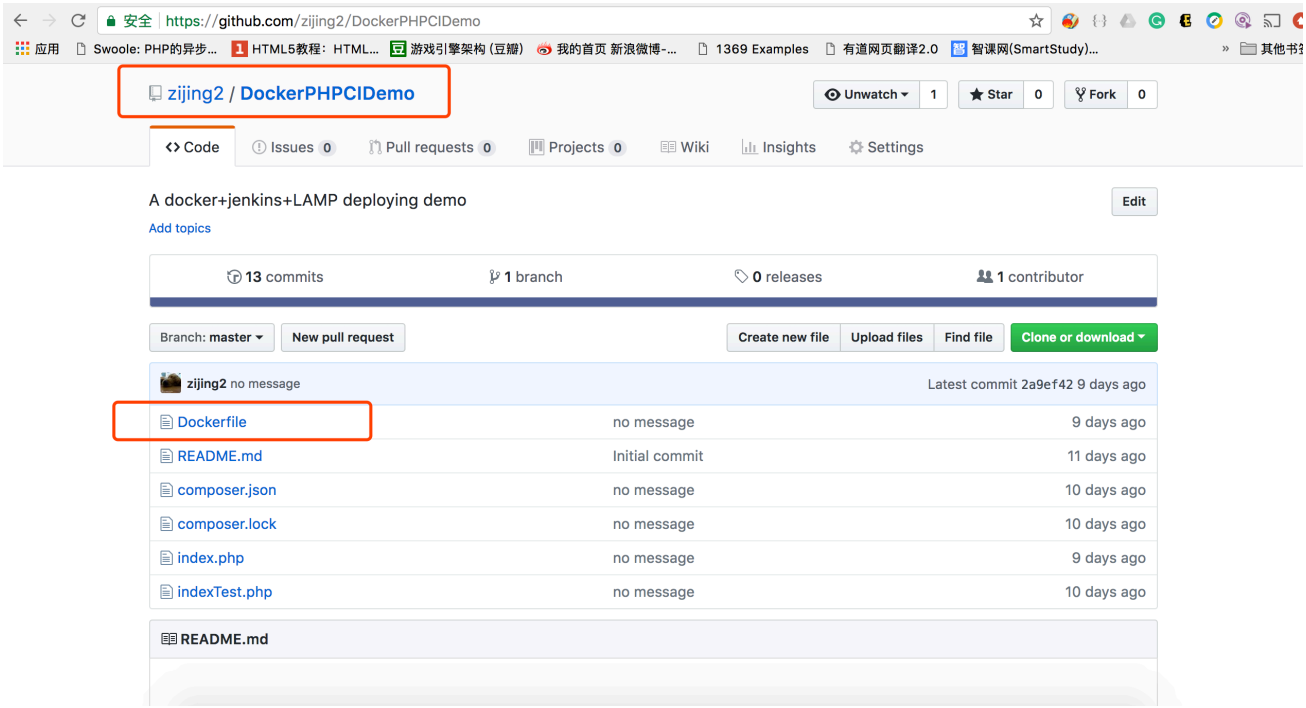
Before starting, you may know first.

In order to deploy a CD environment, we need to know the basic usage of the following tools:

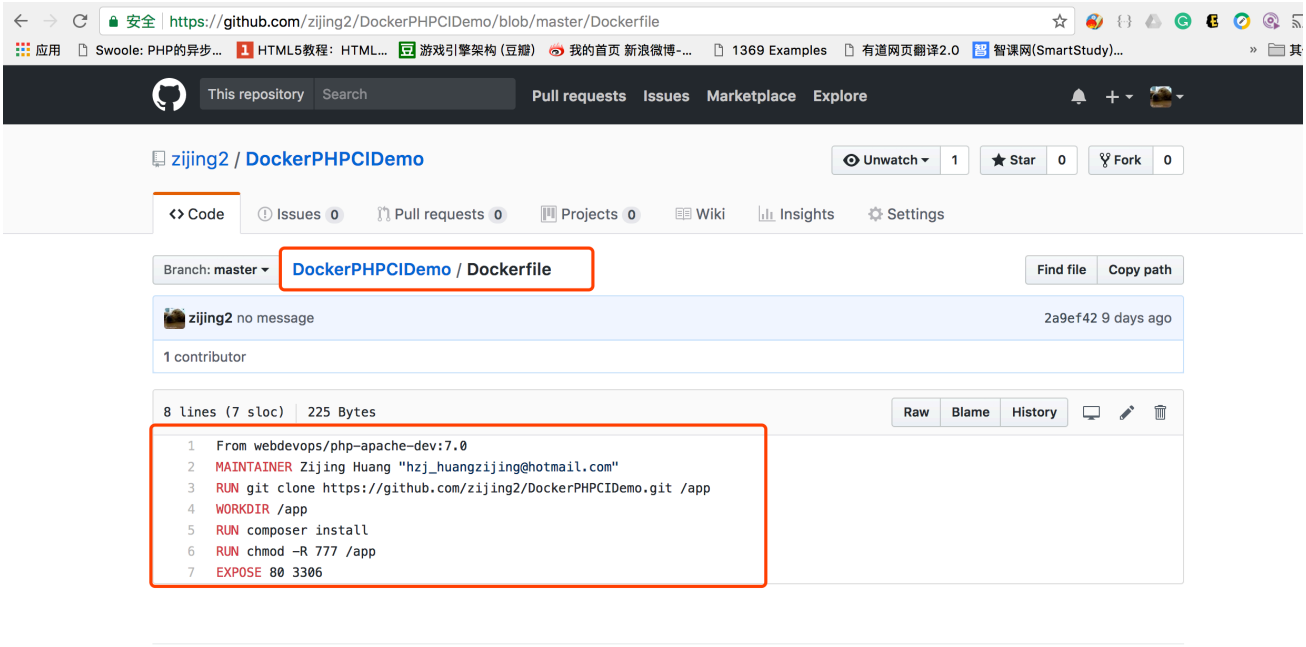
1. Jenkins: including Jenkins plugin, Jenkins jobs and Jenkins pipeline.
2. Git: the basic usage of Git such as clone, pull and push etc.
3. Docker: need to know the usage of image, container and dockerfile etc.
4. DockerHub: we will need to use the automated build in dockerhub.
5. Shipyard: A GUI for managing the docker container, image and cluster based on swarm.
6. Swarm: A distributed management system for docker. It's just like K8s.
7. Consul: need to deploy and register service.
8. Fabio: need to register load balance.

Deploying Details

1. We need to create a project and push it to GitHub. We create a web App based on LAMP this time.

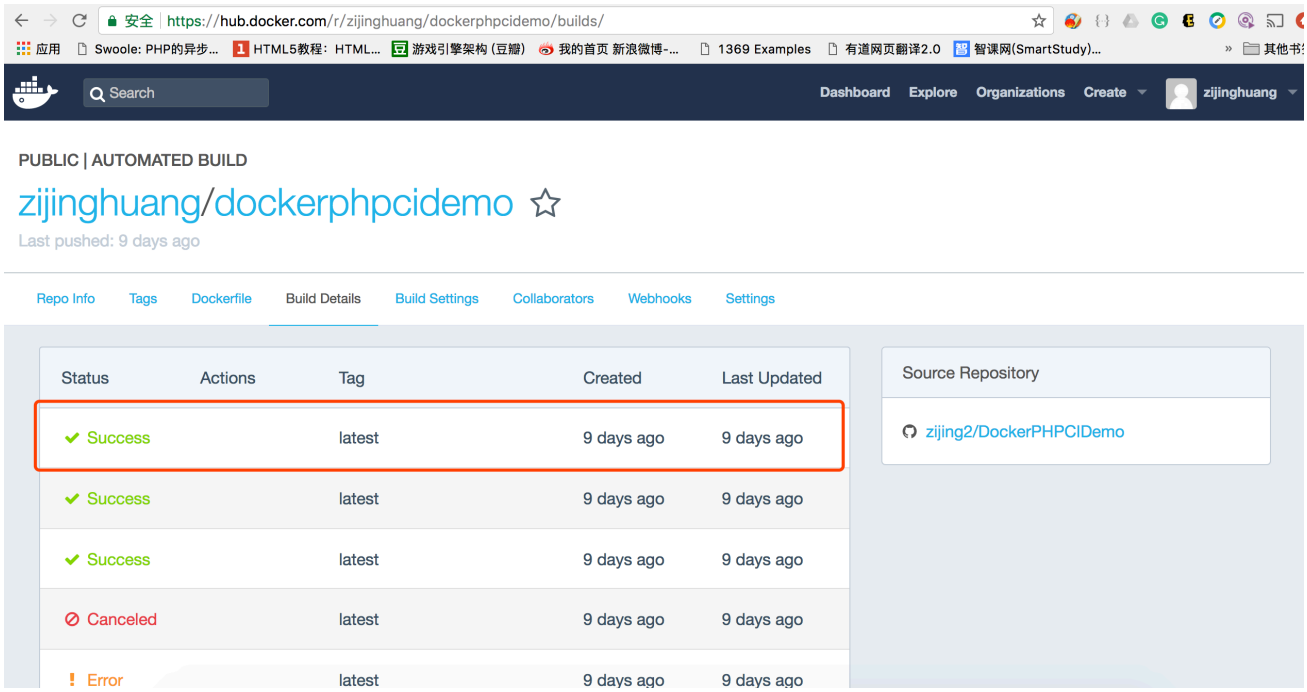


2. We will include a dockerfile which describe a docker image(environment) for our app.

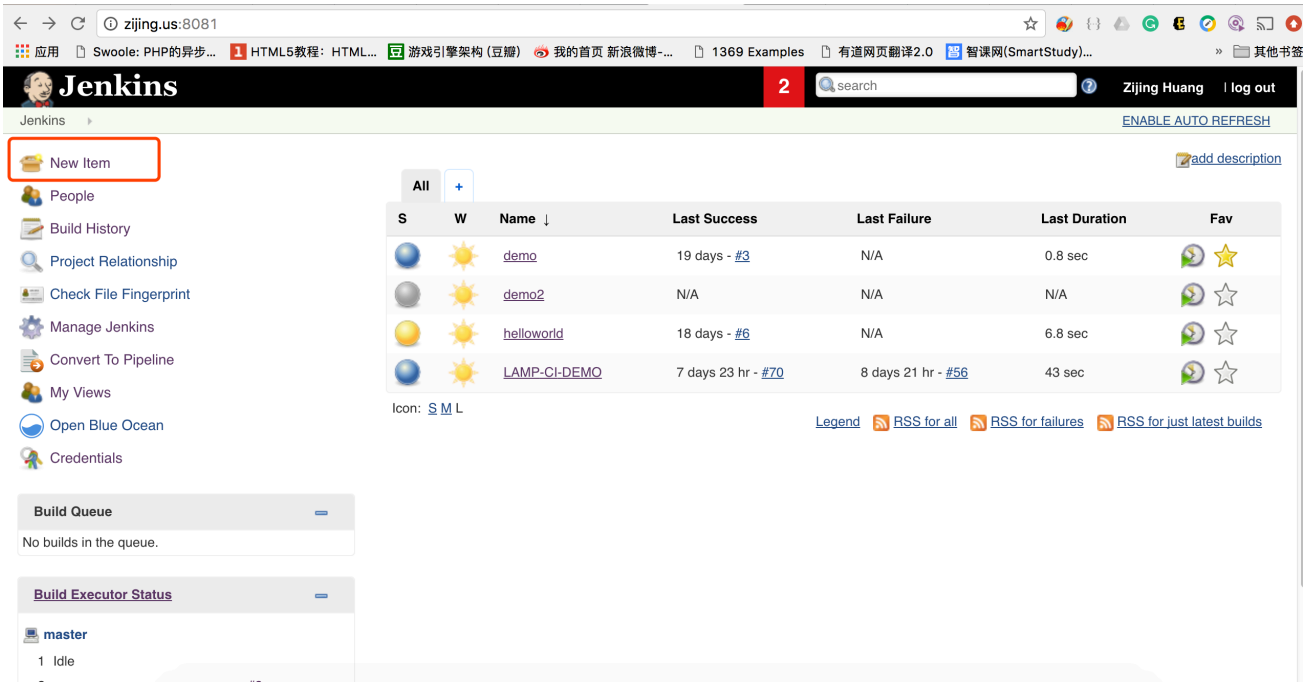


3. Create an automated build project on your dockerhub and map it to the project that we create previously on Git.

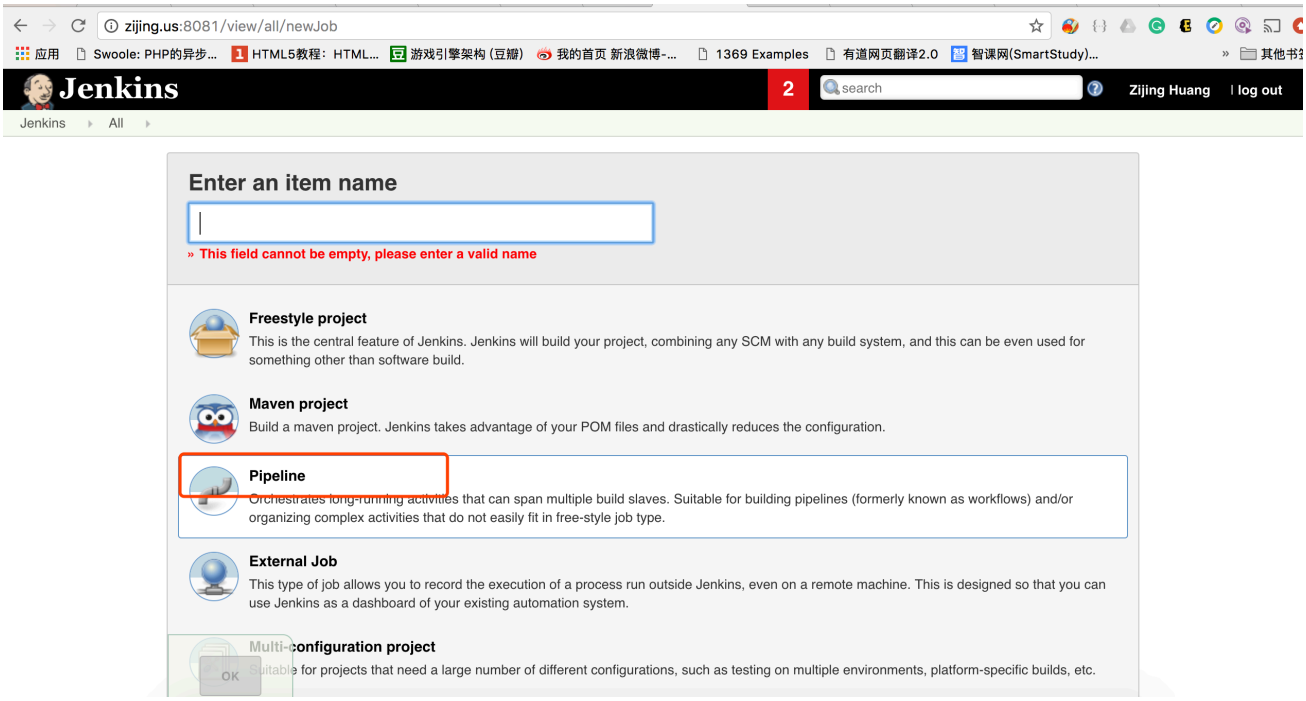
The docker image will be built automatically based on the dockerfile on your project each time you push your code on Git.



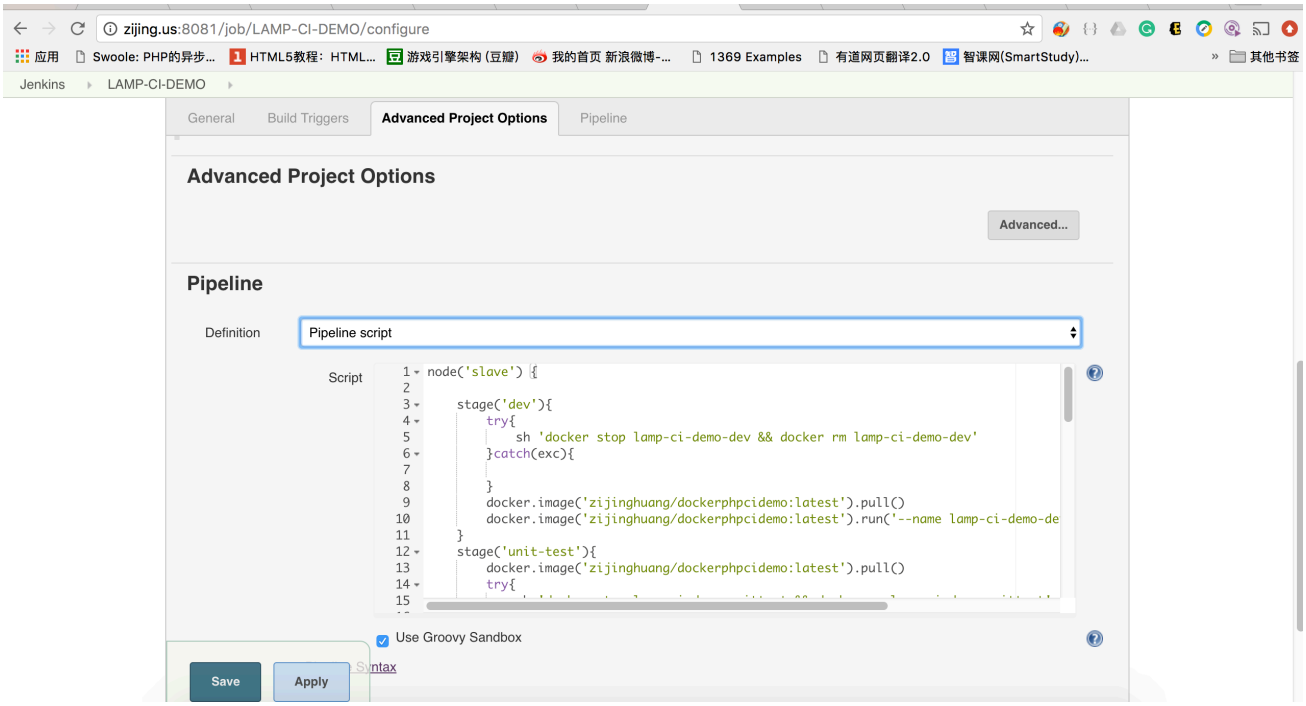
4. Install and configure Jenkins



If you want to create a Jenkins job, such as deploying a single development environment, then click on Freestyle project. We use pipeline to create a process of releasing our web app in this time.

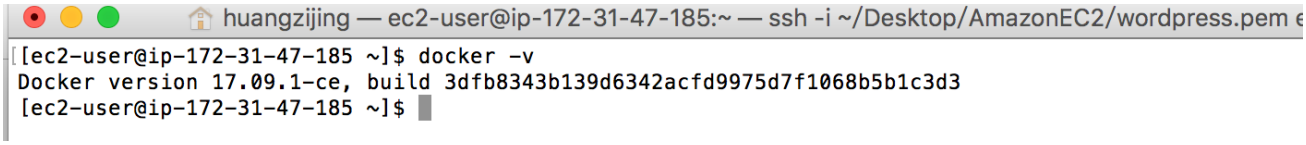


Use pipeline script to describe what you want to do in the remote server(Jenkins node) to deploy your environment in each stage.

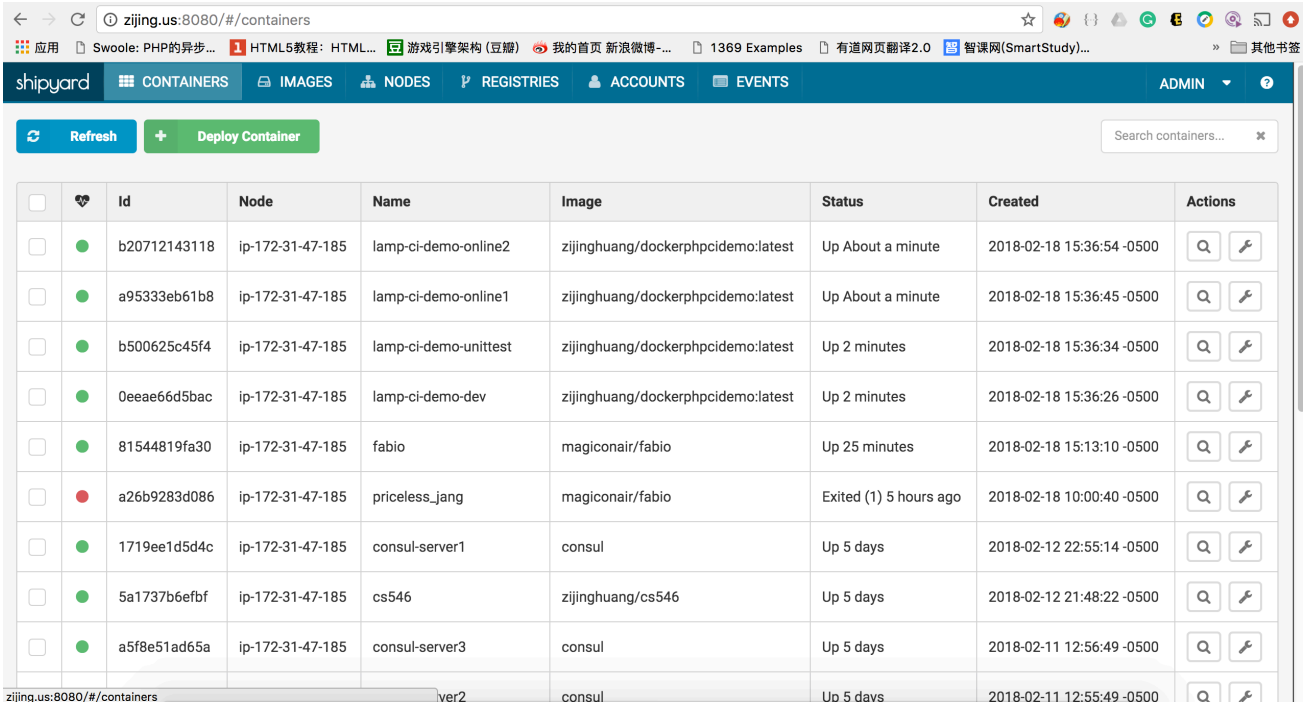


https://github.com/zijing2/DockerPHPCIDemo/blob/master/Jenkinsfile

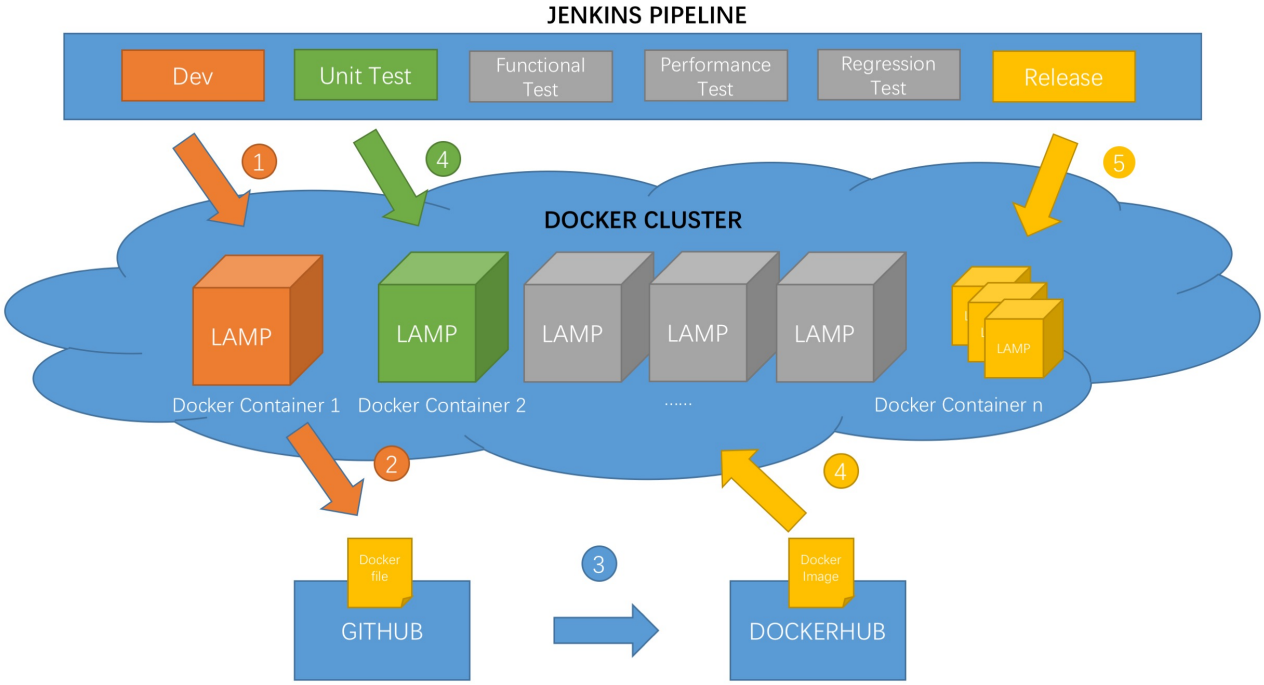
5. Make sure you have installed docker in the Jenkins node that you pointed to.



and I installed shipyard to manage images and containers on this server of the docker cluster.



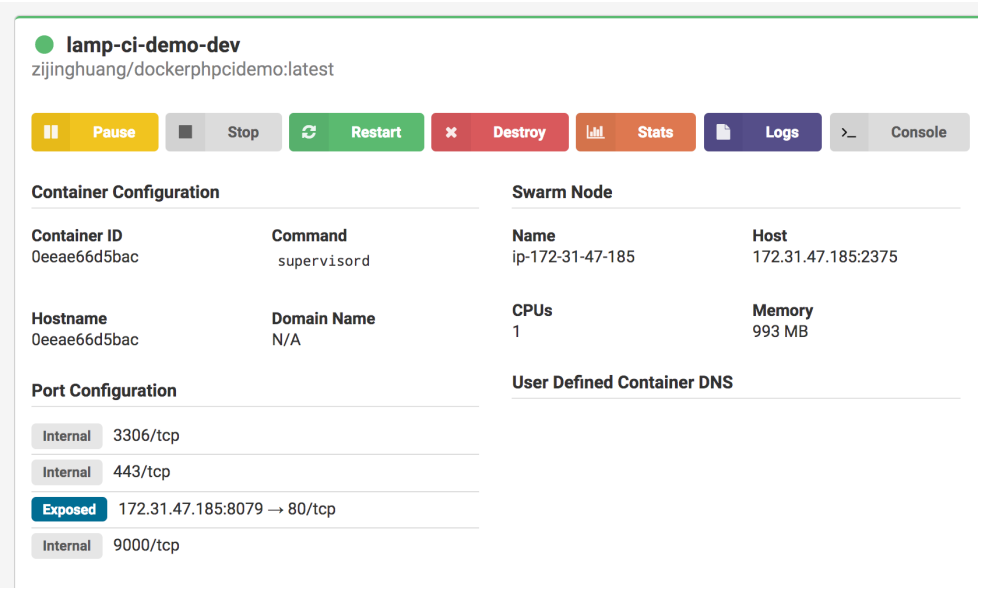
6. The Software release process is as following:

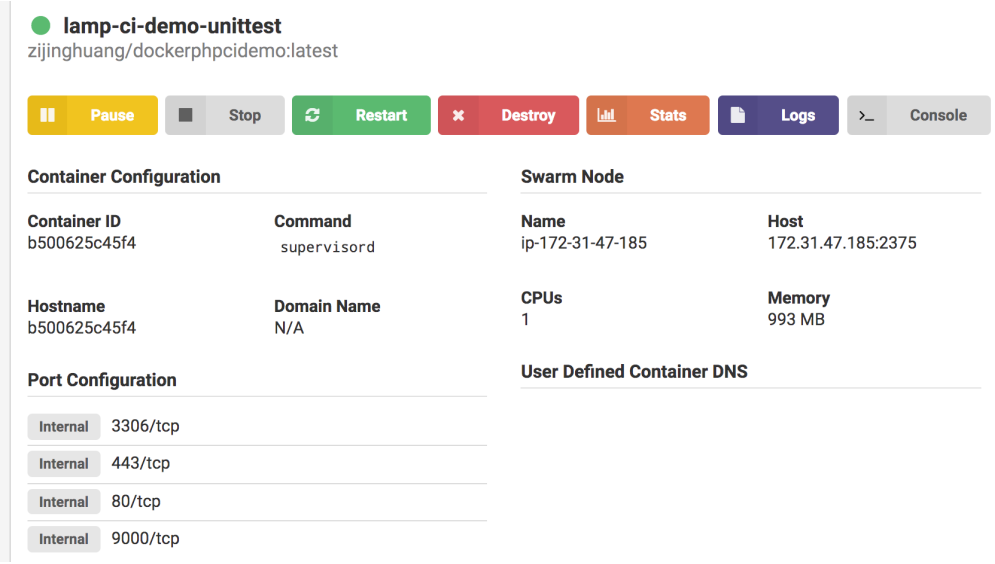


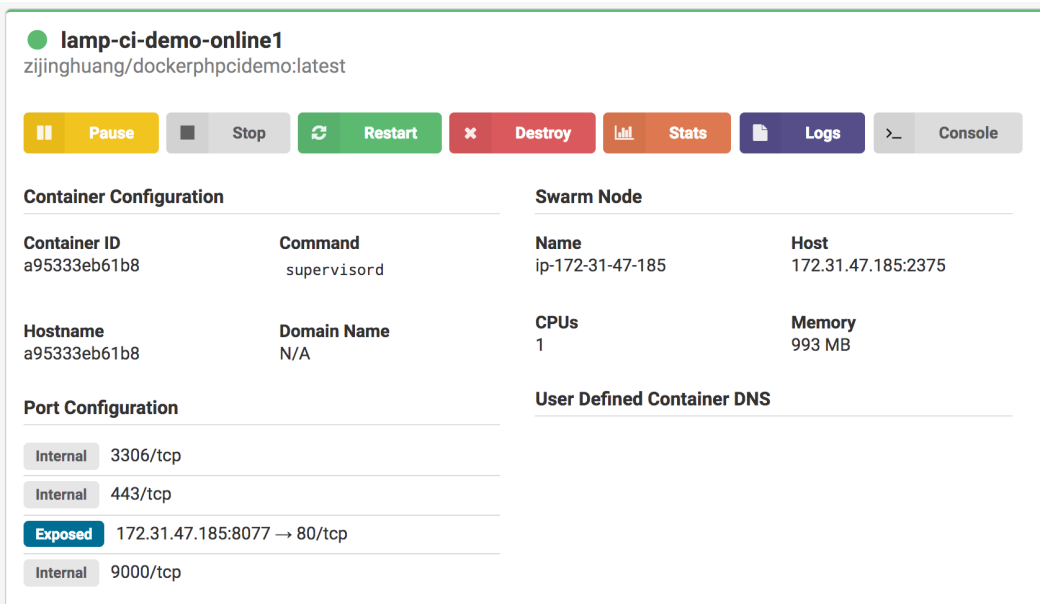
For 1. We use Jenkins to deploy our development environment.

For 2,3,4. We edit the code and push it to Github, which will triger the automated build on dockerhub. Eventually, we get the updated image.

Then we can rebuild the whole process: build the development env – build the test env and automated test – build the release env. Finally, we will get the following containers.





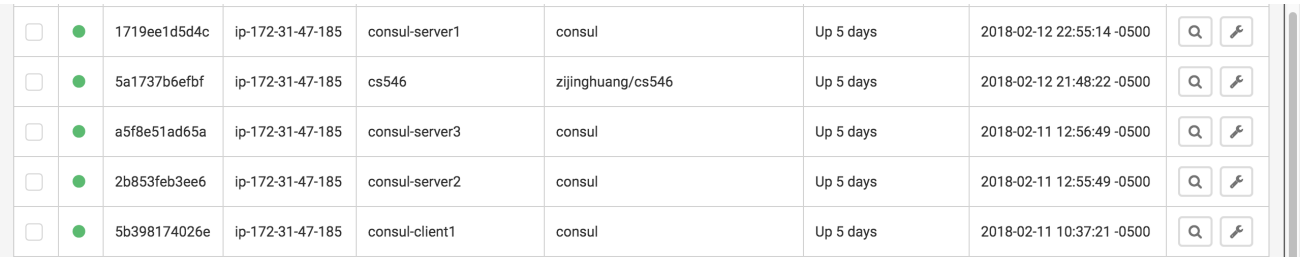


So far, we've already talked about the deploying process. But how do we access to the environment deployed?

Each time we start a docker container, we get a new private IP for it. So, we can't use the same IP for accessing the environment. To solve this problem, we need to use a services discover tool. Therefore, we use Consul in this case. What's more, we use Fabio as a load balancer in our example.

Consul Docker Image:

Fabio Docker Image:



Consul is based on a distributed system. The official document of consul suggests us using no less than 3 servers to maintain the datacenter. So, we start 3 servers and 1 client to do the service discovering.

For the first server of Consul:

UID	PID	PPID	C	STIME	TTY	TIME	CMD
root	8633	8619	0	Feb13	?	00:00:00	/usr/bin/dumb-init /bin/sh /usr/local/bin/docker-entrypoint.sh agent -data-dir=/consul/data -config-dir=/consul/config -client 0.0.0.0 -server -bootstrap -ui
100	8676	8633	0	Feb13	pts/0	01:54:27	consul agent -data-dir=/consul/data -config-dir=/consul/config -data-dir=/consul/data -config-dir=/consul/config -client 0.0.0.0 -server -bootstrap -ui

Port Configuration

Exposed	172.31.47.185:8300 → 8300/tcp
Exposed	172.31.47.185:8301 → 8301/tcp
Internal	8301/udp
Exposed	172.31.47.185:8302 → 8302/tcp
Internal	8302/udp
Exposed	172.31.47.185:8500 → 8500/tcp
Exposed	172.31.47.185:8600 → 8600/tcp
Internal	8600/udp

For the 2nd and 3rd Servers of Consul:

Processes

UID	PID	PPID	C	STIME	TTY	TIME	CMD
root	8845	8800	0	Feb13	?	00:00:00	/usr/bin/dumb-init /bin/sh /usr/local/bin/docker-entrypoint.sh agent -server -join 172.31.47.185
100	8891	8845	0	Feb13	pts/0	02:45:51	consul agent -data-dir=/consul/data -config-dir=/consul/config -server -join 172.31.47.185

For the client of Consul:

UID	PID	PPID	C	STIME	TTY	TIME	CMD
root	9246	9204	0	Feb13	?	00:00:00	/usr/bin/dumb-init /bin/sh /usr/local/bin/docker-entrypoint.sh agent -join 172.31.47.185
100	9297	9246	0	Feb13	pts/0	00:32:30	consul agent -data-dir=/consul/data -config-dir=/consul/config -join 172.31.47.185

For the Fabio Docker:

Processes

UID	PID	PPID	C	STIME	TTY	TIME	CMD
root	21347	21331	0	Feb18	?	00:08:13	/fabio -proxy.addr :8078;proto=tcp

We use the –net-host mode for this docker containter to expose port 9998 and 9999.

The next step, we will register the service to Consul each time we start the container.

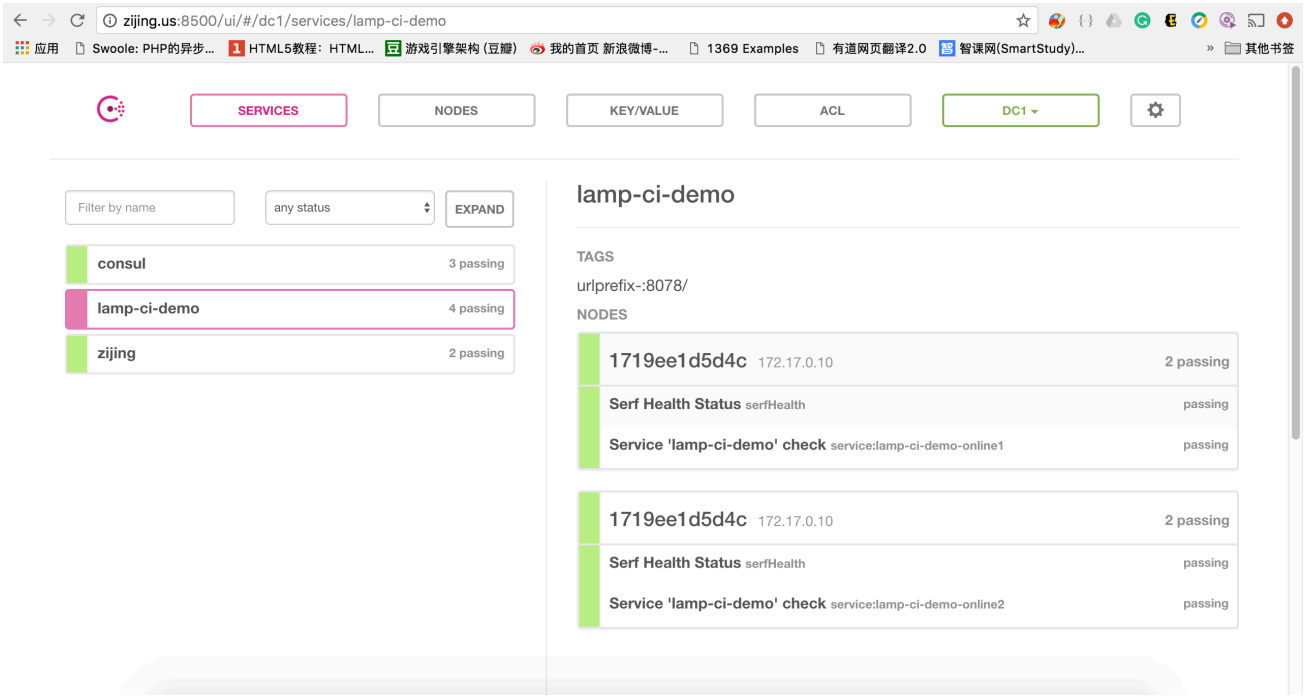
/v1/agent/service/register | http://zijing.us:8500 | http://zijing.us:8500

PUT http://zijing.us:8500/v1/agent/service/register

Authorization | Headers (1) | Body | Pre-request Script | Tests

form-data | x-www-form-urlencoded | raw | binary | JSON (application/json)

```
{
  "ID": "lamp-ci-demo-online1",
  "Name": "lamp-ci-demo",
  "Tags": [
        "urlprefix-zijing.us:6554/lab4.php"
  ],
  "Address": "172.17.0.12",
  "Port": 8000,
  "EnableTagOverride": false,
  "Check": {
    "DeregisterCriticalServiceAfter": "90m",
    "HTTP": "http://www.baidu.com",
    "Interval": "10s"
  }
}
```



Fabio will distribute the load based on the Tags you sent.

To verify that you have successfully deploy the release environment, use command `docker log containerid`.

```
docker.vm:80 172.31.47.185 - - [18/Feb/2018:20:40:38 +0000] "GET / HTTP/1.1" 200 284 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_6) AppleWebKit/537.36 (KHTML, like Gecko) Ch
rome/61.0.3163.100 Safari/537.36"
[httpd:access] zijing.us:80 172.31.47.185 - - [18/Feb/2018:20:40:38 +0000] "GET / HTTP/1.1" 200 bytesIn:1422 bytesOut:284 reqTime:0
127.0.0.1 -  18/Feb/2018:20:40:38 +0000 "GET /index.php" 200 /app/index.php cpu:0.00% mem:2 reqTime:0.000
127.0.0.1 -  18/Feb/2018:20:40:38 +0000 "GET /index.php" 200 /app/index.php cpu:0.00% mem:2 reqTime:0.000
docker.vm:80 172.31.47.185 - - [18/Feb/2018:20:40:38 +0000] "GET / HTTP/1.1" 200 283 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_6) AppleWebKit/537.36 (KHTML, like Gecko) Ch
rome/61.0.3163.100 Safari/537.36"
```

Keyword Explaination:

CD: **Continuous delivery** (**CD**) is a software engineering approach in which teams produce software in short cycles, ensuring that the software can be reliably released at any time.[1] It aims at building, testing, and releasing software faster and more frequently. The approach helps reduce the cost, time, and risk of delivering changes by allowing for more incremental updates to applications in production. A straightforward and repeatable deployment process is important for continuous delivery. Refer to Wikipedia: https://en.wikipedia.org/wiki/Continuous_delivery.

Jenkins: **Jenkins** is an open source automation server written in Java. **Jenkins** helps to automate the non-human part of software development process, with continuous integration and facilitating technical aspects of continuous delivery. It is a server-based system that runs in servlet containers such as Apache Tomcat. It supports version control tools,

including AccuRev, CVS, Subversion, Git, Mercurial, Perforce, ClearCase and RTC, and can execute Apache Ant, Apache Maven and sbt based projects as well as arbitrary shell scriptsand Windows batch commands. The creator of Jenkins is Kohsuke Kawaguchi.[5] Released under the MIT License, Jenkins is free software. Refer to WikiPedia: https://en.wikipedia.org/wiki/Jenkins_(software)

Docker: Docker is an open platform for developers and sysadmins to build, ship, and run distributed applications, whether on laptops, data center VMs, or the cloud.