Container Service

Best Practices

MORE THAN JUST CLOUD | Alibaba Cloud

Best Practices

This document provides a Compose file for you to test the container connectivity within a cluster by visiting the access endpoint of the service.

Scenario

When you need to deploy interdependent applications in a Docker cluster, you must ensure that the applications can access one another, namely cross-host container network connectivity is available. However, due to network problems, the containers deployed on different hosts might not be able to access one another. If this happens, it is very difficult to troubleshoot the problem. Therefore, an easy-to-use Compose file that can be used to test the connectivity among cross-host containers within a cluster can really help a lot.

Solution

You can use the image and application template as shown below to test the connectivity among containers.

```
 web:
image: registry.aliyuncs.com/xianlu/test-link
command: python test-link.py
restart: always
ports:
- 5000
links:
- redis
labels:
aliyun.scale: '3'
aliyun.routing.port_5000: test-link;
redis:
image: redis
restart: always
```

This example uses Flask to test the container connectivity.

The above application template deploys a Web service and a Redis service. The Web service contains three Flask containers and the containers will be distributed on three nodes when started. Therefore, if the containers can ping one another, it means that the current network can realize cross-host container access.

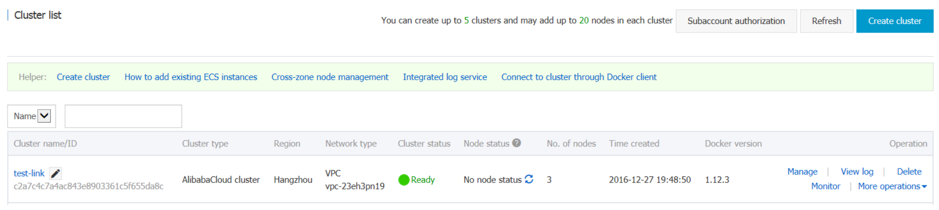
The Redis service runs on one of the three nodes. After the Flask containers start, they register to the Redis service and report their IP addresses; therefore, the Redis service has the IP addresses of all the three Flask containers after the Flask containers start. When you visit any of the three Flask containers, it will send ping commands to the other two and you can check the network connectivity of the cluster according to the ping command response.

Procedure

Create a cluster which contains three nodes.

In this example, the name of the cluster is **test-link**. For more information about how to create a cluster, refer to Create a cluster.

Note: Create a Server Load Balancer instance when you create the cluster.


Cluster list

You can create up to 5 clusters and may add up to 20 nodes in each cluster

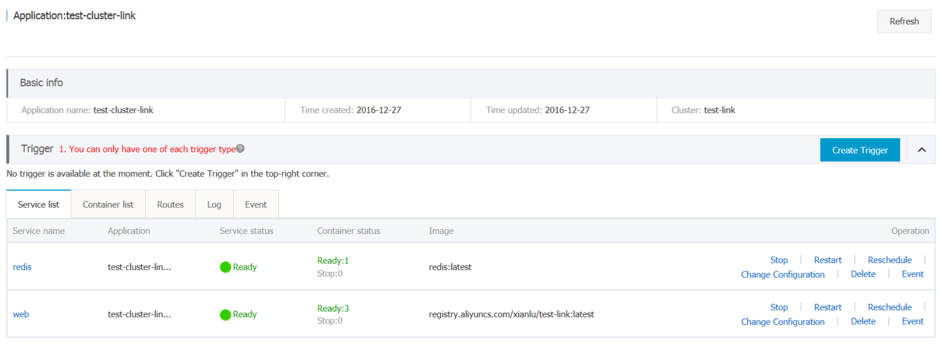
Subaccount authorization
Refresh
Create cluster
Helper: Create cluster | How to add existing ECS instances | Cross-zone node management | Integrated log service | Connect to cluster through Docker client
Name

Cluster name/ID	Cluster type	Region	Network type	Cluster status	Node status	No. of nodes	Time created	Docker version	Operation
test-link c2a7c4c7a4ac843e8903361c5f655da8c	AlibabaCloud cluster	Hangzhou	VPC vpc-23eh3pn19	● Ready	No node status	3	2016-12-27 19:48:50	1.12.3	Manage | View log | Delete Monitor | More operations

Use the above template to create an application (in this example, the name of the application is **test-cluster-link**) to deploy the **web** service and **redis** service.

For more information about how to create an application, refer to Create an application.

In the **Application List** page, click the name of the application to view the services created.


Application:test-cluster-link

Basic info

Application name: test-cluster-link | Time created: 2016-12-27 | Time updated: 2016-12-27 | Cluster: test-link

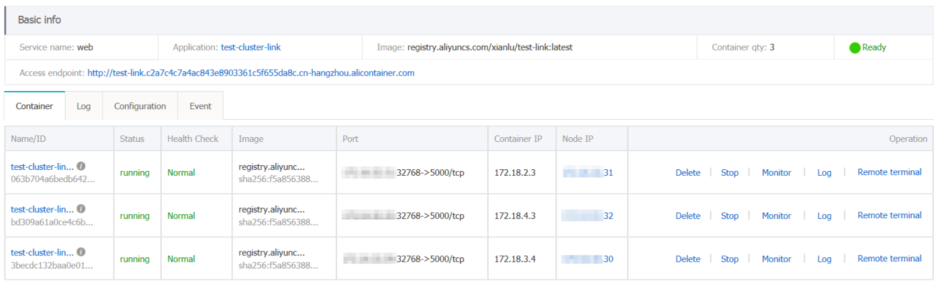
Trigger 1. You can only have one of each trigger type

No trigger is available at the moment. Click "Create Trigger" in the top-right corner.

Service name	Application	Service status	Container status	Image	Operation
redis	test-cluster-lin...	● Ready	Ready:1 Stop:0	redis:latest	Stop | Restart | Reschedule | Change Configuration | Delete | Event
web	test-cluster-lin...	● Ready	Ready:3 Stop:0	registry.aliyuncs.com/xianlu/test-link:latest	Stop | Restart | Reschedule | Change Configuration | Delete | Event

Click the name of the **web** service to enter the service details page.

You can see that the three containers (**test-cluster-link_web_1**, **test-cluster-link_web_2**, **test-cluster-link_web_3** ） all start and are distributed on different nodes.


Basic info

Service name: web | Application: test-cluster-link | Image: registry.aliyuncs.com/xianlu/test-link:latest | Container qty: 3 | ● Ready

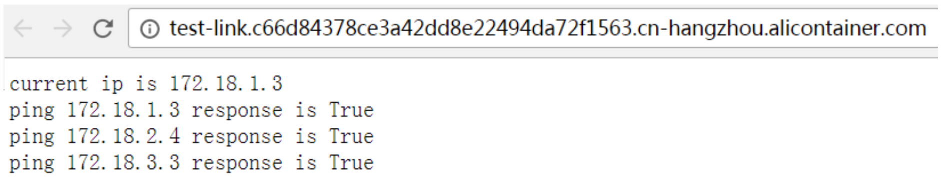
Access endpoint: http://test-link.c2a7c4c7a4ac843e8903361c5f655da8c.cn-hangzhou.alicontainer.com

Container | Log | Configuration | Event

Name/ID	Status	Health Check	Image	Port	Container IP	Node IP	Operation
test-cluster-lin... 063b704a6bedb642...	running	Normal	registry.aliyunc... sha256:f5a856388...	[redacted]32768->5000/tcp	172.18.2.3	[redacted]31	Delete | Stop | Monitor | Log | Remote terminal
test-cluster-lin... bd309a61a0ce4c6b...	running	Normal	registry.aliyunc... sha256:f5a856388...	[redacted]32768->5000/tcp	172.18.4.3	[redacted]32	Delete | Stop | Monitor | Log | Remote terminal
test-cluster-lin... 3becdc132baa0e01...	running	Normal	registry.aliyunc... sha256:f5a856388...	[redacted]32768->5000/tcp	172.18.3.4	[redacted]30	Delete | Stop | Monitor | Log | Remote terminal

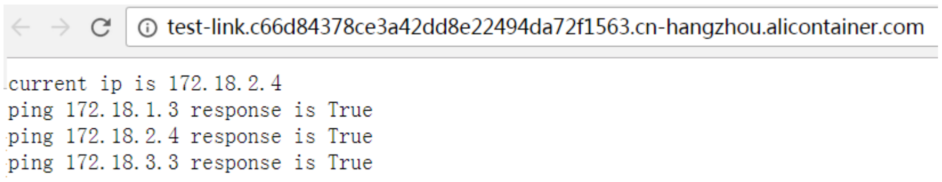
Visit the access endpoint of the **web** service.

As shown in the figure below, the container **test-cluster-link_web_1** can access the container **test-cluster-link_web_2** and container **test-cluster-link_web_3**.


test-link.c66d84378ce3a42dd8e22494da72f1563.cn-hangzhou.alicontainer.com

```
current ip is 172.18.1.3
ping 172.18.1.3 response is True
ping 172.18.2.4 response is True
ping 172.18.3.3 response is True
```

Refresh the webage. As shown in the figure below, the container **test-cluster-link_web_2** can visit the container **test-cluster-link_web_1** and container **test-cluster-link_web_3**.


test-link.c66d84378ce3a42dd8e22494da72f1563.cn-hangzhou.alicontainer.com

```
current ip is 172.18.2.4
ping 172.18.1.3 response is True
ping 172.18.2.4 response is True
ping 172.18.3.3 response is True
```

As the above results show, the containers in the cluster can access one another.

Use OSSFS data volumes to share WordPress attachments

This document describes how to share WordPress attachments among different containers by creating OSSFS data volumes on Alibaba Cloud Container Service.

Scenario

Docker containers simplify WordPress deployment. With Alibaba Cloud Container Service, you can use an application template for one-click deployment of WordPress.

Note: For details about how to use Alibaba Cloud Container Service to create a WordPress application, refer to Create WordPress by using an application template.

In this example, the following application template is used to create an application named **wordpress**.

```
web:
image: registry.aliyuncs.com/acs-sample/wordpress:4.3
ports:
- '80'
environment:
WORDPRESS_AUTH_KEY: changeme
WORDPRESS_SECURE_AUTH_KEY: changeme
WORDPRESS_LOGGED_IN_KEY: changeme
WORDPRESS_NONCE_KEY: changeme
WORDPRESS_AUTH_SALT: changeme
WORDPRESS_SECURE_AUTH_SALT: changeme
WORDPRESS_LOGGED_IN_SALT: changeme
WORDPRESS_NONCE_SALT: changeme
WORDPRESS_NONCE_AA: changeme
restart: always
links:
- 'db:mysql'
labels:
aliyun.logs: /var/log
aliyun.probe.url: http://container/license.txt
aliyun.probe.initial_delay_seconds: '10'
aliyun.routing.port_80: http://wordpress
aliyun.scale: '3'
db:
image: registry.aliyuncs.com/acs-sample/mysql:5.7
environment:
MYSQL_ROOT_PASSWORD: password
restart: always
labels:
aliyun.logs: /var/log/mysql
```

This application consists of a MySQL container and three WordPress containers (aliyun.scale: '3' is the extension label of Alibaba Cloud Container Service, and specifies the number of containers. For details about the labels supported by Alibaba Cloud Container Service, refer to Label description). The WordPress containers access MySQL through a link. The aliyun.routing.port_80: http://wordpress label defines the load balancing among the three WordPress containers (for details, refer to Exposing HTTP service through acsrouting).

In this example, the application can be deployed with complete features. However, the attachments uploaded by WordPress are stored in the local disk, which means they cannot be shared across different containers or opened once requests are routed to other containers.

Solution

This document describes how to use OSSFS data volumes on Alibaba Cloud Container Service to share WordPress attachments across different containers without any code modifications.

OSSFS data volume is a third-party data volume provided by Alibaba Cloud Container Service to package various cloud storages (for example, OSS) into data volumes and to directly mount these data volumes to the containers. This means the data volumes can be shared across different containers and automatically re-mounted upon container restart and migration.

Operating procedure

Step 1: Create OSSFS data volumes

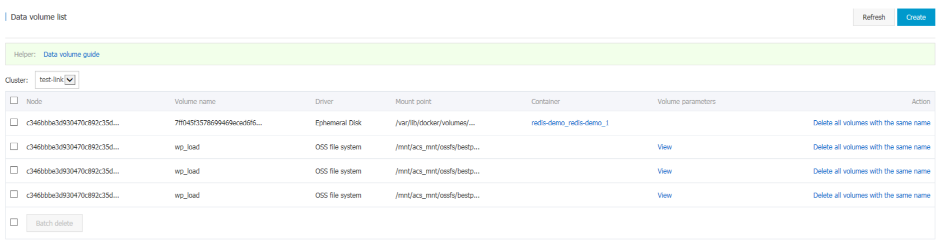
Log on to the Container Service console.

Click **Data Volumes** in the left navigation pane.

Select the desired cluster and click **Create** in the upper-right corner.

For details about how to create OSSFS data volumes, refer to Create an OSSFS data volume.

Here the created OSSFS data volumes are named **wp_upload**. The Container Service uses the same name to create data volumes on all nodes of a cluster.



Node	Volume name	Driver	Mount point	Container	Volume parameters	Action
c346bbbe3d930470c892c35d...	7ff045f3578699469eced6f6...	Ephemeral Disk	/var/lib/docker/volumes/...	redis-demo_redis-demo_1		Delete all volumes with the same name
c346bbbe3d930470c892c35d...	wp_load	OSS file system	/mnt/acs_mnt/ossfs/bestp...		View	Delete all volumes with the same name
c346bbbe3d930470c892c35d...	wp_load	OSS file system	/mnt/acs_mnt/ossfs/bestp...		View	Delete all volumes with the same name
c346bbbe3d930470c892c35d...	wp_load	OSS file system	/mnt/acs_mnt/ossfs/bestp...		View	Delete all volumes with the same name

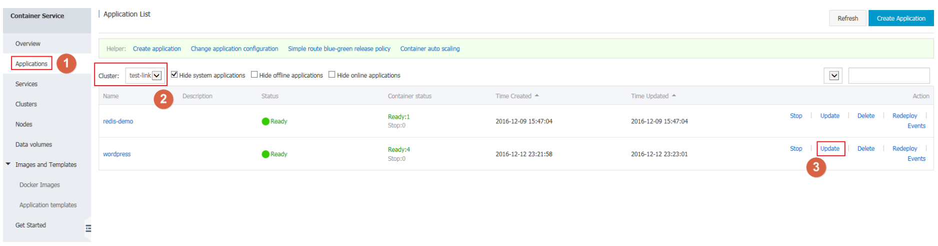
Step 2: Use the OSSFS data volumes

The WordPress attachments are stored in the /var/www/html/wp-content/uploads directory by default. In this example, we only need to map OSSFS data volumes to this directory for sharing of an OSS bucket across different WordPress containers.

Log on to the Container Service console.

Click **Applications** in the left navigation pane.

Select the target cluster as well as the created application **wordpress** and click **Update** at the right side.



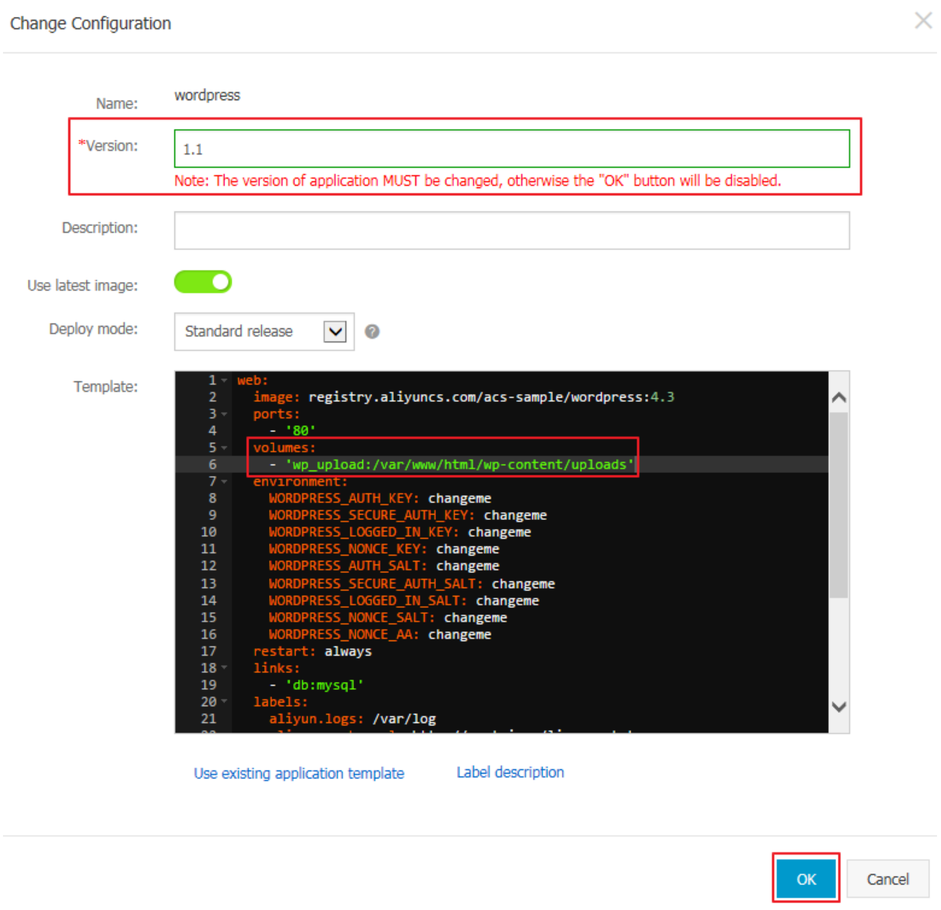
Cluster: test-link

☑ Hide system applications ☐ Hide offline applications ☐ Hide online applications

Name	Description	Status	Container status	Time Created	Time Updated	Action
redis-demo		● Ready	Ready:1 Stop:0	2016-12-09 15:47:04	2016-12-09 15:47:04	Stop | Update | Delete | Redeploy | Events
wordpress		● Ready	Ready:4 Stop:0	2016-12-12 23:21:58	2016-12-12 23:23:01	Stop | Update | Delete | Redeploy | Events

In **Template**, add the mapping of OSSFS data volumes to the WordPress directory.

Note: You must modify the **Version**; otherwise, the application cannot be re-deployed.



Name: wordpress

*Version: 1.1

Note: The version of application MUST be changed, otherwise the "OK" button will be disabled.

Description:

Use latest image: ●

Deploy mode: Standard release

Template:

```
web:
  image: registry.aliyuncs.com/acs-sample/wordpress:4.3
  ports:
    - '80'
  volumes:
    - 'wp_upload:/var/www/html/wp-content/uploads'
  environment:
    WORDPRESS_AUTH_KEY: changeme
    WORDPRESS_SECURE_AUTH_KEY: changeme
    WORDPRESS_LOGGED_IN_KEY: changeme
    WORDPRESS_NONCE_KEY: changeme
    WORDPRESS_AUTH_SALT: changeme
    WORDPRESS_SECURE_AUTH_SALT: changeme
    WORDPRESS_LOGGED_IN_SALT: changeme
    WORDPRESS_NONCE_SALT: changeme
    WORDPRESS_NONCE_AA: changeme
  restart: always
  links:
    - 'db:mysql'
  labels:
    aliyun.logs: /var/log
```

Click **OK** to re-deploy the application.

Open WordPress and upload attachments. Then you can see the uploaded attachments in the OSS bucket.

Log

Use ELK in Container Service

This document introduces how to use ELK in the Container Service.

Background

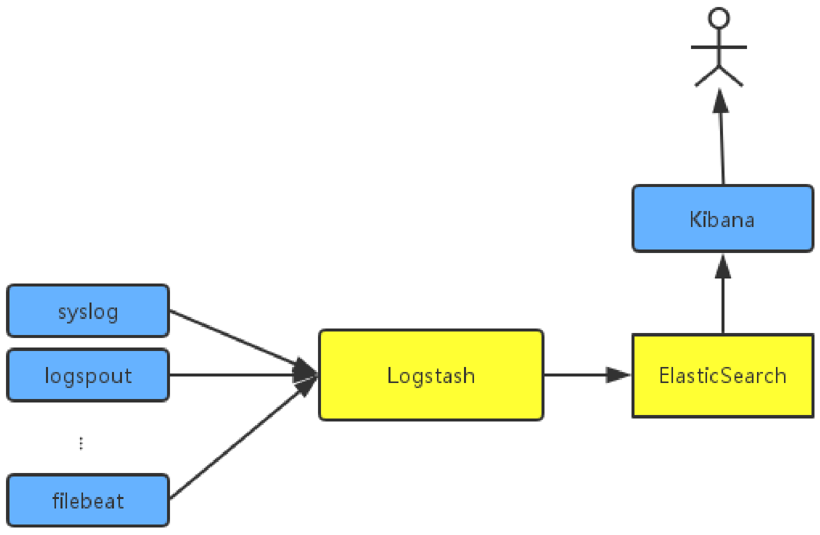
Logs are an important component of the IT system. They records system events and the time when the events occur. We can troubleshoot system faults according to the logs and make statistical analysis.

Logs are usually stored in the local log files. You need to log on to the server and filter keywords using grep or other tools to view the logs. But when the application will be deployed on multiple servers, this log viewing method is very inconvenient. To locate the logs for a specific error, you' d have to log on to all the servers and filter keywords one file after another. That is why concentrated log storage came about: all the logs are collected in the log service and you can view and search for a log in the log service.

In the Docker environment, concentrated log storage is even more important. Compared with the traditional O&M model, Docker usually uses the orchestration system to manage containers. The mapping between the containers and hosts is not fixed and containers may also be constantly migrated between hosts. You cannot view the logs by logging on to the server and the concentrated log becomes the only choice.

The Container Service integrates Alibaba Cloud Log Service and automatically collects container logs to the Log Service through declarations. But some users may prefer the ELK (Elasticsearch + Logstash + Kibana) combination. This document introduces how to use ELK in the Container Service.

Overall structure



In this example, an independent Logstash cluster needs to be deployed. Logstash is large and resource-consuming, so it won’ t run on every server, not to mention in every Docker container. To collect the container logs, Syslog, Logspout and Filebeat are used. Of course, you may also use other collection methods.

To fit the actual scenario as much as possible, two clusters are created here: one is the **testelk** cluster for deploying ELK, and the other is the **app** cluster for deploying applications.

Operation steps

Note: The clusters and Server Load Balancers created in this document are all located in the same region.

Step 1. Create a Server Load Balancer instance

To enable other services to send logs to Logstash, a Server Load Balancer is configured before Logstash.

Before creating an application, first create a Server Load Balancer instance of the **public network** type in the Server Load Balancer Management Console and set the Server Load Balancer to listen to Port 5000 and Port 5044, with no backend services added.

Step 2. Deploy ELK

Log on to the Container Service Management Console and create a cluster named **testelk**.

For how to create a cluster, see Create a cluster.

Note: The cluster must be in the same region with the Server Load Balancer instance created above.

Bind the Server Load Balancer instance created above to the cluster.

On the Cluster List page, select the cluster **testelk** > **Manage** on its right > **Load Balancer Settings** in the left navigation bar > **Bind Server Load Balancer** > Select the Server Load Balancer instance created above and click **OK**.

Deploy ELK using the orchestration template below. In this example, an application named **elk** is created.

For how to create an application using orchestration templates, see Create an application.

Note: You should replace ${SLB_ID} in the orchestration file with the ID of the Server Load Balancer instance created above.

```
version: '2'
services:
elasticsearch:
image: elasticsearch

kibana:
image: kibana
environment:
ELASTICSEARCH_URL: http://elasticsearch:9200/
labels:
aliyun.routing.port_5601: kibana
links:
- elasticsearch

logstash:
image: registry.cn-hangzhou.aliyuncs.com/acs-sample/logstash
hostname: logstash
ports:
- 5044:5044
- 5000:5000
labels:
aliyun.lb.port_5044: 'tcp://${SLB_ID}:5044' #First create a Server Load Balancer instance
aliyun.lb.port_5000: 'tcp://${SLB_ID}:5000'
links:
- elasticsearch
```

In this orchestration file, the official images are used for Elasticsearch and Kibana, with no changes made. Logstash needs a configuration file, so you have to make an image on your own to store the configuration file. The image source code can be found here.

The Logstash configuration file is as follows. This is a very simple Logstash configuration. Two input formats, namely syslog and filebeats, are provided and their external ports are

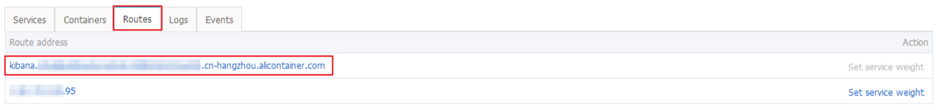
5044 and 5000 respectively.

```
input {
    beats {
        port => 5044
        type => beats
    }

    tcp {
        port => 5000
        type => syslog
    }

}

filter {
}

output {
    elasticsearch {
        hosts => ["elasticsearch:9200"]
    }

    stdout { codec => rubydebug }
}
```

Configure the Kibana index.

Access Kibana.

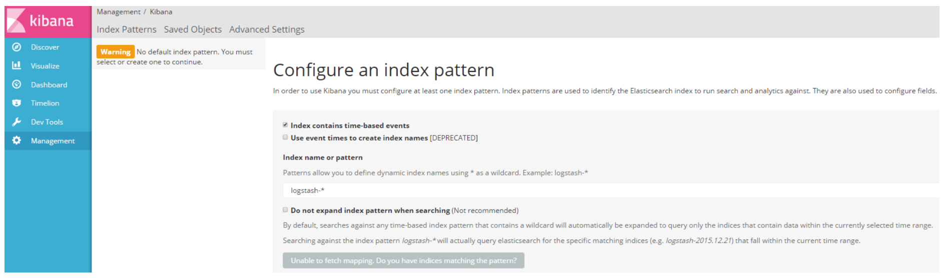
The URL can be found in the application route list (click the name of the application created **elk** > the **Routes** tab > Route address).



Route address	Action
kibana.[redacted].cn-hangzhou.alicontainer.com	Set service weight
[redacted].95	Set service weight

Create an index.

Configure based on your actual needs and click **Create**.



Warning No default index pattern. You must select or create one to continue.

Configure an index pattern

In order to use Kibana you must configure at least one index pattern. Index patterns are used to identify the Elasticsearch index to run search and analytics against. They are also used to configure fields.

[✓] **Index contains time-based events**
[ ] **Use event times to create index names** [DEPRECATED]

Index name or pattern

Patterns allow you to define dynamic index names using * as a wildcard. Example: logstash-*

logstash-*

[ ] **Do not expand index pattern when searching** (Not recommended)

By default, searches against any time-based index pattern that contains a wildcard will automatically be expanded to query only the indices that contain data within the currently selected time range.

Searching against the index pattern *logstash-** will actually query elasticsearch for the specific matching indices (e.g. *logstash-2015.12.21*) that fall within the current time range.

Unable to fetch mapping. Do you have indices matching the pattern?

Step 3. Collect logs

In Docker, the standard logs adopt Stdout file pointer. The following example first demonstrates how to collect Stdout to ELK. If you are using file logs, you can also use Filebeat directly. WordPress is used for the demonstration. The following is an orchestration template of WordPress. An application **wordpress** is created in another cluster.

Log on to the Container Service Management Console and create a cluster named **app**.

For how to create a cluster, see Create a cluster.

Note: The cluster must be in the same region with the Server Load Balancer instance created above.

Create the application **wordpress** using the orchestration template below.

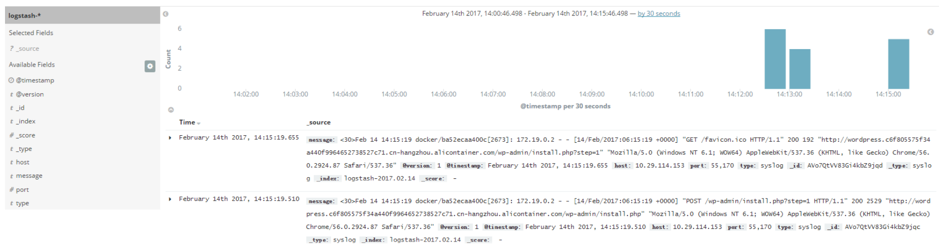
Note: You should replace ${SLB_IP} in the orchestration file with the IP address of the Server Load Balancer instance created above.

```
version: '2'
services:
mysql:
image: mysql
environment:
- MYSQL_ROOT_PASSWORD=password

wordpress:
image: wordpress
labels:
aliyun.routing.port_80: wordpress
links:
- mysql:mysql
environment:
- WORDPRESS_DB_PASSWORD=password
logging:
driver: syslog
options:
syslog-address: 'tcp://${SLB_IP}:5000'
```

After the application is deployed successfully, find the access address of **wordpress** (click the name of the **wordpress** application created > the **Routes** tab > Route address) and you will be able to access the **wordpress** application.

Visit the Kibana page to view the collected logs.

On the application list page, click the application name **elk** > the **Routes** tab > Route address.



```
February 14th 2017, 14:15:19.655   message: <30>Feb 14 14:15:19 docker/ba52ecaa400c[2673]: 172.19.0.2 - - [14/Feb/2017:06:15:19 +0000] "GET /favicon.ico HTTP/1.1" 200 192 "http://wordpress.c6f805575f34a440f9964652738527c71.cn-hangzhou.alicontainer.com/wp-admin/install.php?step=1" "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/56.0.2924.87 Safari/537.36" @version: 1 @timestamp: February 14th 2017, 14:15:19.655 host: 10.29.114.153 port: 55,170 type: syslog _id: AVo7QtVV83Gi4kbZ9jqd _type: syslog _index: logstash-2017.02.14 _score: -

February 14th 2017, 14:15:19.510   message: <30>Feb 14 14:15:19 docker/ba52ecaa400c[2673]: 172.19.0.2 - - [14/Feb/2017:06:15:19 +0000] "POST /wp-admin/install.php?step=1 HTTP/1.1" 200 2529 "http://wordpress.c6f805575f34a440f9964652738527c71.cn-hangzhou.alicontainer.com/wp-admin/install.php" "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/56.0.2924.87 Safari/537.36" @version: 1 @timestamp: February 14th 2017, 14:15:19.510 host: 10.29.114.153 port: 55,170 type: syslog _id: AVo7QtVV83Gi4kbZ9jqc _type: syslog _index: logstash-2017.02.14 _score: -
```

A new Docker log collection scheme: fluentd-pilot

This document introduces a new log collector for Docker: Fluentd-pilot. Fluentd-pilot is a log collecting image we provide for you. You can deploy a Fluentd-pilot instance on each machine to collect logs of all the Docker applications.

Fluentd-pilot has the following features:

- A separate fluentd process to collect logs of all the containers on the machine. There is no need to start a fluentd process for each container.
- It supports file logs and stdout logs. Docker log drivers or Logspout can only process stdout, while fluentd-pilot not only supports collection of stdout logs, but also supports collection of file logs.
- Declarative configuration. When your container has logs to collect, the fluentd-pilot will automatically collect logs of the new container as long as the path of the log file to be collected is declared through the label, requiring no changes to any other configuration.
- It supports multiple log storage methods. Whether it is a powerful Alibaba Cloud Log Service, or the more popular ElasticSearch combination, or even Graylog, Fluentd-pilot can deliver the log to the correct location.
- Open-source. Fluentd-pilot is fully open-source. You can download its code here. If the current features cannot meet your requirements, welcome to raise an issue.

Quick boot

Next I will show a simple scenario: first start Fluentd-pilot, then start a Tomcat container, and let Fluentd-pilot collect Tomcat logs. For the sake of simplicity, here Alibaba Cloud Log Service or ELK is not involved. If you want to run it locally, you just need a machine that runs Docker.

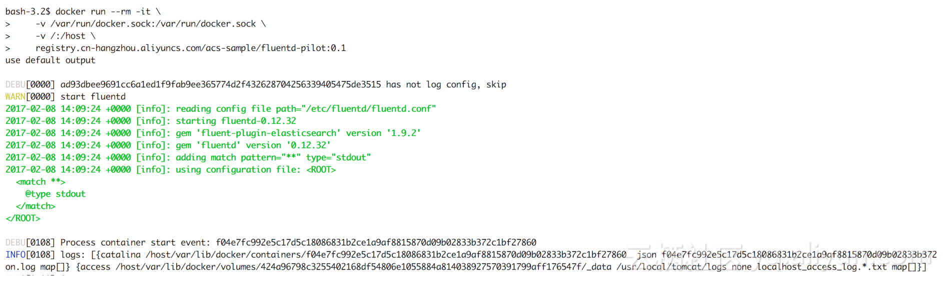
First, start Fluentd-pilot.

> **Note:** When Fluentd-pilot is started in this approach, because there is no log storage configured for backend use, all the collected logs will be directly output to the console.

Open the terminal and input the following commands:

```
docker run --rm -it \
-v /var/run/docker.sock:/var/run/docker.sock \
-v /:/host \
registry.cn-hangzhou.aliyuncs.com/acs-sample/fluentd-pilot:0.1
```

You will see the startup logs of Fluentd-pilot.



```
bash-3.2$ docker run --rm -it \
>     -v /var/run/docker.sock:/var/run/docker.sock \
>     -v /:/host \
>     registry.cn-hangzhou.aliyuncs.com/acs-sample/fluentd-pilot:0.1
use default output

DEBU[0000] ad93dbee9691cc6a1ed1f9fab9ee365774d2f43262870425633940547 5de3515 has not log config, skip
WARN[0000] start fluentd
2017-02-08 14:09:24 +0000 [info]: reading config file path="/etc/fluentd/fluentd.conf"
2017-02-08 14:09:24 +0000 [info]: starting fluentd-0.12.32
2017-02-08 14:09:24 +0000 [info]: gem 'fluent-plugin-elasticsearch' version '1.9.2'
2017-02-08 14:09:24 +0000 [info]: gem 'fluentd' version '0.12.32'
2017-02-08 14:09:24 +0000 [info]: adding match pattern="**" type="stdout"
2017-02-08 14:09:24 +0000 [info]: using configuration file: <ROOT>
  <match **>
    @type stdout
  </match>
</ROOT>

DEBU[0108] Process container start event: f04e7fc992e5c17d5c18086831b2ce1a9af8815870d09b02833b372c1bf27860
INFO[0108] logs: [{catalina /host/var/lib/docker/containers/f04e7fc992e5c17d5c18086831b2ce1a9af8815870d09b02833b372c1bf27860  json f04e7fc992e5c17d5c18086831b2ce1a9af8815870d09b02833b372
on.log map[]} {access /host/var/lib/docker/volumes/424a96798c3255402168df54806e1055884a8140389275703917 99aff176547f/_data /usr/local/tomcat/logs none localhost_access_log.*.txt map[]}]
```

Do not close the terminal. Open a new terminal to start Tomcat. The Tomcat image is among the few Docker images that use stdout and file logs at the same time, and it is very suitable for the demonstration here.

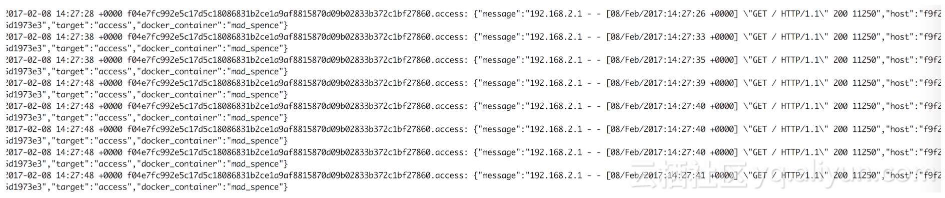
```
docker run -it --rm  -p 10080:8080 \
-v /usr/local/tomcat/logs \
--label aliyun.logs.catalina=stdout \
--label aliyun.logs.access=/usr/local/tomcat/logs/localhost_access_log.*.txt \
tomcat
```

Note:

- aliyun.logs.catalina=stdout tells Fluentd-pilot that this container wants to collect stdout logs.
- aliyun.logs.access=/usr/local/tomcat/logs/localhost_access_log.*.txt indicates to collect all log files whose names comply with the localhost_access_log.*.txt format under the /usr/local/tomcat/logs/ directory in the container. The label usage will be introduced in detail later.

> **Note:** If you deploy Tomcat locally, instead of on the Alibaba Cloud container service, you should specify -v /usr/local/tomcat/logs. Otherwise, Fluentd-pilot may be unable to read log files. The Container Service has implemented the optimization and you don’ t need to specify -v on your own.

Fluentd-pilot will monitor the events in the Docker container. When it finds any container with aliyun.logs.xxx, it will automatically parse the container configuration and starts to collect the corresponding logs. After you start Tomcat, you will find a pile of contents is output immediately by the Fluentd-pilot terminal, including the stdout logs output at the Tomcat startup, and some debugging information output by Fluentd-pilot itself.



```
017-02-08 14:27:28 +0000 f04e7fc992e5c17d5c18086831b2ce1a9af8815870d09b02833b372c1bf27860.access: {"message":"192.168.2.1 - - [08/Feb/2017:14:27:26 +0000] \"GET / HTTP/1.1\" 200 11250","host":"f9f2
d1973e3","target":"access","docker_container":"mad_spence"}
017-02-08 14:27:38 +0000 f04e7fc992e5c17d5c18086831b2ce1a9af8815870d09b02833b372c1bf27860.access: {"message":"192.168.2.1 - - [08/Feb/2017:14:27:33 +0000] \"GET / HTTP/1.1\" 200 11250","host":"f9f2
d1973e3","target":"access","docker_container":"mad_spence"}
017-02-08 14:27:38 +0000 f04e7fc992e5c17d5c18086831b2ce1a9af8815870d09b02833b372c1bf27860.access: {"message":"192.168.2.1 - - [08/Feb/2017:14:27:35 +0000] \"GET / HTTP/1.1\" 200 11250","host":"f9f2
d1973e3","target":"access","docker_container":"mad_spence"}
017-02-08 14:27:48 +0000 f04e7fc992e5c17d5c18086831b2ce1a9af8815870d09b02833b372c1bf27860.access: {"message":"192.168.2.1 - - [08/Feb/2017:14:27:39 +0000] \"GET / HTTP/1.1\" 200 11250","host":"f9f2
d1973e3","target":"access","docker_container":"mad_spence"}
017-02-08 14:27:48 +0000 f04e7fc992e5c17d5c18086831b2ce1a9af8815870d09b02833b372c1bf27860.access: {"message":"192.168.2.1 - - [08/Feb/2017:14:27:40 +0000] \"GET / HTTP/1.1\" 200 11250","host":"f9f2
d1973e3","target":"access","docker_container":"mad_spence"}
017-02-08 14:27:48 +0000 f04e7fc992e5c17d5c18086831b2ce1a9af8815870d09b02833b372c1bf27860.access: {"message":"192.168.2.1 - - [08/Feb/2017:14:27:40 +0000] \"GET / HTTP/1.1\" 200 11250","host":"f9f2
d1973e3","target":"access","docker_container":"mad_spence"}
017-02-08 14:27:48 +0000 f04e7fc992e5c17d5c18086831b2ce1a9af8815870d09b02833b372c1bf27860.access: {"message":"192.168.2.1 - - [08/Feb/2017:14:27:40 +0000] \"GET / HTTP/1.1\" 200 11250","host":"f9f2
d1973e3","target":"access","docker_container":"mad_spence"}
017-02-08 14:27:48 +0000 f04e7fc992e5c17d5c18086831b2ce1a9af8815870d09b02833b372c1bf27860.access: {"message":"192.168.2.1 - - [08/Feb/2017:14:27:41 +0000] \"GET / HTTP/1.1\" 200 11250","host":"f9f2
d1973e3","target":"access","docker_container":"mad_spence"}
```

云栖社区 yq.aliyun.com

You can access the just-deployed Tomcat in the browser, and you will find that similar records can be found on the Fluentd-pilot terminal every time you refresh the browser. In specific, the content after message is the logs collected from /usr/local/tomcat/logs/localhost_access_log.XXX.txt.

Use ElasticSearch + Kibana

First you should deploy ElastichSearch + Kibana. You can refer to Use ELK in the Container Service to deploy ELK in the Alibaba Cloud Container Service, or deploy them directly on your machine following the ElasticSearch/Kibana documents. This document assumes that you have deployed the two components.

If you are still running the Fluentd-pilot, close it first, and then start it again using the command below.

Note: Before executing the following commands, first replace the two variables of ELASTICSEARCH_HOST and ELASTICSEARCH_PORT with the actual values you are using. ELASTICSEARCH_PORT is usually 9200.

```
docker run --rm -it \
-v /var/run/docker.sock:/var/run/docker.sock \
-v /:/host \
-e FLUENTD_OUTPUT=elasticsearch \
-e ELASTICSEARCH_HOST=${ELASTICSEARCH_HOST} \
-e ELASTICSEARCH_PORT=${ELASTICSEARCH_PORT}
registry.cn-hangzhou.aliyuncs.com/acs-sample/fluentd-pilot:0.1
```

Compared with the previous Fluentd-pilot startup method, here three environmental variables are added:

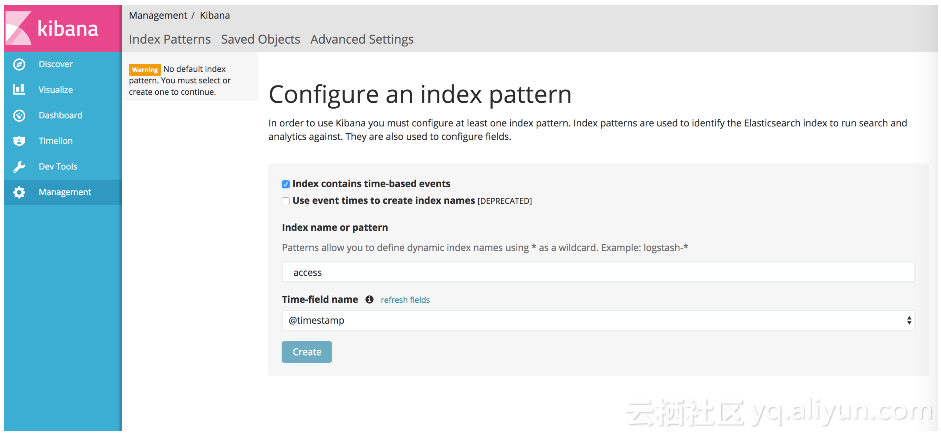
```
- FLUENTD_OUTPUT=elasticsearch: Send the logs to ElasticSearch.
- ELASTICSEARCH_HOST=${ELASTICSEARCH_HOST}: The domain name of ElasticSearch.
- ELASTICSEARCH_PORT=${ELASTICSEARCH_PORT}: The port number of ElasticSearch.
```

Continue to run Tomcat started previously, and access it again to make Tomcat generate some logs. All these newly generated logs will be sent to ElasticSearch.

Open Kibana, and no new logs are visible yet. You need to create an index first. Fluentd-pilot will write logs to the specific index of ElasticSearch. The rules are as follows:

If the tag aliyun.logs.tags is used in the application, and the tags contains target, you should make the target as the index in ElasticSearch. Otherwise, you should make the XXX in the tag aliyun.logs.XXX as the index.

In the previous example about Tomcat, the tag aliyun.logs.tags is not used, so access and catalina are used by default as the index. First create the index access.



No default index pattern. You must select or create one to continue.

Configure an index pattern

In order to use Kibana you must configure at least one index pattern. Index patterns are used to identify the Elasticsearch index to run search and analytics against. They are also used to configure fields.

Index contains time-based events

Use event times to create index names [DEPRECATED]

Index name or pattern

Patterns allow you to define dynamic index names using * as a wildcard. Example: logstash-*

access

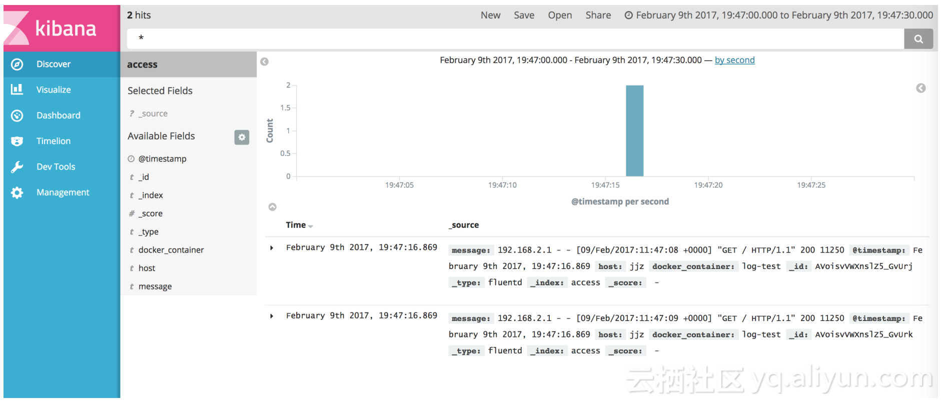
Time-field name refresh fields

@timestamp

Create

云栖社区 yq.aliyun.com

After the index is ready, you can view the logs.



2 hits

February 9th 2017, 19:47:00.000 to February 9th 2017, 19:47:30.000

access

? _source

@timestamp, _id, _index, _score, _type, docker_container, host, message

February 9th 2017, 19:47:00.000 - February 9th 2017, 19:47:30.000 — by second

```
Time       _source
February 9th 2017, 19:47:16.869   message: 192.168.2.1 - - [09/Feb/2017:11:47:08 +0000] "GET / HTTP/1.1" 200 11250 @timestamp: Fe
                                  bruary 9th 2017, 19:47:16.869 host: jjz docker_container: log-test _id: AVoisvVWXnslZ5_GvUrj
                                  _type: fluentd _index: access _score: -
February 9th 2017, 19:47:16.869   message: 192.168.2.1 - - [09/Feb/2017:11:47:09 +0000] "GET / HTTP/1.1" 200 11250 @timestamp: Fe
                                  bruary 9th 2017, 19:47:16.869 host: jjz docker_container: log-test _id: AVoisvVWXnslZ5_GvUrk
                                  _type: fluentd _index: access _score: -
```

云栖社区 yq.aliyun.com

Use Fluentd-pilot in the Alibaba Cloud Container Service

The Container Service makes some special optimization for Fluentd-pilot to best adapt to running Fluentd-pilot.

To run Fluentd-pilot in the Container Service, you only need to create a new application using the following orchestration file. For how to create an application, see Create an application.

```
pilot:
image: registry.cn-hangzhou.aliyuncs.com/acs-sample/fluentd-pilot:0.1
volumes:
- /var/run/docker.sock:/var/run/docker.sock
- /:/host
environment:
FLUENTD_OUTPUT: ElasticSearch # can be replaced based on your requirements
ELASTICSEARCH_HOST: ${elasticsearch} # can be replaced based on your requirements
ELASTICSEARCH_PORT: 9200
labels:
aliyun.global: true
```

Next you can use the aliyun.logs.xxx tag on the application you want to collect logs for.

Label description

When Tomcat is started, the following two labels are declared to tell Fluentd-pilot the location of the container logs.

```
--label aliyun.logs.catalina=stdout
--label aliyun.logs.access=/usr/local/tomcat/logs/localhost_access_log.*.txt
```

You can also add more labels on the application container.

aliyun.logs.$name = $path

- The variable name is the log name and can only contain 0~9, a~z, A~Z and hyphen (-).
- The variable path is the path of the logs to collect. It must specify the file, and should not only be a directory. Wildcards are supported as part of the file name, such as /var/log/he.log and /var/log/*.log. However, /var/log is not valid as the path should not be only a directory. stdout is a special value, indicating standard output.

aliyun.logs.$name.format: the log format. Currently only the following formats are supported.

- none: unformatted plain text.
- json: JSON format. One JSON string in each line.
- csv: CSV format.

aliyun.logs.$name.tags: The additional field added when the logs are reported. The format is k1=v1,k2=v2. The key-value pairs are separated by commas, such as aliyun.logs.access.tags="name=hello,stage=test". The logs reported to the storage will contain the name field and the stage field.

If ElasticSearch is used for log storage, the target tag will have a special meaning, indicating the corresponding index in the ElasticSearch.

Fluentd-pilot extension

For most users, the existing features of Fluentd-pilot can meet their requirements. If Fluentd-pilot is unable to meet your requirements, you can:

- Submit an issue at https://github.com/jzwlqx/fluentd-pilot/issues.
- Directly change the code and then raise the PR.