



Sangfor KubeManager

Application Deployment From Localhost

Product Version	6.0
Document Version	01
Released on	Dec. 12, 2021



Copyright © Sangfor Technologies Inc. 2021. All rights reserved.

Unless otherwise stated or authorized, Sangfor Technologies Inc. (hereinafter referred to as "Sangfor") and its affiliates reserve all intellectual property rights, including but not limited to copyrights, trademarks, patents, and trade secrets, and related rights to text, images, pictures, photographs, audio, videos, charts, colors, and layouts as presented in or concerning this document and content therein. Without prior written consent of Sangfor, this document and content therein must not be reproduced, forwarded, adapted, modified or displayed or distributed by any other means for any purpose.

Disclaimer

Products, services or features described in this document, whether wholly or in part, may be not within your purchase scope or usage scope. The products, services or features you purchase must be subject to the commercial contract and terms as agreed by you and Sangfor. Unless otherwise provided in the contract, Sangfor disclaims warranties of any kind, either express or implied, for the content of this document.

Due to product version upgrades or other reasons, the content of this document will be updated from time to time. Unless otherwise agreed, this document is used for reference only, and all statements, information, and recommendations therein do not constitute any express or implied warranties.

Technical Support

For technical support, please visit: <https://www.sangfor.com/en/about-us/contact-us/technical-support>

Send information about errors or any product related problem to tech.support@sangfor.com.

About This Document






This document describes the application deployment guide in the KubeManager localhost.

Intended Audience

This document is intended for:

- Cloud Engineer
- Operations Manager

Note Icons

English Icon	Description
	Indicates an imminently hazardous situation which, if not avoided, will result in death or serious injury.
	Indicates a potentially hazardous situation which, if not avoided, could result in death or serious injury.
	Indicates a hazardous situation, which if not avoided, could result in minor or moderate injury.
	Indicates a hazardous situation, which if not avoided, could result in settings failing to take effect, equipment damage, or data loss. NOTICE addresses practices not related to personal injury.
	Calls attention to important information, best practices, and tips. NOTE addresses information not related to personal injury or equipment damage.

Change Log

Date	Change Description
Oct. 27, 2021	This is the first release of this document.

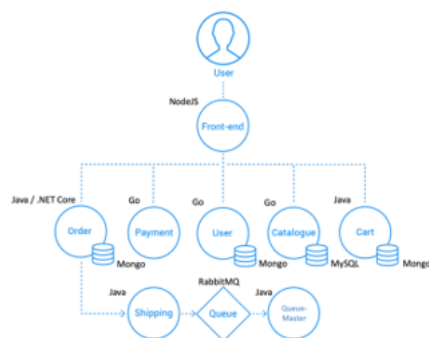
Contents

Technical Support	1
Change Log	2
1 Introduction	4
2 User Cluster Preparation	4
3 Images Preparation	5
4 Services Deployment.....	8
5 Publish Services	11

1 Introduction

In this guide, we will deploy an online sock-shop service. Below is the basic overview for the deployment of an online sock shop.

Sock-Shop System Component



Front-end : User interface services
 Order : Order services
 Order-db : Mongo database for order
 Payment : Payment services
 User : User services
 User-db : Mongo database for user
 Catalogue : Product listing
 Catalogue-db : MySQL database for product listing
 Cart : Shopping cart services
 Cart-db : Mongo database for shopping cart
 Shipping : shipping services

The demonstration will be using the following IP:

Kubemanager IP: 192.168.20.166

Harbor IP: 192.168.20.167

2 User Cluster Preparation

Use KubeManager to create a user cluster with at least one worker node more than 4 Cores 8G.

Nodes							Add Node
<div> <div> <div></div> <div>Cordon</div> </div> <div> <div></div> <div>Drain</div> </div> <div> <div></div> <div>Delete</div> </div> </div> <div>Search</div>							
State	Name	Roles	Version	CPU	RAM	Pods	
Active	node1-c3fb68 192.168.20.162	etcd	v1.16.13 18.6.3	0.3/4 Cores	0.4/9 GiB	15/110	
Active	node2-8235bb 192.168.20.163	Control Plane Worker	v1.16.13 18.6.3	0.4/4 Cores	0.1/4.9 GiB	9/110	
Active	node3-dd33fc 192.168.20.164	Control Plane Worker	v1.16.13 18.6.3	0.4/4 Cores	0.1/4.9 GiB	9/110	

Configuration		Advanced	
	Processor	4 core(s)	
	Memory	8 GB	
	Disk 1	100 GB	
	Disk 2	150 GB	
	CD/DVD 1	None	

3 Images Preparation

1. SSH to Harbor or one of the worker node backend, upload the docker file
(In this demonstration, the docker file name is **sock-shop-image.tar.gz**) to the host. Get the file from Sangfor TAC if needed.

Credentials for SSH

Username: root

Password : Sangfor-paas.237

2. After this, extract the file using the **docker load** command (The extract process may take some time if it consists of many images).

Command: docker load -i sock-shop-image.tar.gz

```
Sangfor:PaaS/host-8cd5ba9c ~ o docker load -i sock-shop-image.tar.gz
7cbbc42c44: Loading layer [=====>] 5.05MB/5.05MB
da07d9b32b00: Loading layer [=====>] 3.584kB/3.584kB
bef6fa0c97dd: Loading layer [=====>] 141MB/141MB
f44ee9e8a8fc: Loading layer [=====>] 947.7kB/947.7kB
da8df155c285: Loading layer [=====>] 3.584kB/3.584kB
a7f17f2c3061: Loading layer [=====>] 3.584kB/3.584kB
1ed133331105: Loading layer [=====>] 2.56kB/2.56kB
4dbfb53514c: Loading layer [=====>] 25.86MB/25.86MB
e66bb4d953ba: Loading layer [=====>] 25.86MB/25.86MB
Loaded image: library/sock-shop/weaveworksdemos/orders:0.4.7
9f8566ee5135: Loading layer [=====>] 5.054MB/5.054MB
d614d652f606: Loading layer [=====>] 875kB/875kB
0038e1fdd596: Loading layer [=====>] 13.4MB/13.4MB
a91e82e93753: Loading layer [=====>] 13.4MB/13.4MB
Loaded image: library/sock-shop/weaveworksdemos/payment:0.4.3
2bfc89c6857f: Loading layer [=====>] 2.56kB/2.56kB
402c813bf853: Loading layer [=====>] 26.34MB/26.34MB
e0b8c2119015: Loading layer [=====>] 26.34MB/26.34MB
Loaded image: library/sock-shop/weaveworksdemos/shipping:0.4.8
cd67935891dc: Loading layer [=====>] 118.5MB/118.5MB
5f70bf18a086: Loading layer [=====>] 1.024kB/1.024kB
82b0a3f79157: Loading layer [=====>] 5.12kB/5.12kB
8b045c0967a2: Loading layer [=====>] 95.12MB/95.12MB
Loaded image: library/sock-shop/openzipkin/zipkin-dependencies:1.4.0
b6ca02dfe5e6: Loading layer [=====>] 128.9MB/128.9MB
54dbf265f995: Loading layer [=====>] 344.6kB/344.6kB
fadfb904b96a: Loading layer [=====>] 483.8kB/483.8kB
45f4d47a35ee: Loading layer [=====>] 4.335MB/4.335MB
f0735232b183: Loading layer [=====>] 5.632kB/5.632kB
```

3. Check whether the images have loaded successfully. In this project, the docker file consists of 17 images.

Command:

- | | |
|----|--------------------------------------|
| 1. | docker images -a grep sock |
| 2. | docker images -a grep sock wc -l |

```
Sangfor:PaaS/host-8cd5ba9c ~ o docker images -a | grep sock
library/sock-shop/mongo:latest
6d11486a97a7 15 months ago 388MB
library/sock-shop/openzipkin/zipkin:latest
40f2f21f6707 16 months ago 158MB
library/sock-shop/redis:alpine
b546e82a6d0e 16 months ago 31.5MB
library/sock-shop/weaveworksdemos/user:0.4.7
c276a3cc0418 3 years ago 35.7MB
library/sock-shop/weaveworksdemos/load-test:latest
3ab3fd5cd04e 4 years ago 570MB
library/sock-shop/weaveworksdemos/catalogue:0.3.5
0bd359b6d6e8 4 years ago 41.2MB
library/sock-shop/rabbitmq:3.6.8
8cdcbee37f62 4 years ago 179MB
library/sock-shop/weaveworksdemos/payment:0.4.3
4f2c23055dcd 4 years ago 32.5MB
library/sock-shop/weaveworksdemos/front-end:0.3.12
b54402ef78a5 4 years ago 120MB
library/sock-shop/weaveworksdemos/shipping:0.4.8
4fc533e8180a 4 years ago 199MB
library/sock-shop/weaveworksdemos/carts:0.4.8
c00473736118 4 years ago 198MB
library/sock-shop/weaveworksdemos/orders:0.4.7
8275c5b9181b 4 years ago 198MB
library/sock-shop/weaveworksdemos/queue-master:0.3.1
76f0de7a12ac 4 years ago 179MB
library/sock-shop/openzipkin/zipkin-dependencies:1.4.0
d34639ff6789 4 years ago 213MB
library/sock-shop/zipkin-mysql:1.20.0
```

```
Sangfor:PaaS/host-8cd5ba9c ~ o docker images -a | grep sock | wc -l
17
```

4. Provide tagging to the image and save the output to b.sh (or another file name). After this, provide file permission to b.sh and run the script.

Command:	
1.	<code>docker images -a grep sock awk '{print "docker tag \"\$1\":\"\$2\" HarborIP/\"\$1\":\"\$2\"}' > b.sh</code>
2.	<code>chmod 777 b.sh</code>
3.	<code>./b.sh</code>
4.	<code>docker images -a grep HarborIP/library/sock awk '{print \$1":"\$2}'</code>

```
Sangfor:PaaS/host-8cd5ba9c ~ x docker images -a | grep sock | awk '{print "docker tag \"$1\":\"$2\" 192.168.20.167/\"$1\":\"$2\"}' > b.sh
Sangfor:PaaS/host-8cd5ba9c ~ o chmod 777 b.sh
Sangfor:PaaS/host-8cd5ba9c ~ o ./b.sh
```

We can see **b.sh** was consists of the tagging of the image.

```
Sangfor:PaaS/host-8cd5ba9c ~ o cat b.sh
docker tag library/sock-shop/mongo:latest 192.168.20.167/library/sock-shop/mongo:latest
docker tag library/sock-shop/openzipkin/zipkin:latest 192.168.20.167/library/sock-shop/openzipkin/zipkin:latest
docker tag library/sock-shop/redis:alpine 192.168.20.167/library/sock-shop/redis:alpine
docker tag library/sock-shop/weaveworksdemos/user:0.4.7 192.168.20.167/library/sock-shop/weaveworksdemos/user:0.4.7
docker tag library/sock-shop/weaveworksdemos/load-test:latest 192.168.20.167/library/sock-shop/weaveworksdemos/load-test:latest
docker tag library/sock-shop/weaveworksdemos/catalogue:0.3.5 192.168.20.167/library/sock-shop/weaveworksdemos/catalogue:0.3.5
docker tag library/sock-shop/rabbitmq:3.6.8 192.168.20.167/library/sock-shop/rabbitmq:3.6.8
docker tag library/sock-shop/weaveworksdemos/payment:0.4.3 192.168.20.167/library/sock-shop/weaveworksdemos/payment:0.4.3
docker tag library/sock-shop/weaveworksdemos/front-end:0.3.12 192.168.20.167/library/sock-shop/weaveworksdemos/front-end:0.3.12
docker tag library/sock-shop/weaveworksdemos/shipping:0.4.8 192.168.20.167/library/sock-shop/weaveworksdemos/shipping:0.4.8
docker tag library/sock-shop/weaveworksdemos/carts:0.4.8 192.168.20.167/library/sock-shop/weaveworksdemos/carts:0.4.8
docker tag library/sock-shop/weaveworksdemos/orders:0.4.7 192.168.20.167/library/sock-shop/weaveworksdemos/orders:0.4.7
docker tag library/sock-shop/weaveworksdemos/queue-master:0.3.1 192.168.20.167/library/sock-shop/weaveworksdemos/queue-master:0.3.1
docker tag library/sock-shop/openzipkin/zipkin-dependencies:1.4.0 192.168.20.167/library/sock-shop/openzipkin/zipkin-dependencies:1.4.0
docker tag library/sock-shop/zipkin-mysql:1.20.0 192.168.20.167/library/sock-shop/zipkin-mysql:1.20.0
docker tag library/sock-shop/weaveworksdemos/catalogue-db:0.3.0 192.168.20.167/library/sock-shop/weaveworksdemos/catalogue-db:0.3.0
docker tag library/sock-shop/weaveworksdemos/user-db:0.4.0 192.168.20.167/library/sock-shop/weaveworksdemos/user-db:0.4.0
```


As shown above, the images have been tagged.

```
Sangfor:PaaS/host-8cd5ba9c ~ x docker images -a | grep 192.168.20.167/library/sock | awk '{print $1":"$2}'
192.168.20.167/library/sock-shop/mongo:latest
192.168.20.167/library/sock-shop/openzipkin/zipkin:latest
192.168.20.167/library/sock-shop/redis:alpine
192.168.20.167/library/sock-shop/weaveworksdemos/user:0.4.7
192.168.20.167/library/sock-shop/weaveworksdemos/load-test:latest
192.168.20.167/library/sock-shop/weaveworksdemos/catalogue:0.3.5
192.168.20.167/library/sock-shop/rabbitmq:3.6.8
192.168.20.167/library/sock-shop/weaveworksdemos/payment:0.4.3
192.168.20.167/library/sock-shop/weaveworksdemos/front-end:0.3.12
192.168.20.167/library/sock-shop/weaveworksdemos/shipping:0.4.8
192.168.20.167/library/sock-shop/weaveworksdemos/carts:0.4.8
192.168.20.167/library/sock-shop/weaveworksdemos/orders:0.4.7
192.168.20.167/library/sock-shop/weaveworksdemos/queue-master:0.3.1
192.168.20.167/library/sock-shop/openzipkin/zipkin-dependencies:1.4.0
192.168.20.167/library/sock-shop/zipkin-mysql:1.20.0
192.168.20.167/library/sock-shop/weaveworksdemos/catalogue-db:0.3.0
192.168.20.167/library/sock-shop/weaveworksdemos/user-db:0.4.0
```

5. Login to Harbor and prepare to push the images (Default Harbor login password is **Harbor12345**).

Command:	
1.	mkdir -p /etc/docker/certs.d/HarborIP
2.	curl -kL http://HarborIP/api/systeminfo/getcert -o /etc/docker/certs.d/HarborIP/ca.crt
3.	docker login -u admin -p Harbor12345 HarborIP

```
Sangfor:PaaS/host-8cd5ba9c ~ x mkdir -p /etc/docker/certs.d/192.168.20.167
Sangfor:PaaS/host-8cd5ba9c ~ o curl -kL http://192.168.20.167/api/systeminfo/getcert -o /etc/docker/certs.d/192.168.20.167/ca.crt
% Total % Received % Xferd Average Speed Time Time Time Current
Dload Upload Total Spent Left Speed
100 169 100 169 0 0 122k 0 --:--:-- --:--:-- --:--:-- 165k
100 1094 100 1094 0 0 1350 0 --:--:-- --:--:-- --:--:-- 0
Sangfor:PaaS/host-8cd5ba9c ~ o docker login -u admin -p Harbor12345 192.168.20.167
WARNING! Using --password via the CLI is insecure. Use --password-stdin.
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store
Login Succeeded
Sangfor:PaaS/host-8cd5ba9c ~ o
```

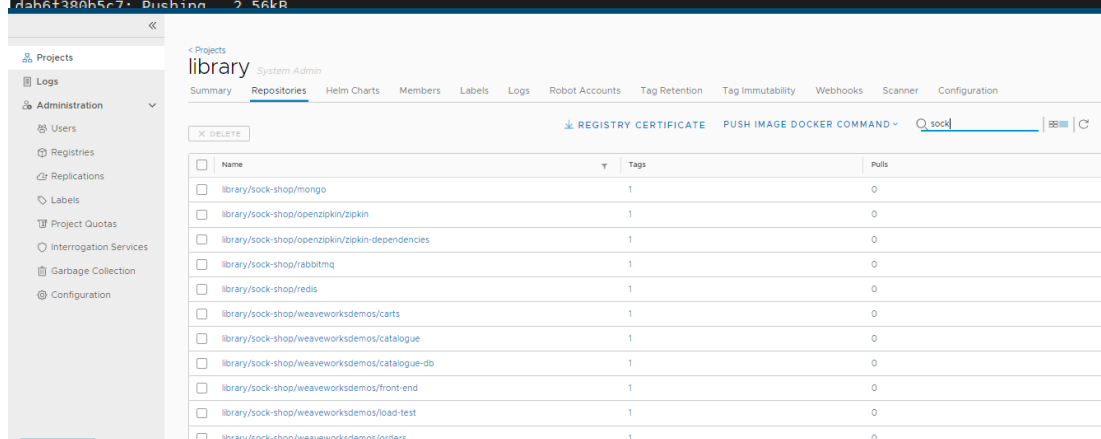
6. Push the images to Harbor. First, we will group all the **docker push** images commands and save the output to a.sh (or another file name). After this, grant file permission to a.sh and execute it.

Command:	
1.	docker images -a grep HarborIP/library/sock awk '{print "docker push \$1":"\$2}" > a.sh
2.	chmod 777 a.sh
3.	./a.sh

```

Sangfor:PaaS/host-8cd5ba9c ~ o docker images -a | grep 192.168.20.167/library/sock | awk '{print "docker push \"$1\":\"$2\"' > a.sh
Sangfor:PaaS/host-8cd5ba9c ~ o chmod 777 a.sh
Sangfor:PaaS/host-8cd5ba9c ~ o ./a.sh
The push refers to repository [192.168.20.167/library/sock-shop/mongo]
0724475c39ec: Pushing [=====] 14.34kB
dah6f380b5c7: Pushing [=====] 2.56kB

```

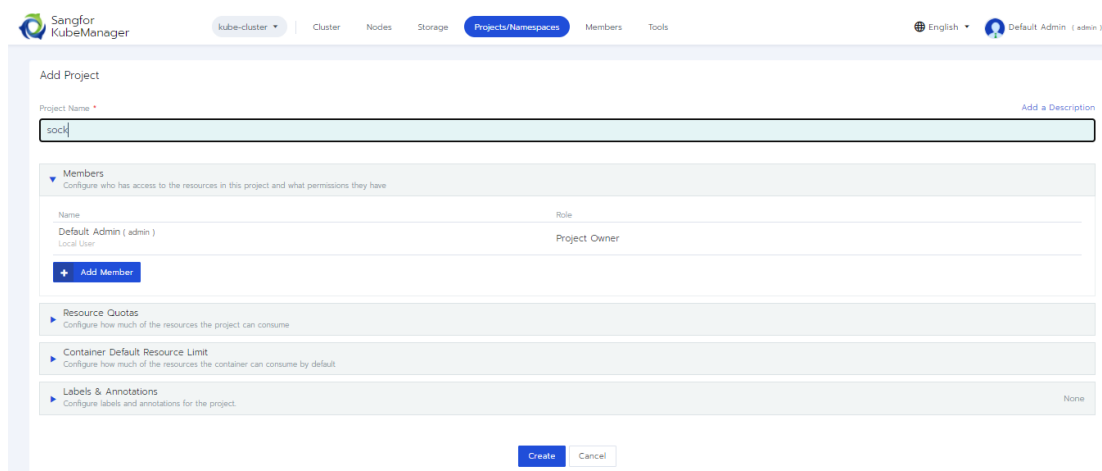
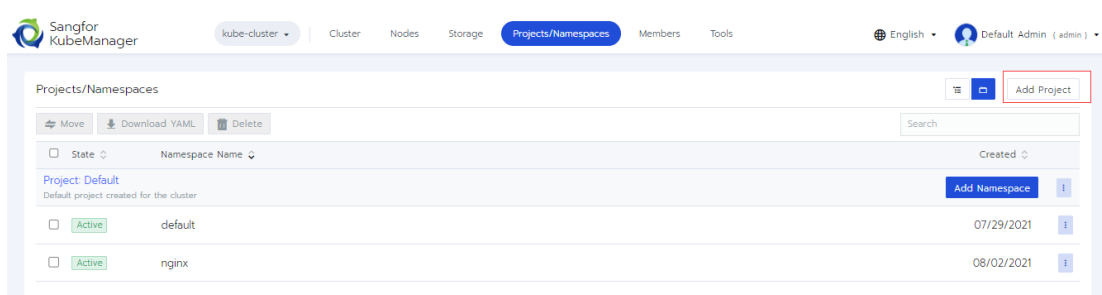


7. After this, the images should be able to push to Harbor successfully.

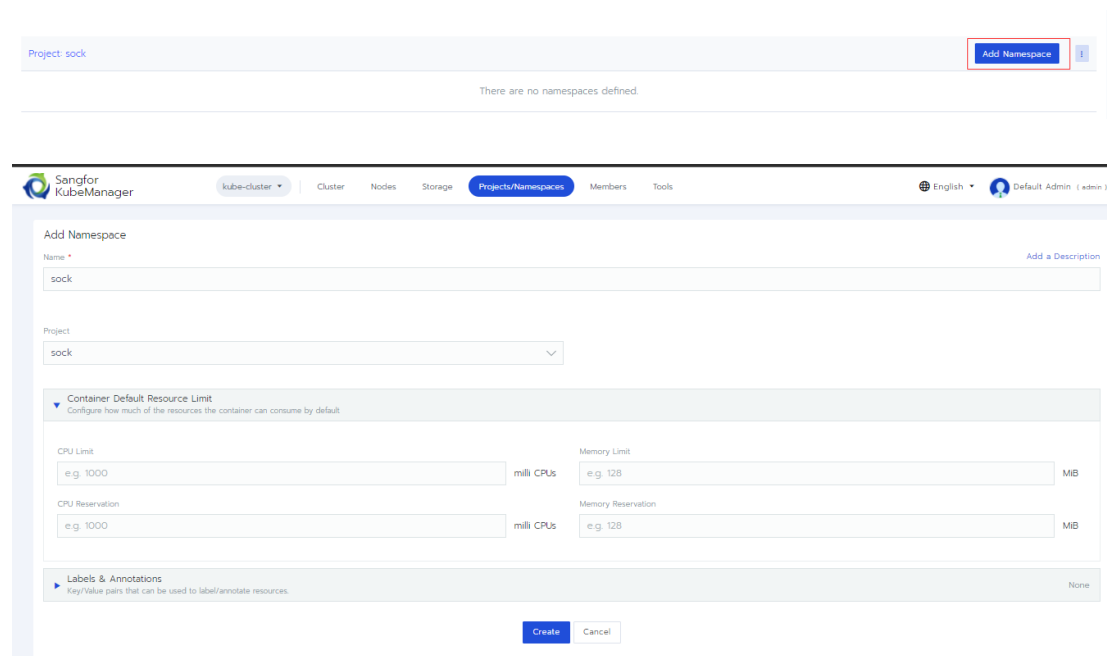
4 Services Deployment

1. Login KubeManager, create a new Namespaces and project.

Project Deployment:

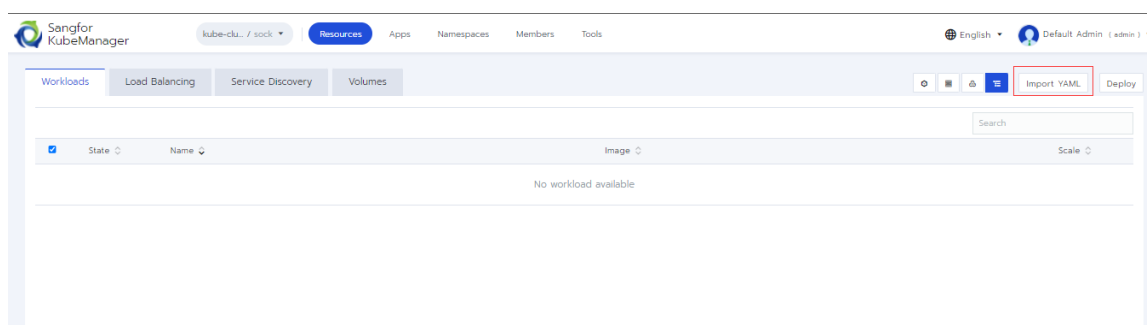


Namespace Deployment:



The screenshot shows the 'Add Namespace' form in Sangfor KubeManager. At the top, there is a 'Project: sock' label and an 'Add Namespace' button. Below this, a message states 'There are no namespaces defined.' The main form area is titled 'Add Namespace' and includes a 'Name' field with the value 'sock' and an 'Add a Description' link. A 'Project' dropdown menu is set to 'sock'. Under the 'Container Default Resource Limit' section, there are four input fields: 'CPU Limit' (e.g. 1000), 'Memory Limit' (e.g. 128), 'CPU Reservation' (e.g. 1000), and 'Memory Reservation' (e.g. 128). The units are 'milli CPUs' and 'MB'. A 'Labels & Annotations' section is also present with a 'None' value. At the bottom, there are 'Create' and 'Cancel' buttons.

2. Browse to the project page, import the image by importing **YAML files**.

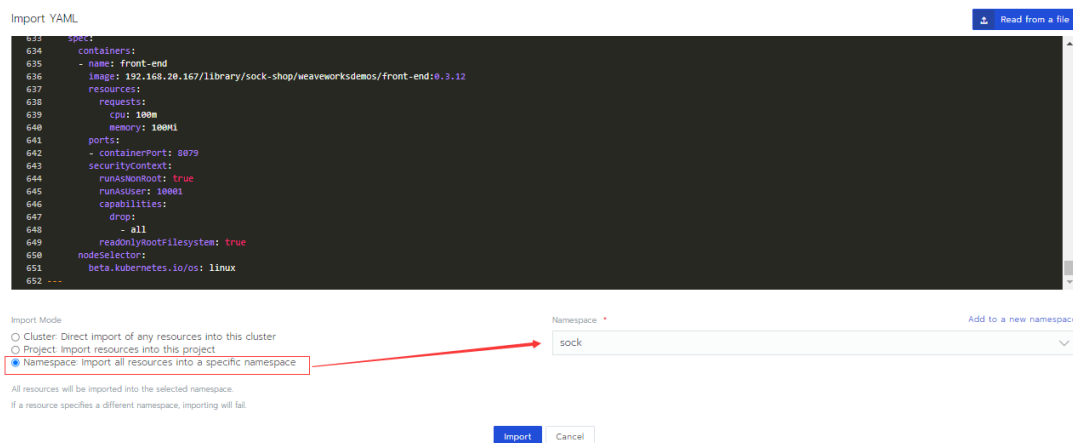


The screenshot shows the 'Resources' page in Sangfor KubeManager. The breadcrumb navigation is 'kube-clu... / sock'. The 'Resources' tab is selected. In the top right, there is an 'Import YAML' button. Below the navigation bar, there are tabs for 'Workloads', 'Load Balancing', 'Service Discovery', and 'Volumes'. A search bar is present. Below the search bar, there is a table with columns 'State', 'Name', 'Image', and 'Scale'. The table is currently empty, and a message 'No workload available' is displayed.

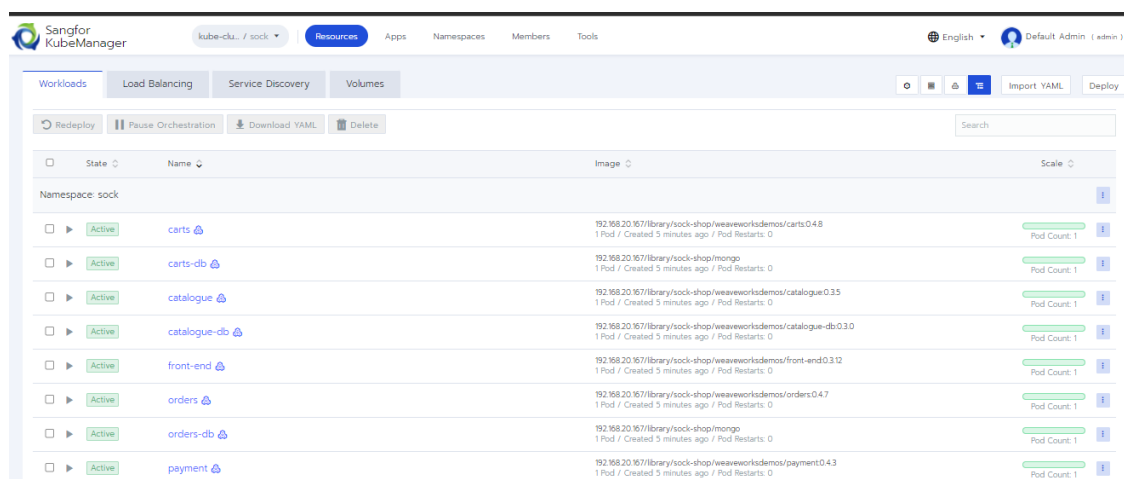


If the docker project consists of more than one workload, it's suggested to have a YAML file to define the relationship between workloads and how they should be deployed.

Copy and paste the YAML file content. Remember to replace all the Harbor Ip according to the environment.

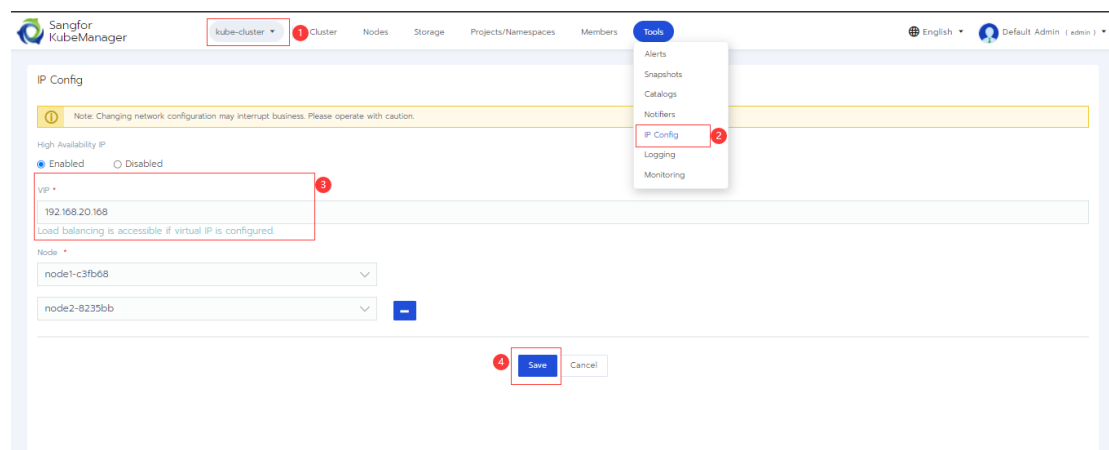


3. After the workload is created, it will need to wait for some time until all the workload turns to active status.

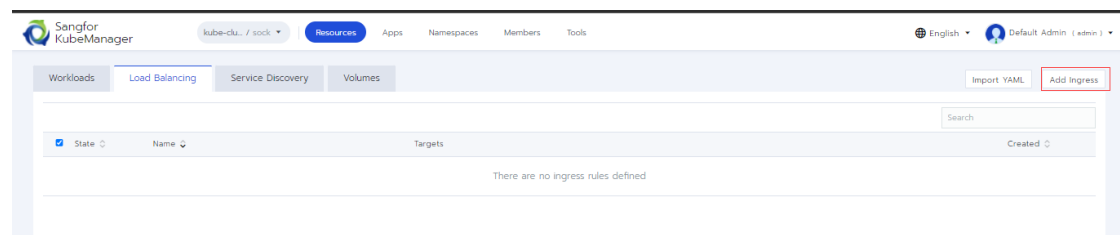


5 Publish Services

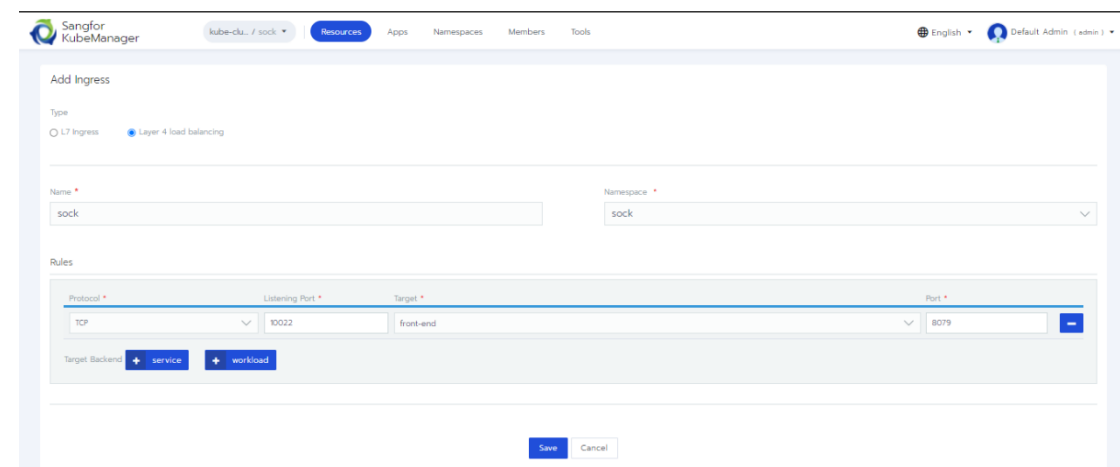
1. Access cluster mode, proceed to **Tools > IP config**, set a virtual IP for the cluster, and click **Save** once it is completed.



2. Access the project and click **Add Ingress** to configure for Load Balancing.

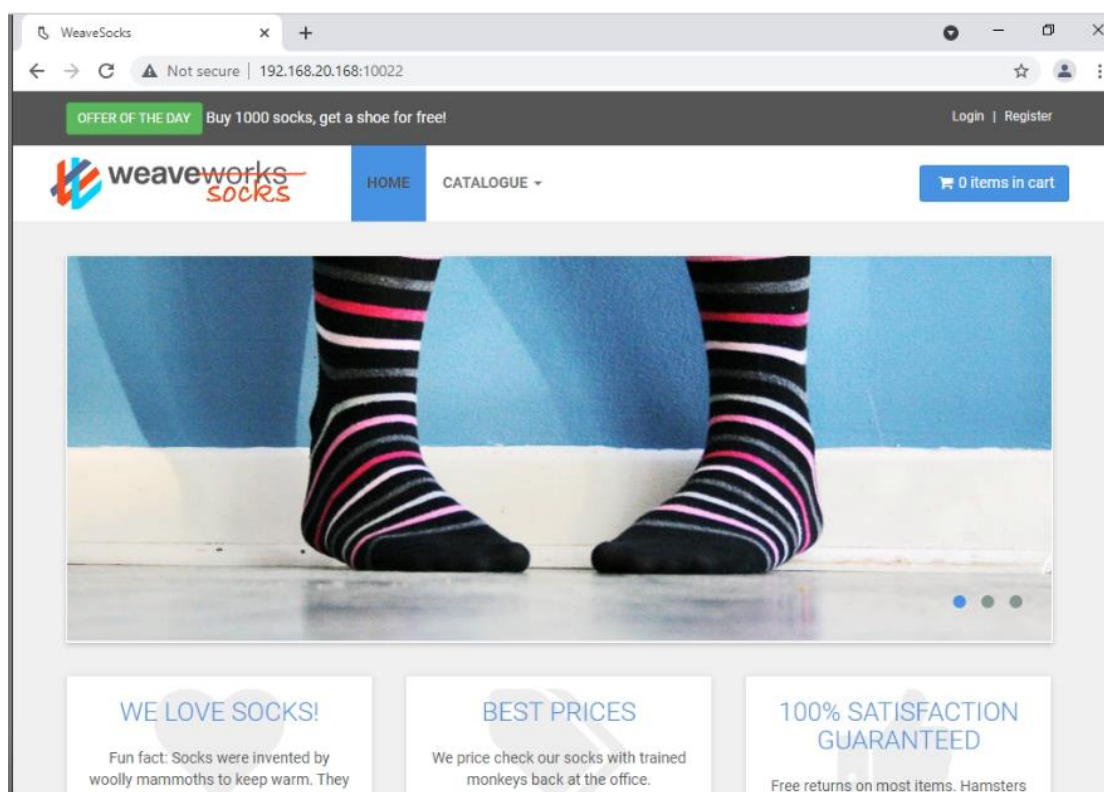


Provide a name for the project. After this, configure the load balancing policy. In this project, the **front-end** workload interacts with the user. Therefore it will be the target for load balancing. The port used for the workload is 8079, while the listening port for localhost can be **any value**.





3. Access the web service through the VIP with the listening port. If the project is deployed successfully, the web services should be accessible by the browser.





SANGFOR

