

Continuous Delivery using Jenkins + Docker + Consul + Fabio

Goal: To deploy a Continuous Integration(CI) & Continuous Delivery(CD) environment to 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>.

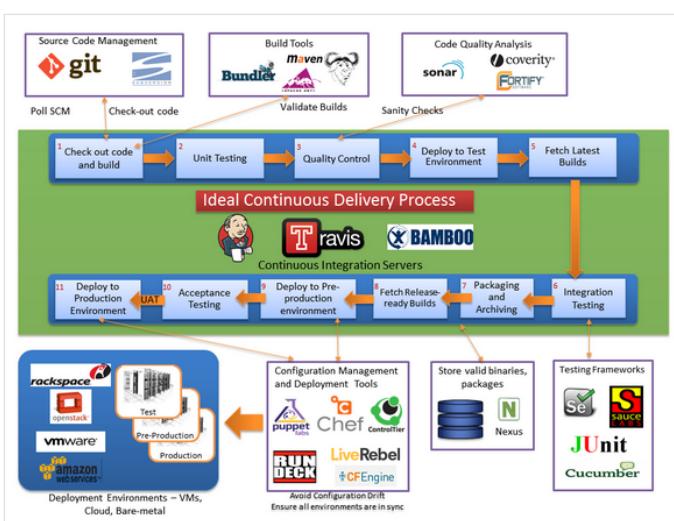
What are the popular choices for CI & CD?

<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.

A docker+jenkins+LAMP deploying demo

13 commits 1 branch 0 releases 1 contributor

Dockerfile README.md composer.json composer.lock index.php indexTest.php

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

The screenshot shows a GitHub repository page for 'zijing2 / DockerPHPCIDemo'. The 'Dockerfile' tab is selected. A red box highlights the Dockerfile code block. The code is as follows:

```
1 From webdevops/php-apache-dev:7.0
2 MAINTAINER Zijing Huang "hzj_huangzijing@hotmail.com"
3 RUN git clone https://github.com/zijing2/DockerPHPCIDemo.git /app
4 WORKDIR /app
5 RUN composer install
6 RUN chmod -R 777 /app
7 EXPOSE 80 3306
```

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.

PUBLIC | AUTOMATED BUILD

zijinghuang/dockerphpcidemo ☆

Last pushed: 9 days ago

Repo Info Tags Dockerfile Build Details Build Settings Collaborators Webhooks Settings

Status	Actions	Tag	Created	Last Updated
✓ Success		latest	9 days ago	9 days ago
✓ Success		latest	9 days ago	9 days ago
✓ Success		latest	9 days ago	9 days ago
⌚ Canceled		latest	9 days ago	9 days ago
❗ Error		latest	9 days ago	9 days ago

Source Repository
zijing2/DockerPHPCIDemo

4. Install and configure Jenkins

Jenkins

Zijing Huang | log out

New Item

ENABLE AUTO REFRESH

All

S	W	Name ↓	Last Success	Last Failure	Last Duration	Fav
●	☀️	demo	19 days - #3	N/A	0.8 sec	
●	☀️	demo2	N/A	N/A	N/A	
●	☀️	helloworld	18 days - #6	N/A	6.8 sec	
●	☀️	LAMP-CI-DEMO	7 days 23 hr - #70	8 days 21 hr - #56	43 sec	

Icon: S M L

Legend: RSS for all RSS for failures RSS for just latest builds

Build Queue

No builds in the queue.

Build Executor Status

master

1 Idle

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.

The screenshot shows the Jenkins interface for creating a new job. The title bar says 'zijing.us:8081/view/all/newJob'. The main content area has a heading 'Enter an item name' with a red error message below it: '» This field cannot be empty, please enter a valid name'. Below this, there are five options: 'Freestyle project', 'Maven project', 'Pipeline' (which is highlighted with a red border), 'External Job', and 'Multi-configuration project'.

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

The screenshot shows the Jenkins 'Advanced Project Options' page for a job named 'LAMP-CI-DEMO'. The 'Pipeline' tab is selected. The 'Definition' dropdown is set to 'Pipeline script'. The script content is a Groovy pipeline script:

```
1 <node('slave') {  
2     stage('dev'){  
3         try{  
4             sh 'docker stop lamp-ci-demo-dev && docker rm lamp-ci-demo-dev'  
5         }catch(exc){  
6             echo "Error: ${exc}"  
7         }  
8     }  
9     docker.image('zijinghuang/dockerphpcidemo:latest').pull()  
10    docker.image('zijinghuang/dockerphpcidemo:latest').run('--name lamp-ci-demo-de  
11    stage('unit-test'){  
12        docker.image('zijinghuang/dockerphpcidemo:latest').pull()  
13        try{  
14            sh 'echo "Unit test successful!"'  
15        }catch(exc){  
16            echo "Unit test failed: ${exc}"  
17        }  
18    }  
19}
```

At the bottom, there are 'Save' and 'Apply' buttons, and a 'Syntax' link.

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

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

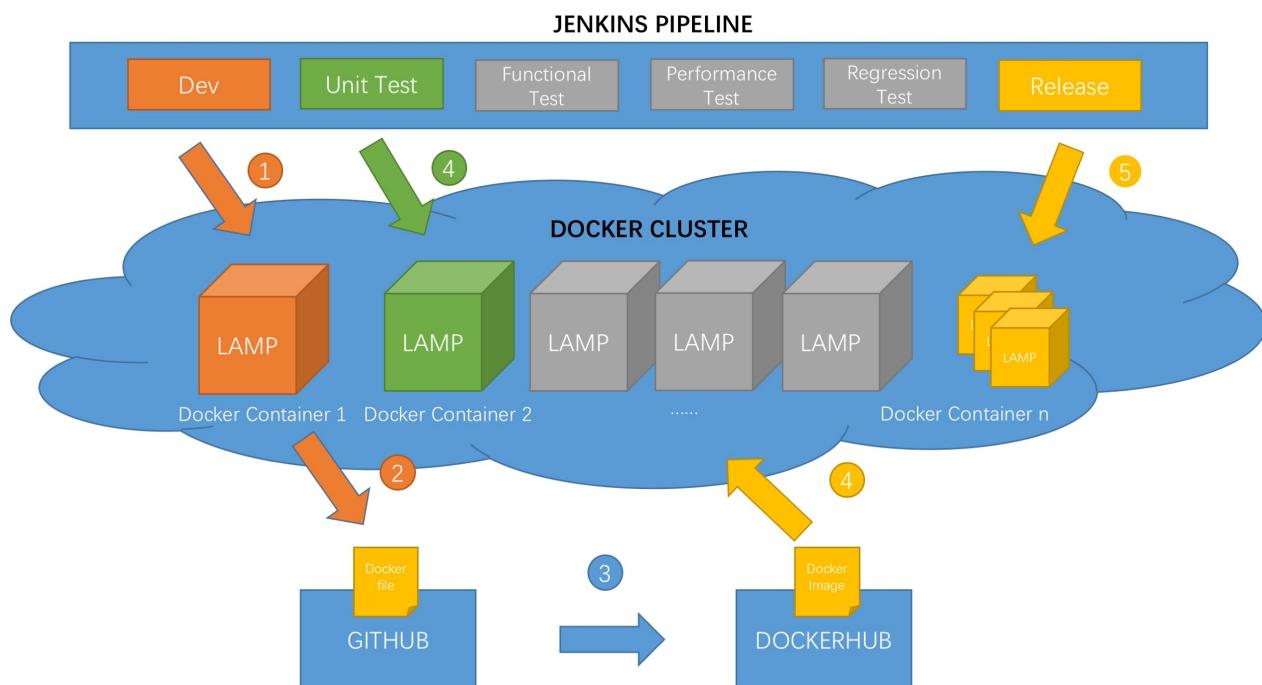
```
[ec2-user@ip-172-31-47-185 ~]$ docker -v
Docker version 17.09.1-ce, build 3dfb8343b139d6342acfd9975d7f1068b5b1c3d3
[ec2-user@ip-172-31-47-185 ~]$
```

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

The screenshot shows the Shipyard application running at zijing.us:8080/#/containers. The interface has a top navigation bar with tabs for CONTAINERS, IMAGES, NODES, REGISTRIES, ACCOUNTS, and EVENTS, and an ADMIN dropdown. Below the navigation is a search bar labeled "Search containers...". A "Refresh" button and a "Deploy Container" button are visible. The main area is a table listing 11 Docker containers:

	Id	Node	Name	Image	Status	Created	Actions
	b20712143118	ip-172-31-47-185	lamp-ci-demo-online2	zijinhuang/dockerphpcidemo:latest	Up About a minute	2018-02-18 15:36:54 -0500	View Edit
	a95333eb61b8	ip-172-31-47-185	lamp-ci-demo-online1	zijinhuang/dockerphpcidemo:latest	Up About a minute	2018-02-18 15:36:45 -0500	View Edit
	b500625c45f4	ip-172-31-47-185	lamp-ci-demo-unittest	zijinhuang/dockerphpcidemo:latest	Up 2 minutes	2018-02-18 15:36:34 -0500	View Edit
	0eeae66d5bac	ip-172-31-47-185	lamp-ci-demo-dev	zijinhuang/dockerphpcidemo:latest	Up 2 minutes	2018-02-18 15:36:26 -0500	View Edit
	81544819fa30	ip-172-31-47-185	fabio	magiconair/fabio	Up 25 minutes	2018-02-18 15:13:10 -0500	View Edit
	a26b9283d086	ip-172-31-47-185	priceless_jang	magiconair/fabio	Exited (1) 5 hours ago	2018-02-18 10:00:40 -0500	View Edit
	1719ee1d5d4c	ip-172-31-47-185	consul-server1	consul	Up 5 days	2018-02-12 22:55:14 -0500	View Edit
	5a1737b6efbf	ip-172-31-47-185	cs546	zijinhuang/cs546	Up 5 days	2018-02-12 21:48:22 -0500	View Edit
	a5f8e51ad65a	ip-172-31-47-185	consul-server3	consul	Up 5 days	2018-02-11 12:56:49 -0500	View Edit
	zijing.us:8080/#/containers	ver2	consul		Up 5 days	2018-02-11 12:55:49 -0500	View Edit

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 trigger the automated build on dockerhub. Eventually, we get the updated image.

zijinghuang/dockerphpcidemo:latest	sha256:d9702	2018-02-17 16:36:31 -0500	728.26 MB	/	Delete
------------------------------------	--------------	---------------------------	-----------	-------------------	------------------------

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.

		Id	Node	Name	Image	Status	Created	Actions
		b20712143118	ip-172-31-47-185	lamp-ci-demo-online2	zijinghuang/dockerphpcidemo:latest	Up 8 days	2018-02-18 15:36:54 -0500	/ Edit
		a9533eb61b8	ip-172-31-47-185	lamp-ci-demo-online1	zijinghuang/dockerphpcidemo:latest	Up 8 days	2018-02-18 15:36:45 -0500	/ Edit
		b500625c45f4	ip-172-31-47-185	lamp-ci-demo-unittest	zijinghuang/dockerphpcidemo:latest	Up 8 days	2018-02-18 15:36:34 -0500	/ Edit
		0eeae66d5bac	ip-172-31-47-185	lamp-ci-demo-dev	zijinghuang/dockerphpcidemo:latest	Up 8 days	2018-02-18 15:36:26 -0500	/ Edit

lamp-ci-demo-dev
zijinghuang/dockerphpcidemo:latest

[Pause](#) [Stop](#) [Restart](#) [Destroy](#) [Stats](#) [Logs](#) [Console](#)

Container Configuration		Swarm Node	
Container ID 0eeae66d5bac	Command supervisord	Name ip-172-31-47-185	Host 172.31.47.185:2375
Hostname 0eeae66d5bac	Domain Name N/A	CPUs 1	Memory 993 MB
Port Configuration		User Defined Container DNS	
Internal 3306/tcp			
Internal 443/tcp			
Exposed 172.31.47.185:8079 → 80/tcp			
Internal 9000/tcp			

lamp-ci-demo-unittest
zijinghuang/dockerphpcidemo:latest

[Pause](#) [Stop](#) [Restart](#) [Destroy](#) [Stats](#) [Logs](#) [Console](#)

Container Configuration		Swarm Node	
Container ID b500625c45f4	Command supervisord	Name ip-172-31-47-185	Host 172.31.47.185:2375
Hostname b500625c45f4	Domain Name N/A	CPUs 1	Memory 993 MB
Port Configuration		User Defined Container DNS	
Internal 3306/tcp			
Internal 443/tcp			
Internal 80/tcp			
Internal 9000/tcp			

Container Configuration

Container ID a95333eb61b8	Command supervisord
Hostname a95333eb61b8	Domain Name N/A

Swarm Node

Name ip-172-31-47-185	Host 172.31.47.185:2375
CPUs 1	Memory 993 MB

Port Configuration

Internal 3306/tcp
Internal 443/tcp
Exposed 172.31.47.185:8077 → 80/tcp
Internal 9000/tcp

User Defined Container DNS

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:

consul:latest	sha256:5f491	2018-02-09 15:08:09 -0500	51.72 MB	</>	
---------------	--------------	---------------------------	----------	-----	--

Fabio Docker Image:

fabiolb/fabio:latest	sha256:129bf	2018-02-08 03:11:50 -0500	7.68 MB	</>	
----------------------	--------------	---------------------------	---------	-----	--

<input type="checkbox"/>		1719ee1d5d4c	ip-172-31-47-185	consul-server1	consul	Up 5 days	2018-02-12 22:55:14 -0500		
<input type="checkbox"/>		5a1737b6efbf	ip-172-31-47-185	cs546	zijinhuang/cs546	Up 5 days	2018-02-12 21:48:22 -0500		
<input type="checkbox"/>		a5f8e51ad65a	ip-172-31-47-185	consul-server3	consul	Up 5 days	2018-02-11 12:56:49 -0500		
<input type="checkbox"/>		2b853feb3ee6	ip-172-31-47-185	consul-server2	consul	Up 5 days	2018-02-11 12:55:49 -0500		
<input type="checkbox"/>		5b398174026e	ip-172-31-47-185	consul-client1	consul	Up 5 days	2018-02-11 10:37:21 -0500		

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 container to expose port 9998 and 9999.

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

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

Authorization Headers (1) Body **JSON (application/json)**

```

1 { "ID": "lamp-ci-demo-online1",
2   "Name": "lamp-ci-demo",
3   "Tags": [
4     "urlprefix-zijing.us:6554/lab4.php"
5   ],
6   "Address": "172.17.0.12",
7   "Port": 8000,
8   "EnableTagOverride": false,
9   "Check": {
10     "DeregisterCriticalServiceAfter": "90m",
11     "HTTP": "http://www.baidu.com",
12     "Interval": "10s"
13   }
14 }
15 }
```

zijing.us:8500/ui/#/dc1/services/lamp-ci-demo

SERVICES NODES KEY/VALUE ACL DC1 ▾

Filter by name any status EXPAND

consul 3 passing

lamp-ci-demo 4 passing

zijing 2 passing

lamp-ci-demo

TAGS urlprefix:-8078/

NODES

1719ee1d5d4c 172.17.0.10	2 passing
Serf Health Status serfHealth	passing
Service 'lamp-ci-demo' check service:lamp-ci-demo-online1	passing

1719ee1d5d4c 172.17.0.10	2 passing
Serf Health Status serfHealth	passing
Service 'lamp-ci-demo' check service:lamp-ci-demo-online2	passing

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

To verify that you have successfully deployed 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) Chrome/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) Chrome/61.0.3163.100 Safari/537.36"
```

Keyword Explanation:

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 scripts and 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_\(software\)](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.