Sangfor PaaS Platform

Test Guidance V1.0

Sangfor Technologies Inc.

January 2021

Revisions

Rev.	Author	Date	Abstract
V1.0	Li Junpeng Zhao Zhenyang	2021-01-19	Sangfor PaaS Platform Experiment Manual V1.0

Contents

1 Preface

1.1 Introduction

KubeManager is a cloud-native multi-Kubernetes management platform built by Sangfor with a simple structure, stable performance, and easiness of use. It is also a container management platform for companies using containers. The platform, based on Docker + Kubernetes technology stack, simplifies the process of using Kubernetes (K8s).

This experiment manual provides guidance on how to use KubeManager, the PaaS platform developed by Sangfor.

1.2 Content Description

This experiment manual covers the following experiments[1].

- Experiment 1: Creating a K8s Cluster

  Start from the creation of a K8s cluster on the platform and introduce created cluster parameters and the function of managing the K8s cluster's lifecycle from the KubeManager interface.

- Experiment 2: Migrating Applications to a K8s Cluster

  Download open-source images from hub.docker.com, deploy container applications on KubeManager, and realize the deployment and use of general container applications on the platform.

- Experiment 3: K8s Storage Configuration and Use

  Master the configuration and docking details for K8s clusters to use the Sangfor aSAN storage plug-in by configuring the storage of K8s clusters, including SCP licensing, the iSCSI server of aCloud clusters, and storage servers and storage classes of K8s clusters. Finally, deploy a workload to verify the use of storage with data volumes.

- Experiment 4: Enabling K8s Cluster Monitoring

  Enable cluster monitoring in a K8s cluster, and know the real-time monitoring indicators and events of the entire K8s cluster through monitoring.

- Experiment 5: Enabling K8s Cluster Logs

  Enable logs in a K8s cluster, and know the log collection and retrieval of K8s applications by deploying workloads and enabling log collection.

- Experiment 6: Creating a K8s Application

  Create a project and namespace. Select the project and namespace to deploy the K8s workloads and container applications. Then, deploy an Nginx application and configure port mapping, storage of data volumes, and container parameters. This experiment illustrates the configuration for fully refined support of workloads and allows us to further deploy more abundant container applications.

  Through service publishing, we can know how to publish L4 services for a service and how to configure the high-availability network port for L4 services and provide the south-north traffic for external access.

- Experiment 7: Use of App Store

[1]Zoom in the figures/texts in a browser if they are too small.

In this experiment, we upload a tomcat application image and the corresponding helm chart package to the built-in registry through docker push and deploy K8s applications in a way similar to software package management using the app store from the KubeManager interface.

1.3 Experiment Tools

Name	Download Link	Purpose
chrome	https://chrome.google.com	Provide a browser environment to access KubeManager

1.4 Experiment Resources

- The following infrastructure software and servers must be ready for the experiment:

Server	Required software versions or specifications
Sangfor aCloud	v6.2.0
Sangfor SCP	v6.2.0
Sangfor KubeManager[2]	v6.0

- The cloud license of containers is available, and there are sufficient licensed nodes for creating K8s clusters.
- The security correlation license has been granted.
- The VM template is available for creating VMs of cluster nodes.
  - VM template for K8s cluster nodes on the "CentOS-7-x86_64-Minimal-1908-user_cluster.vma x86_64" platform
- IP Resources

Type of IP Address	Purpose	Quantity
IP addresses for creating K8s cluster nodes	One IP address for each K8s cluster node	The number of nodes
Network port IP addresses of K8s clusters	One network port IP address for each K8s cluster to publish the L4 workload service	1
Access IP address of aCloud iSCSI server	For K8s clusters to dock with the storage; one IP address and one access IP address for each physical host of aCloud	The number of physical hosts + 1

[2] KubeManager access includes the KubeManager VIP for accessing the KubeManager management interface and the Harbor VIP for accessing the built-in registry. Image search is available.

2 Experiment 1: Creating a K8s Cluster

2.1 Introduction

2.1.1 About the Experiment

In this experiment, we create a K8s cluster on the interface and manage the K8s cluster in a standard K8s cluster environment.

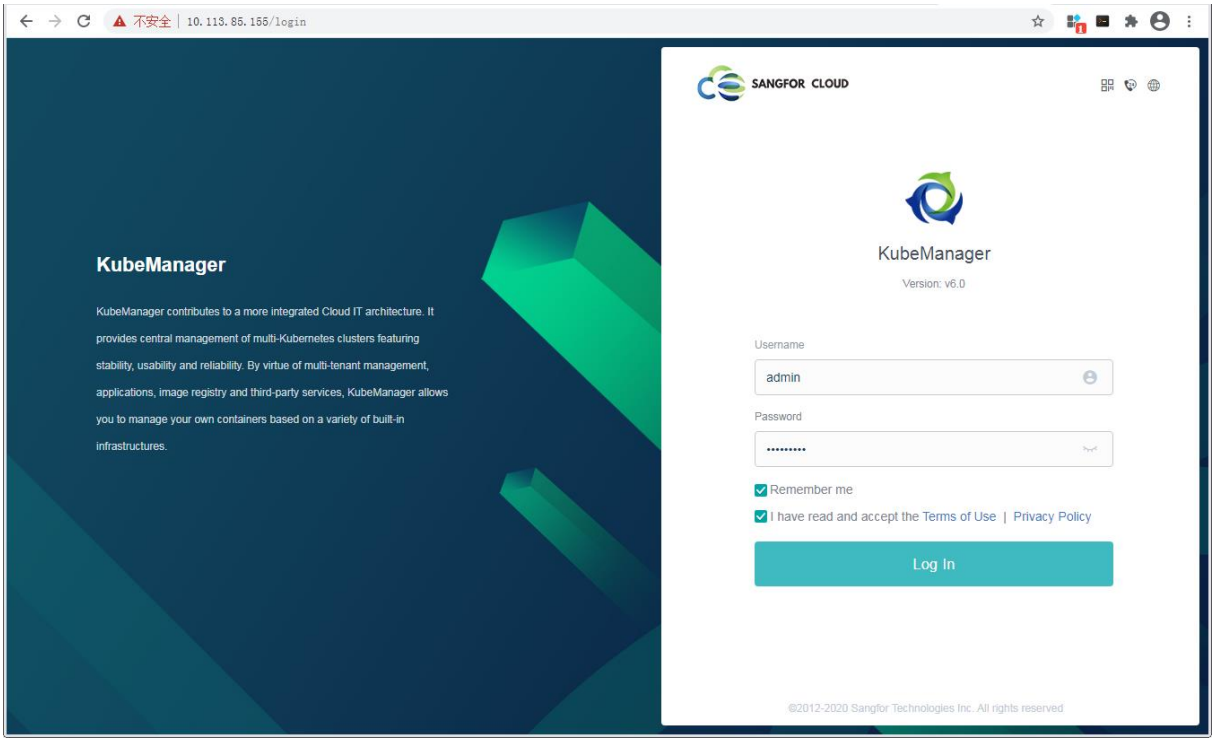
2.1.2 Purpose

- To find out how to create and manage a K8s cluster with KubeManager
- To know configuration parameters for creating the K8s cluster

2.2 Steps

2.2.1 User Login

Step 1: Enter "KubeManager VIP" in the browser and log in to the console with your KubeManager account.

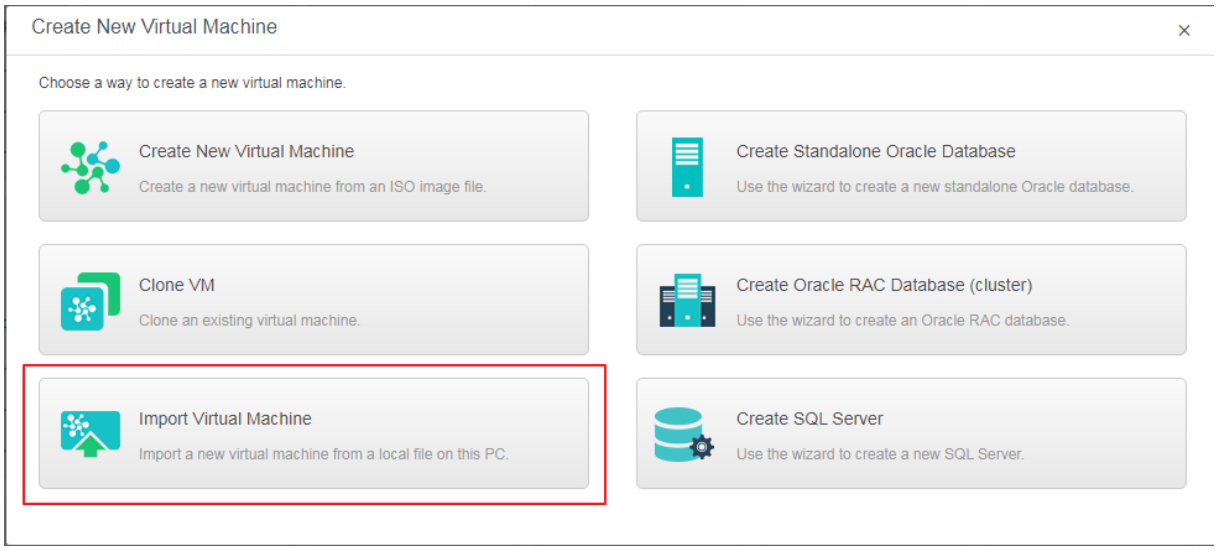


2.2.2 Creating a K8s Cluster

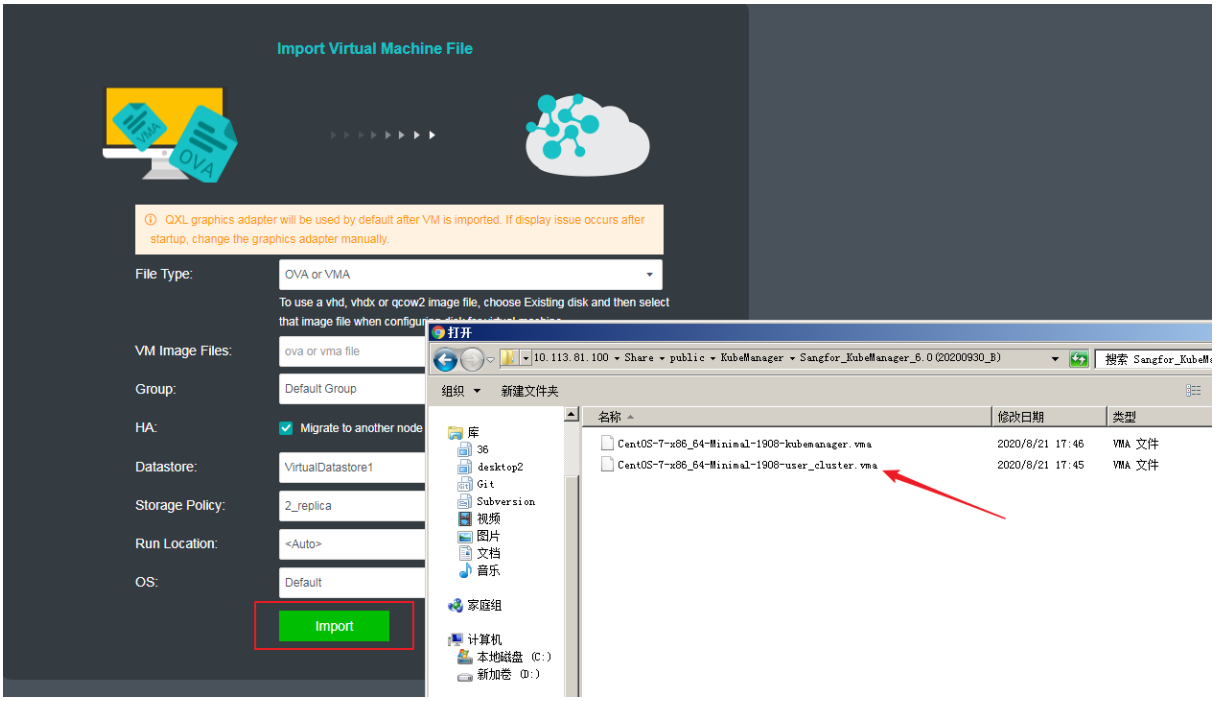
Step 1: Create VMs. Before creating a K8s cluster, import the VMs of K8s cluster nodes with aCloud (Sangfor cloud platform) and get the VMs ready. On the interface, set and record the node IP addresses, username, and password[3].

- On the aCloud platform, click "Import VM".

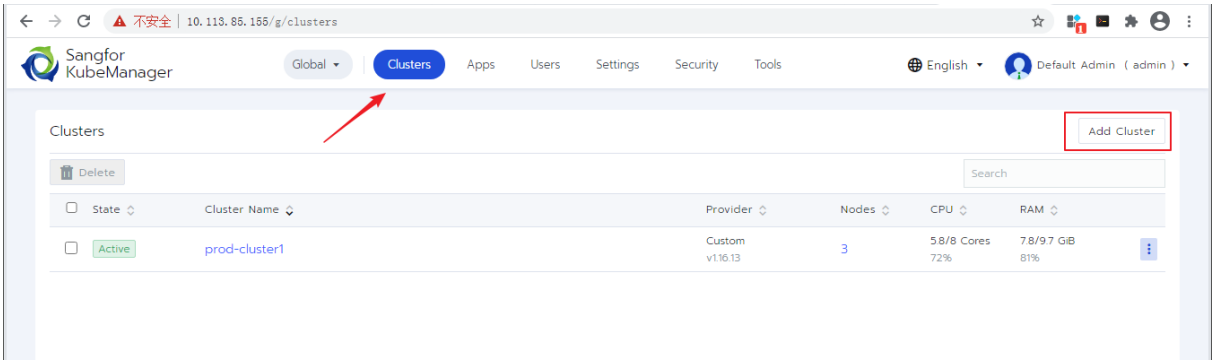
[3] CentOS-7-x86_64-Minimal-1908-user_cluster.vma is an x86_64 VM template for K8s cluster nodes. Other than creating K8s cluster nodes by import, you may also create new VMs by Sangfor aCloud's cloning function. The default username and password for VMs created by vma are: ****/***g**r-****.**7. Contact relevant staff for the password.



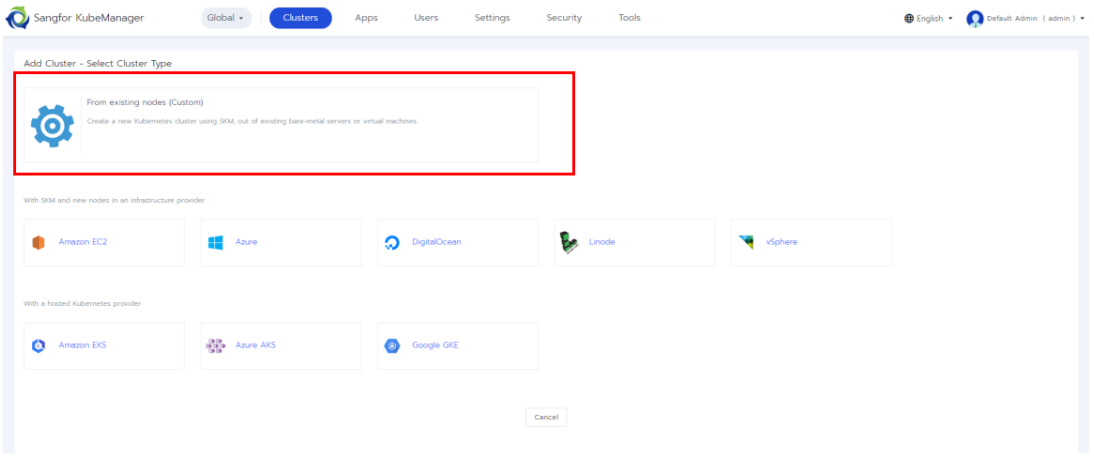
- On the aCloud platform, click "Import".



Step 2: On the Cluster interface, click "Add Cluster".

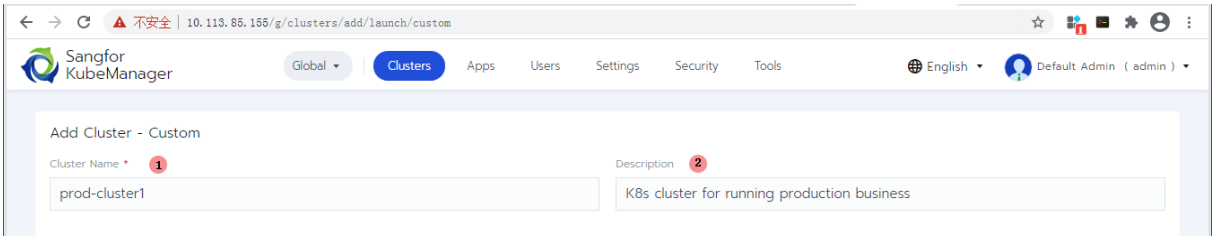


Step 3: Select the cluster type and click "Custom".



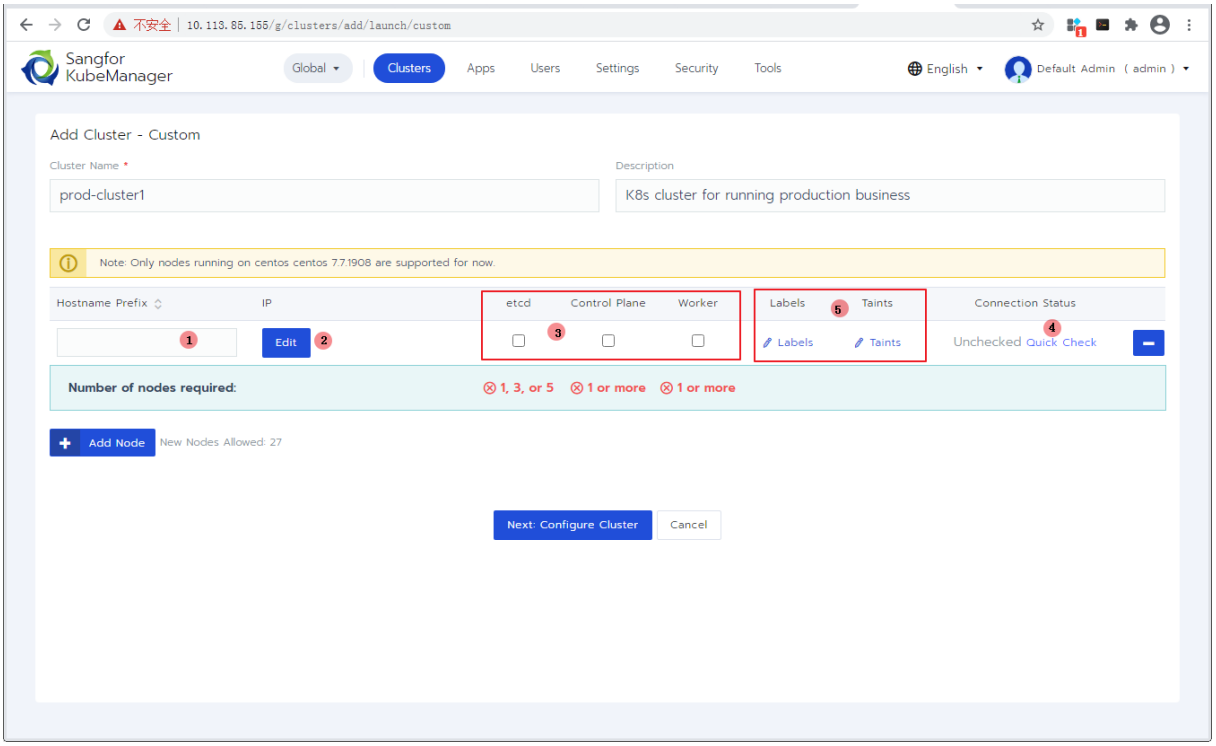
Step 4: Add cluster-custom.

- Cluster Name

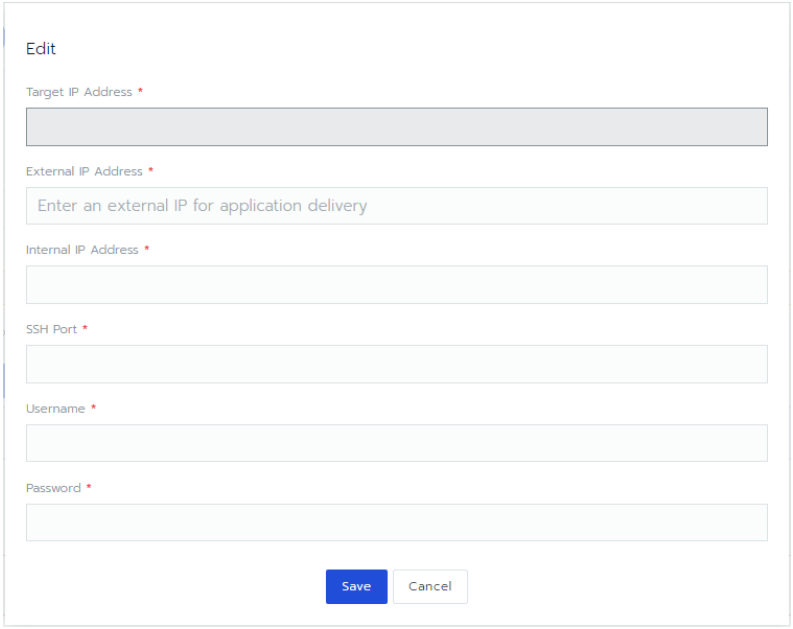


1. Set the cluster name.
2. Click "Add a Description" and briefly describe the K8s cluster.

- Add Host



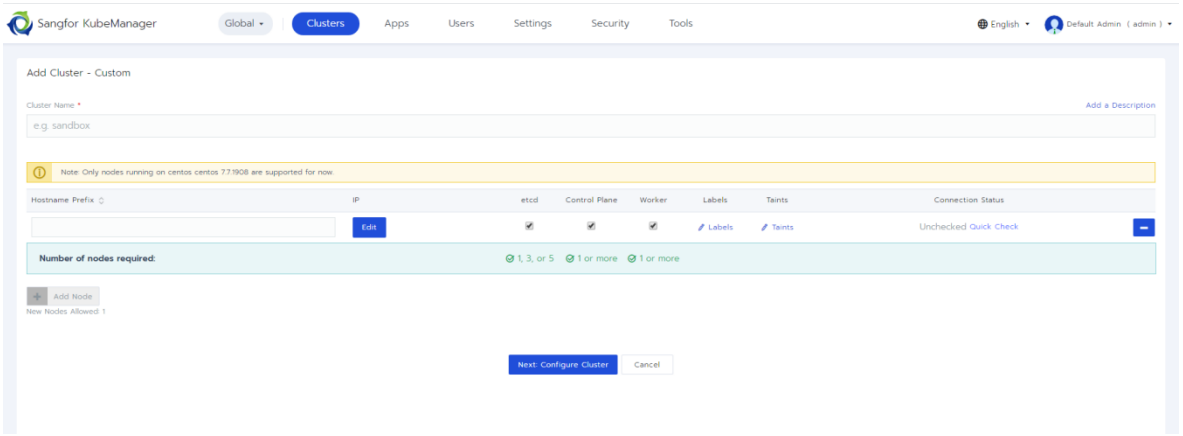
1. Configure the hostname.
2. Enter the IP address of the host and click "Edit" to configure the access information.



Config Items for Access Info	Description
Access IP	SSH IP address for KubeManager to connect with the host
External IP	For north-south external access traffic
Internal IP	For east-west internal network of the K8s cluster
SSH Port	SSH port of node
Host Username	SSH username of node[4]
Host Password	SSH password of node

As shown in the figure, if the node uses a single NIC, the Access IP, External IP, and Internal IP are the same.

3. Check the node role.



Set roles for nodes in the K8s cluster. Node is a computing resource in the cluster and can be a physical server or virtual machine (VM). Anyone capable of communication can serve as a K8s cluster node. You only need to provide the IP address, SSH username, and password. According to their respective roles, we divide these nodes into three categories: etcd nodes, control nodes[5], and

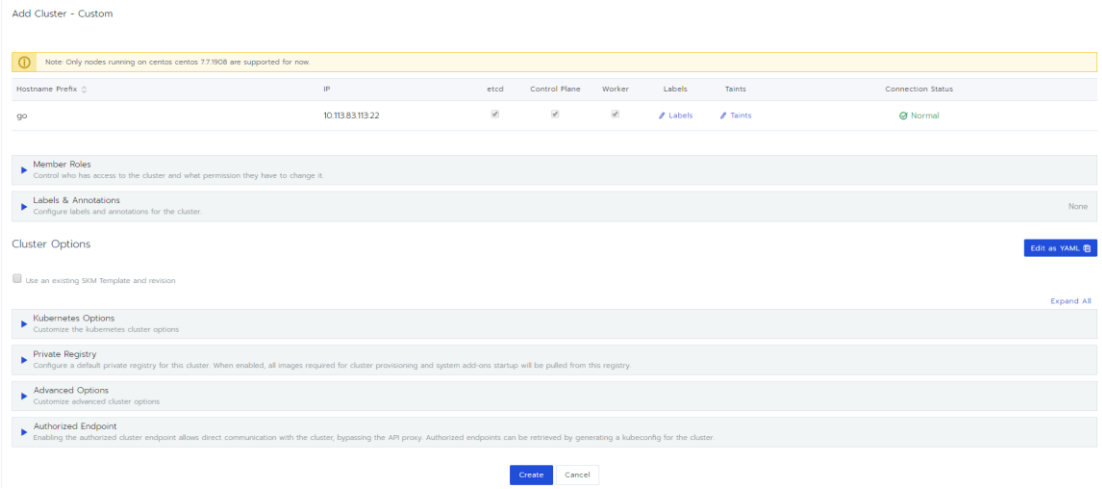
[4]The default username and password are ****/***g**r-****.**7. Contact relevant staff for the password.

[5] Typically, open-source native K8s control nodes are called master nodes. Since the use of the nodes on the KubeManager platform interface emphasizes the control node role, the node name is changed to "control".

worker[6] nodes. A K8s cluster must have at least one etcd node, one control node, and one worker node.

4. Test the connection status by clicking "Start Test". Every node must pass the test before proceeding to the next step.
5. Optional configuration: Tag and Taint[7]

Step 5: Configure the cluster.



Configure cluster parameters on the interface. Generally, no special configuration is required since most scenarios have been covered by the default values on the interface. Just click "Create".

If you need custom configuration, please refer to the following cluster creation parameters:

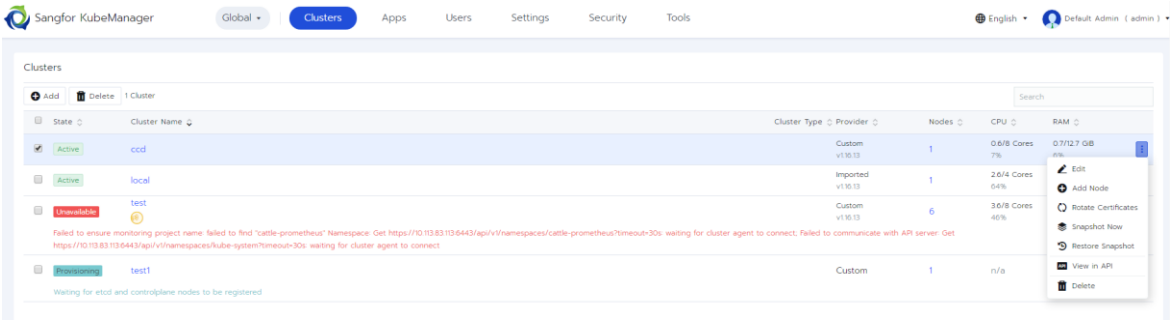
Config Item	Default Value	Description
Role	admin	Control the users that can access the cluster and their permissions of changing the cluster
Tag/Comment	-	Configure tags and comments for the cluster
K8s Version	v1.16.13-sangfor1-1	Select the K8s version
Network Drive	Calico	Select network plug-ins for the K8s cluster
Network Planning	Automatic selection	You can customize the pod network and service network scope of the K8s cluster
Private Registry	Disabled	Configure the private registry for the cluster. When the cluster is being created, all the required images will be pulled from this registry. This item is disabled by default so that the built-in registry of KubeManager is used. If it's enabled, you need to use the mapped registry.
Nginx Ingress	Enabled	Whether to enable Nginx Ingress as the ingress controller of K8s
NodePort Scope	30000-32767	Use range of NodePort in the K8s cluster
Monitoring Indicator	Enabled	Monitoring indicators of the K8s cluster
Pod Security Policy	Disabled	Global PSP security policy of the K8s cluster. It is recommended to disable this parameter during tests. You may enable it during production after strict tests, to ensure that services can run normally.
Etcd Backup Storage	Local	Refers to the etcd data backup of the etcd node. It is backed up to local servers by default or docked with S3 of aws. The

[6] The worker node role. The recommended resource configuration in the production environment is an 8-core 16 GB worker node.

[7] Clicking "Tag" can set labels for hosts, to facilitate the subsequent provision of matching of classification, scheduling, etc. Clicking "Taints" can set the taint attribute of a node. For more details, see Taints and Tolerations on the official site.

Config Item	Default Value	Description
		default backup period is 12 hours, and 6 copies are backed up.
Authorized Cluster Access Address	Enabled	This address can be used to directly access the K8s API SERVER, bypassing the KubeManager API agent.

2.2.3 Cluster Management

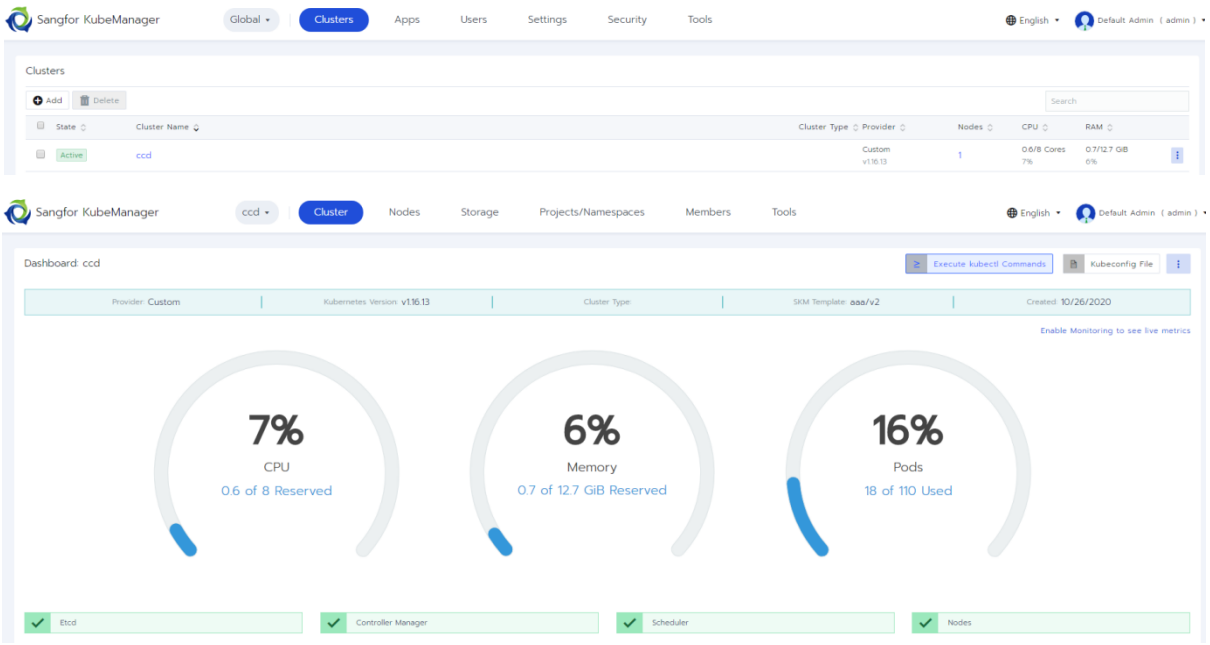


The KubeManager interface provides rich cluster lifecycle management functions and allows you to conveniently scale up/out by adding hosts from the cluster perspective. Click "Cluster Name". The dashboard is displayed, where you can view the performance data and events of the cluster in real time.

2.2.4 Command Line Entrance of Cluster

Step 1: On the Cluster interface, click "Cluster Name".

Step 2: The cluster dashboard is displayed. Click "Execute kubectl Commands".

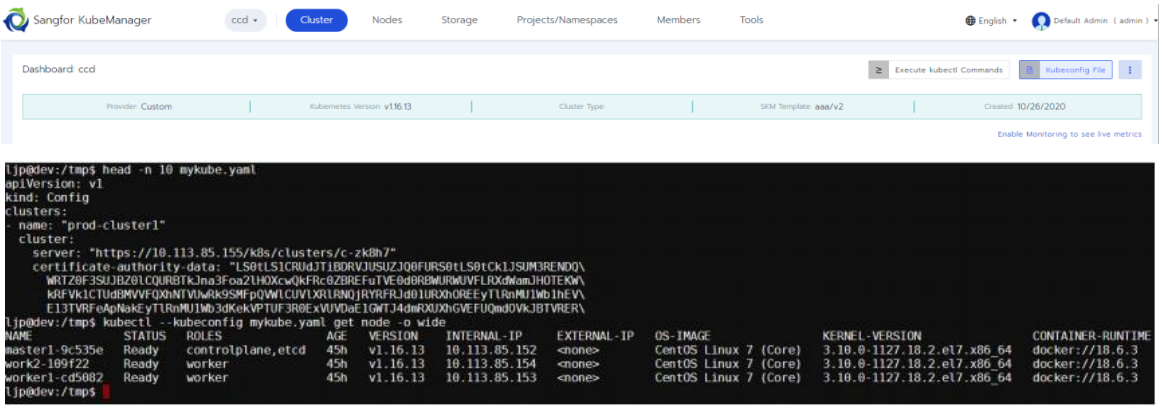


Here, the command line supports all kubectl commands and allows you to have all-round cluster control in command line mode. It is also convenient to use scripts for automatic deployment in this mode.

2.3 Verification

Step 1: On the Cluster interface, click Cluster command-line entrance. Run **kubectl get node -o wide** to check whether the K8s node in the result is the VM when the cluster was created.

Step 2: On the Cluster interface, click "Kubeconfig File". Download the cluster configuration file (for example, download and name it **mykube.yaml**). Then, download kubectl and run with the downloaded cluster configuration file. Run kubectl to create the K8s cluster for command line management, and check whether the K8s cluster created for status verification is healthy.



2.4 Questions for Discussion

When creating a K8s cluster, we use the customization method. Please think about:

1. Do the VM nodes of a custom cluster support other virtualization platforms or physical servers besides the VMs created on aCloud?
2. In addition to creating the custom cluster, what other cluster types are supported? What are they under each type?
3. Can the K8s cluster be created with only one server (VM/physical server)? What roles does the server need to meet in this case?

3 Experiment 2: Migrating Applications to a K8s Cluster

3.1 Introduction

3.1.1 About the Experiment

By deploying the LogicalDOC container application on KubeManager, we can migrate traditional docker applications to a K8s cluster managed by KubeManager.

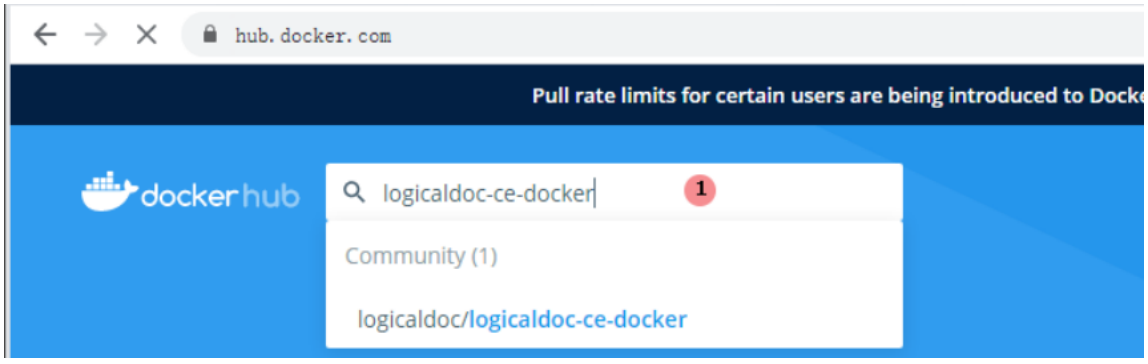
3.1.2 Purpose

- To deploy LogicalDOC DMS - community edition (usually "LogicalDOC CE" for short). It is a Web-based document management system aiming at effectively managing large document archives.
- To find out the configuration and deployment of traditional container applications on KubeManager.
- To have a rough idea of the steps to migrate applications:
    1. Search and download (pull) the container images from hub.docker.com, or build your own container images.
    2. Share images to the built-in registry with docker push.
    3. According to the deployment requirements of container images, configure and deploy the workload on the management interface of KubeManager.
    4. Publish services and access container applications.

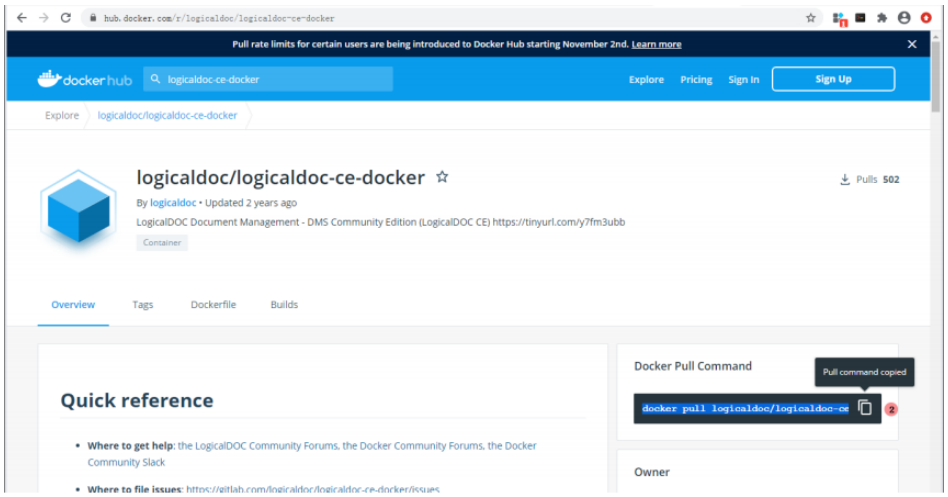
3.2 Steps

3.2.1 Preparation of Container Images

Step 1: Enter https://hub.docker.com/ in the browser and search for the image **logicaldoc-ce-docker**.



Step 2: Copy the download address of the image.



Step 3: On a VM with docker, run the **docker pull** command to download the image.

```
ljp@dev:~$ docker pull logicaldoc/logicaldoc-ce-docker
Using default tag: latest
latest: Pulling from logicaldoc/logicaldoc-ce-docker
e7cb83f94a46: Downloading [=>                                 ]  1.473MB/48.3MB
e2af7c2bd543: Download complete
3b5286ee17a3: Download complete
7cd74685af84: Download complete
3f971e46f30b: Download complete
4ef6c2ba70ca: Download complete
8eecd965c355: Download complete
8c75564a15e6: Downloading [==>                                ]  10.71MB/262.1MB
5bea7d00cb9c: Download complete
32a19a865b82: Download complete
7c669b936fd4: Download complete
0e40d679ebc7: Download complete
30e90bd40924: Download complete
2f7f06f51224: Downloading [=======>                           ]  8.016MB/55.62MB
9fe4992e8c72: Waiting
406c6f490b16: Waiting
63eec291d3b2: Waiting
```

Step 4: Run the **docker tag** command to create the container and tag referencing the built-in registry.

docker tag logicaldoc/logicaldoc-ce-docker: latest 10.113.85.156/library/logicaldoc-ce-docker: latest

Where, 10.113.85.156 is the Harbor VIP address.

Step 5: Run the **docker push** command to push the container image to the built-in registry.

You may encounter the error message "Get https://ip/v2/: x509: certificate signed by unknown authority" when running the **docker push** command. This is because the registry uses the self-signed SSH license of HTTP or HTTPS. You must ensure that the docker configuration meets the insecure configuration requirements.

You must log in to the system (docker login) to push images to the registry with docker push.

- Configuration and login of docker insecure

  sudo mkdir -p /etc/docker/certs.d/10.113.85.156
  sudo curl -kL https://10.113.85.156/api/systeminfo/getcert\
  -o /etc/docker/certs.d/10.113.85.156/ca.crt
  docker login -u admin 10.113.85.156 # Enter password as prompted

```
ljp@dev:~$ sudo mkdir -p /etc/docker/certs.d/10.113.85.156
[sudo] password for ljp:
ljp@dev:~$     sudo curl -kL https://10.113.85.156/api/systeminfo/getcert \
>      -o /etc/docker/certs.d/10.113.85.156/ca.crt
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  1094  100  1094    0     0   133k      0 --:--:-- --:--:-- --:--:--  133k
ljp@dev:~$ docker login -u admin 10.113.85.156 # 根据提示输入密码
Password:
WARNING! Your password will be stored unencrypted in /home/ljp/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
ljp@dev:~$
```

- Run the **docker push** command to push images to the built-in registry.

  docker push 10.113.85.156/library/logicaldoc-ce-docker: latest

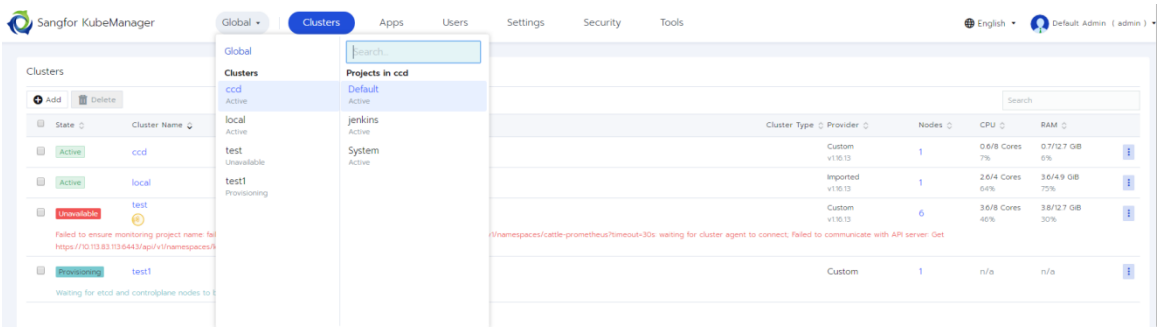
```
ljp@dev:~$ docker tag logicaldoc/logicaldoc-ce-docker:latest 10.113.85.156/library/logicaldoc-ce-docker:latest
ljp@dev:~$ docker push 10.113.85.156/library/logicaldoc-ce-docker:latest
The push refers to repository [10.113.85.156/library/logicaldoc-ce-docker]
00efd4a895d4: Pushed
f486e363906d: Pushing [==================================================>]  178.5MB
60522c513bf5: Pushing [=================>                                 ]  151.1MB/418.8MB
e3ba0f55a074: Pushed
f5d74a0c0ef8: Pushed
013aa0c438c6: Pushed
8134a713eca4: Pushed
57d7b5e73bbe: Pushed
ef82c00ad427: Pushed
324569887367: Pushing [=============>                                     ]  153.4MB/575.7MB
bab1a2822695: Pushed
95715e9be169: Pushed
0ec09adc9099: Pushed
5aa1d1d95cc6: Pushing [=================================>                 ]    114MB/143.7MB
993a2f3a8339: Pushing [=====>                                             ]  1.663MB/16.36MB
45eacd24d590: Waiting
e7752b410e4c: Waiting
```

3.2.2 Creating Workload

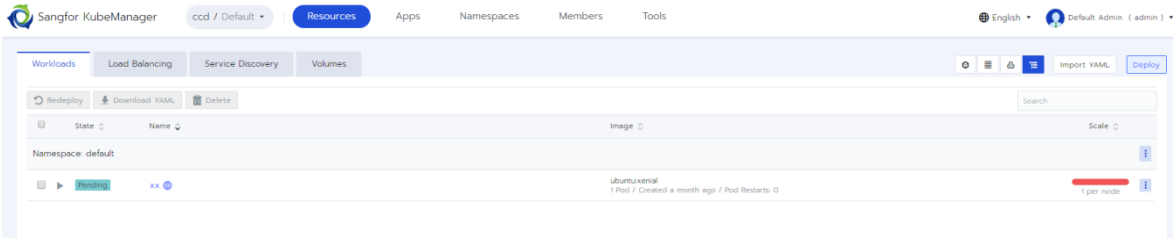
Step 0: Know the parameters to be customized or configured according to the image description document of the container application.

Step 1: Select the "Clusters" and "Projects" in the upper left corner of the interface. Click

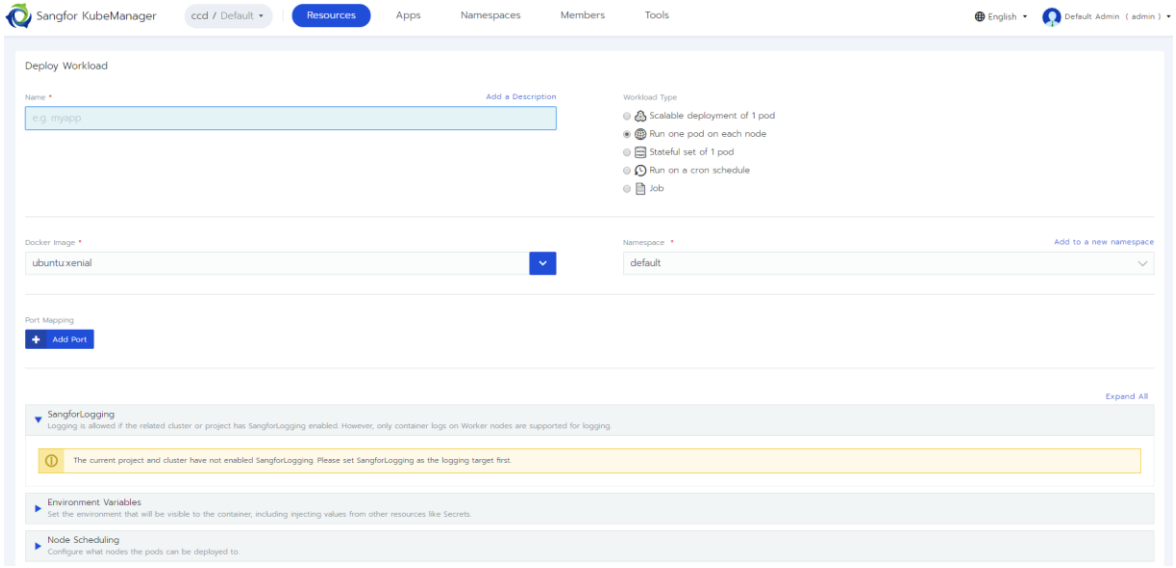
"Project". The "Workload" on the Resource interface is displayed.



Step 2: Click "Deploy".



3.2.3 Workload Configuration



Step 1: Configure the name.

Step 2: Add a short description of the configuration application (optional).

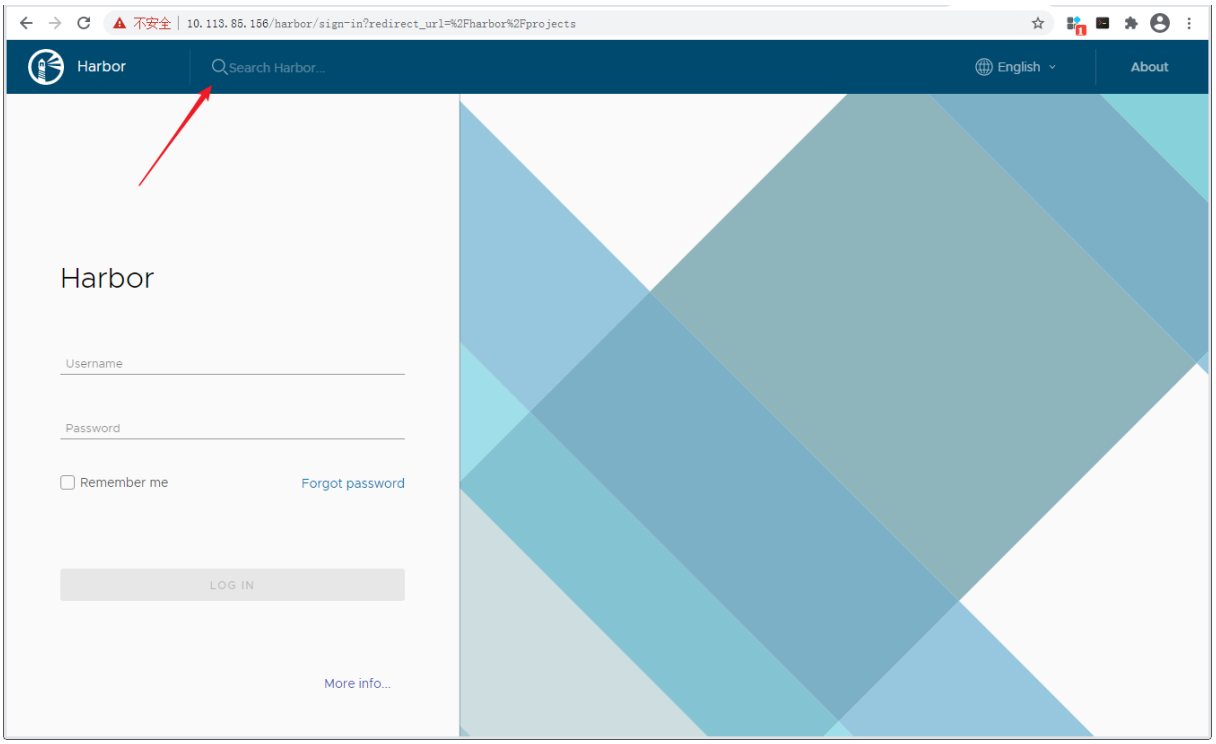
Step 3: Select and configure the K8s pod workload type and use deployment.

Step 4: Select the namespace in which the application will be deployed in the project. You may also create a new namespace. In K8s, the namespace is used for application isolation management.

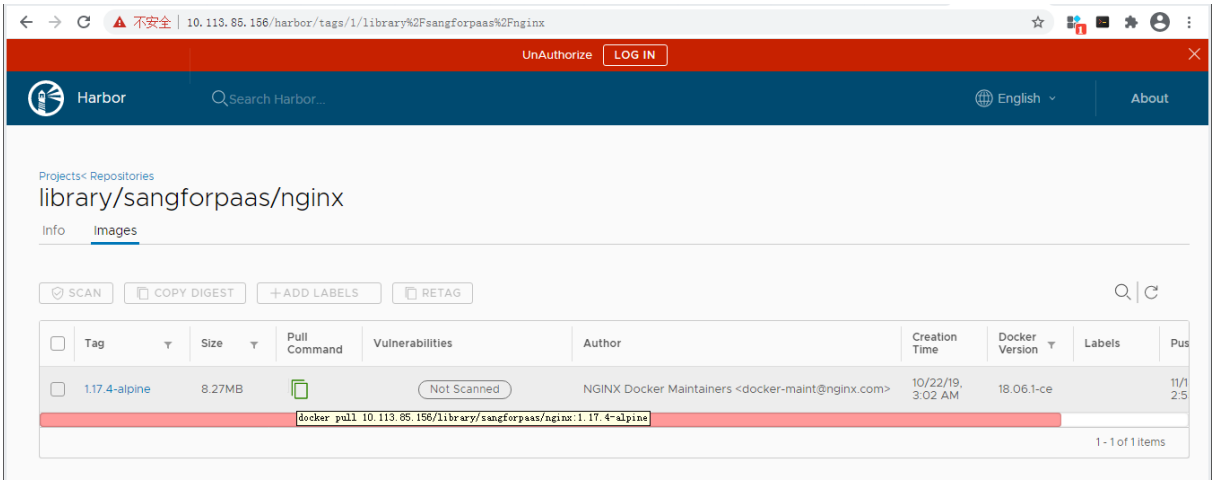
Step 5: Configure the image address of the application (10.113.85.156/library/logicaldoc-ce-Docker: latest) in "Docker Image".

How to obtain the image address?

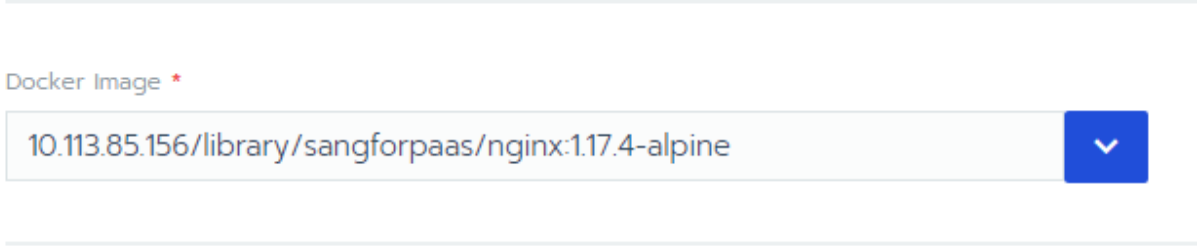
1. Enter HARBOR VIP in the browser and search for the image.



2. Copy the image address.

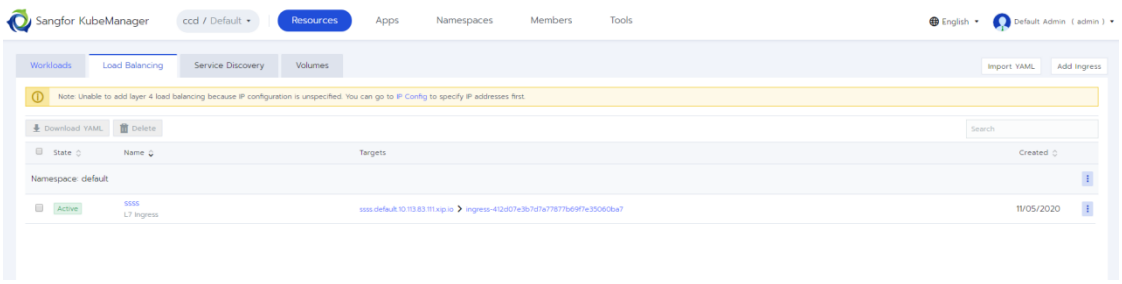


3. Paste the image address and configure the docker image.

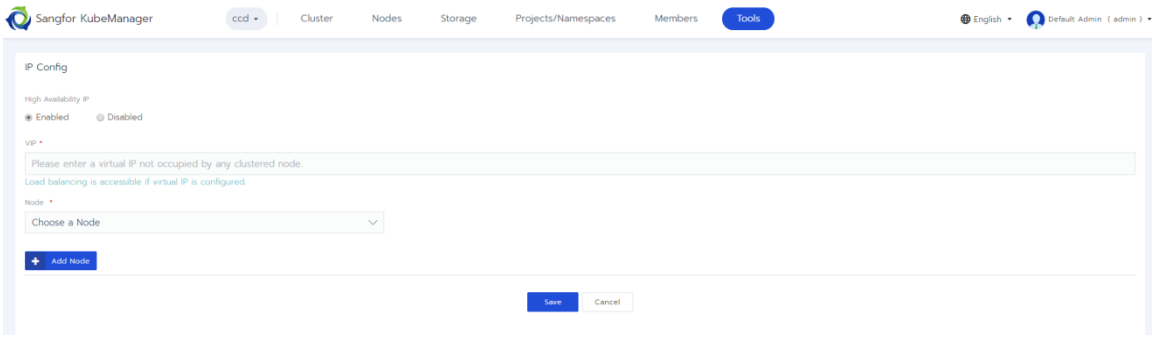


3.2.4 Network Port Settings

Step 1: Select the Resource interface of the project and click "Load Balancing". Currently, you cannot add L4 load balancing since you have not configured the network port. Click "IP Config" to configure it.



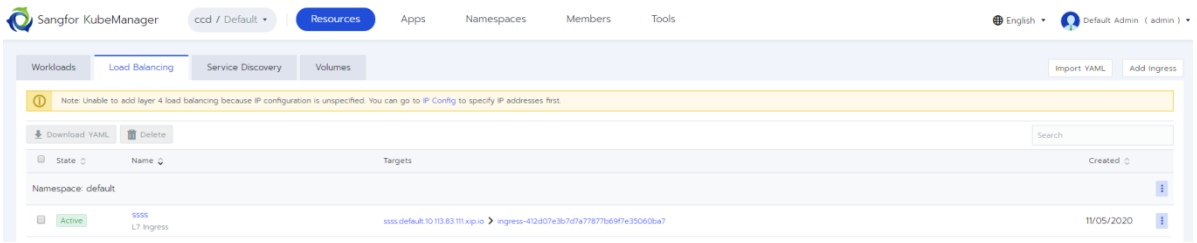
Step 2: On the "IP Config" interface, check "Enabled" for "High-Availability IP" and configure "VIP".



1. Configure the network port VIP.
2. Add the hosts and select two worker nodes. Click "Save".

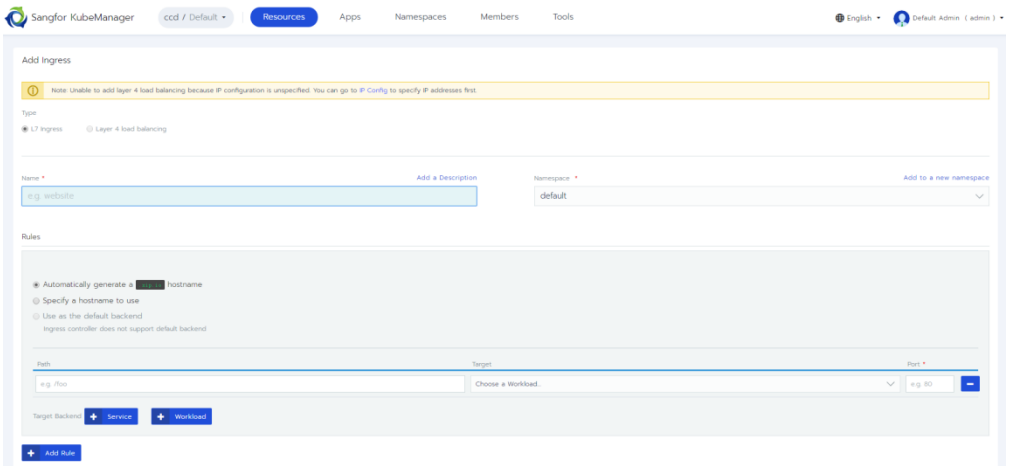
3.2.5 Publishing of L4 Services

Step 1: Select the Resource interface of the project. Click "Load Balancing" and click "Add Ingress".



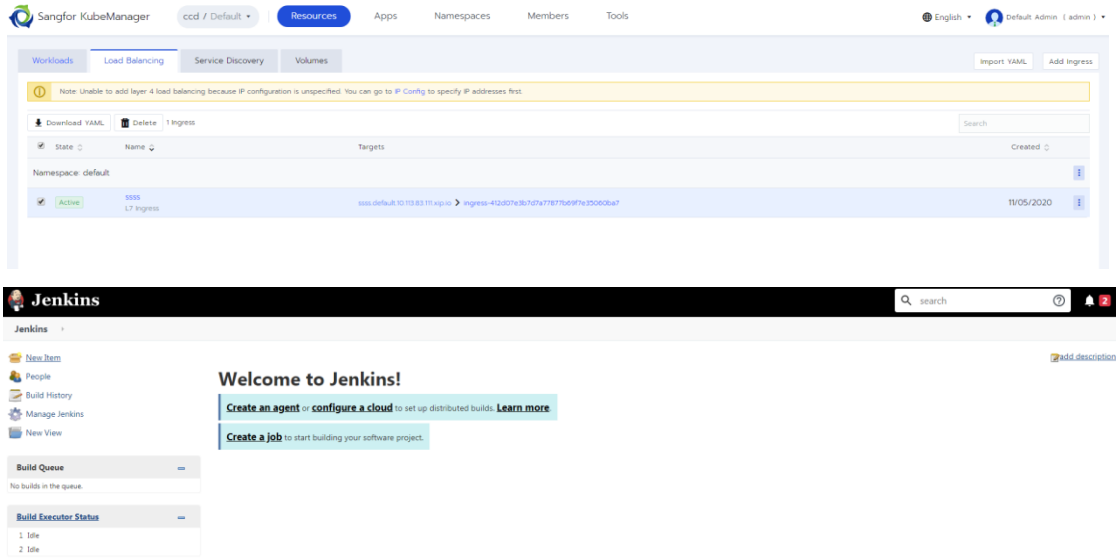
Step 2: Configure L4 load balancing.

According to the usage document of the container application, the container exposes port 8080. Therefore, we have the following configuration:



Configuration	Description
Name	Name of L4 load balancing
Type	Choose L4 type (TCP or UDP)
Monitoring Port	The port used to monitor the host
Service/Workload	If you select Workload, you need to manually configure the container port; if you select Service, just select a port.
Container Port	The port used to monitor actual container applications

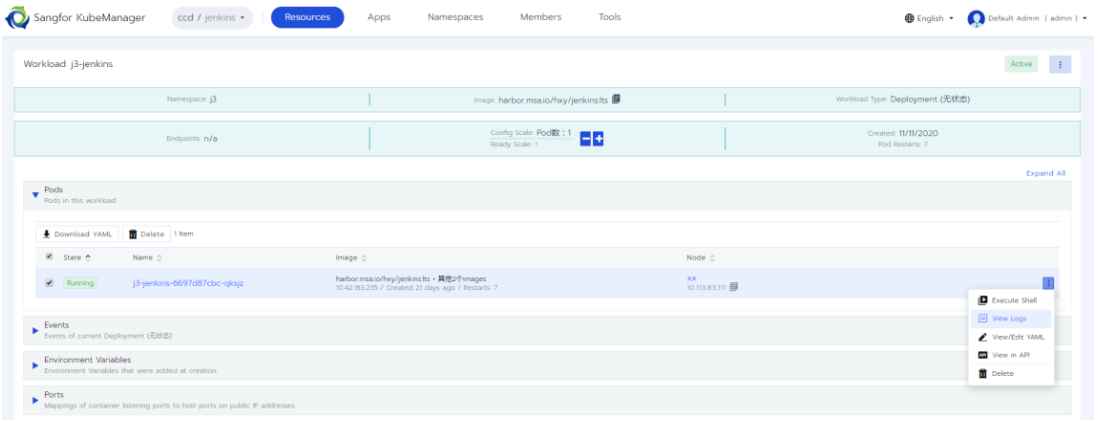
Step 3: On the Load Balancing interface, click the copy button of the target. Check whether the application can be accessed through the browser.

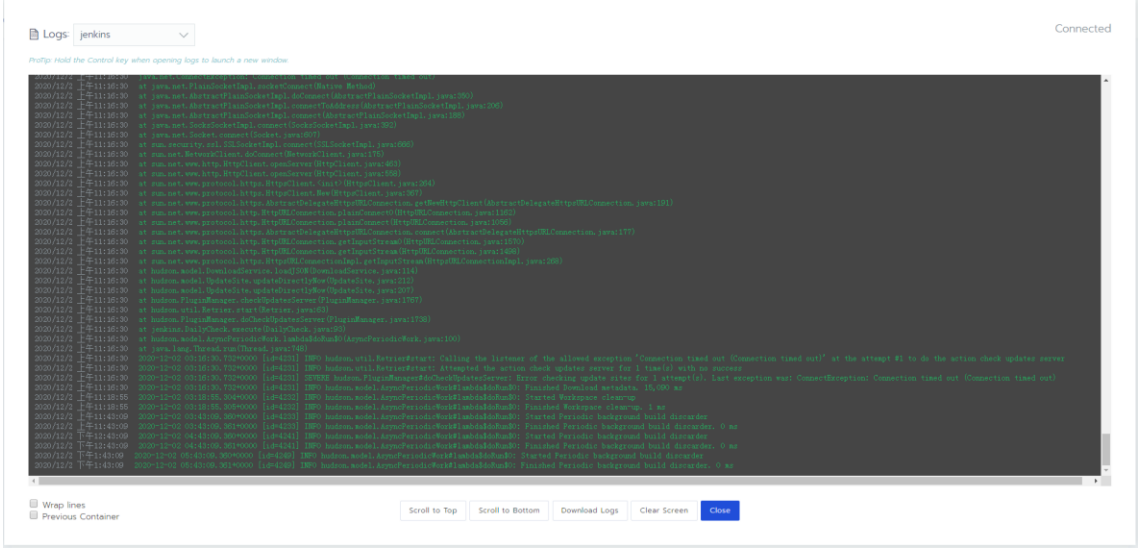


1. Click the copy button of the target.
2. Enter the access address in the browser. The container application can be accessed.

3.3 Verification

1. In the container application pod, click "More" and go to "View Logs". Check whether container application logs are generated and whether the application is normally started.





3.4 Questions for Discussion

When migrating container applications, think about:

1. In addition to the deployment of traditional container applications, what other benefits does KubeManager provides, compared with the direct operation mode of docker (in terms of monitoring, logs, O&M, distributed scheduling, and reliability)?
2. Docker applications use local storage. How is the storage used on K8s?
3. Docker's containers apply port mapping. How is K8s configured for service publishing?
4. How are the docker's environment variables configured on K8s?

4 Experiment 3: K8s Storage Configuration and Use

4.1 Introduction

4.1.1 About the Experiment

In this experiment, the iSCSI storage server is configured on aCloud for external servers to persist and store data using K8s. By creating a storage class in a K8s cluster, we can dock with the aSAN storage plug-in, thus facilitating the cluster's monitoring, logs, and applications using iSCSI virtual storage resources on aCloud through PV- or PVC-referenced storage class.

4.1.2 Purpose

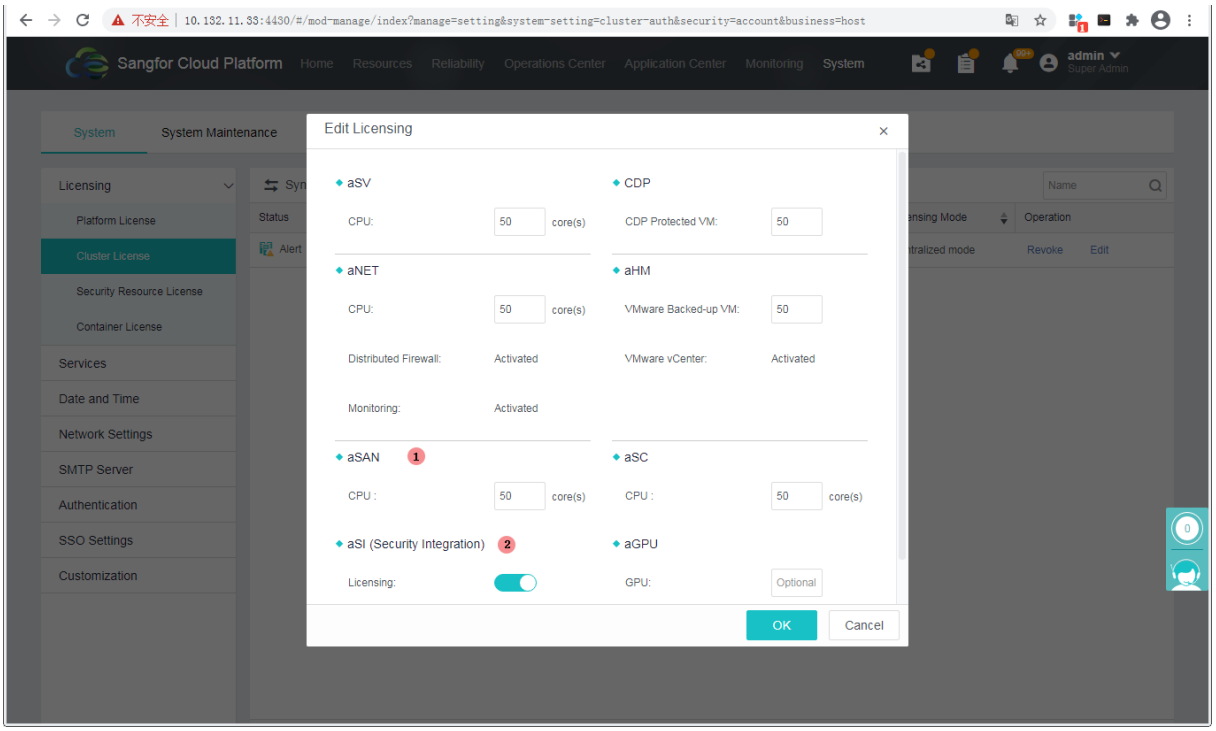
- To find out how to configure the iSCSI storage server on aCloud, and how to configure storage for and use K8s clusters through KubeManager
- To know the PV, PVC, and SC concepts of K8s[8]
- To understand that the monitoring, log, and load balancing of K8s clusters can realize data persistence with storage classes
- To know the steps of configuring Sangfor aSAN storage:
    1. Check the storage and security correlation authorization of aCloud on SCP.
    2. Enable port 4433 on the aCloud cluster and configure the PaaS correlation account.
    3. Configure the virtual iSCSI server in the aCloud cluster.
    4. Configure a storage server on KubeManager.
    5. Configure a storage class on KubeManager.
    6. In the Workload page, configure to have the PVC correlated with a storage class, in order to use the storage.

4.2 Steps

4.2.1 Checking Infrastructure License

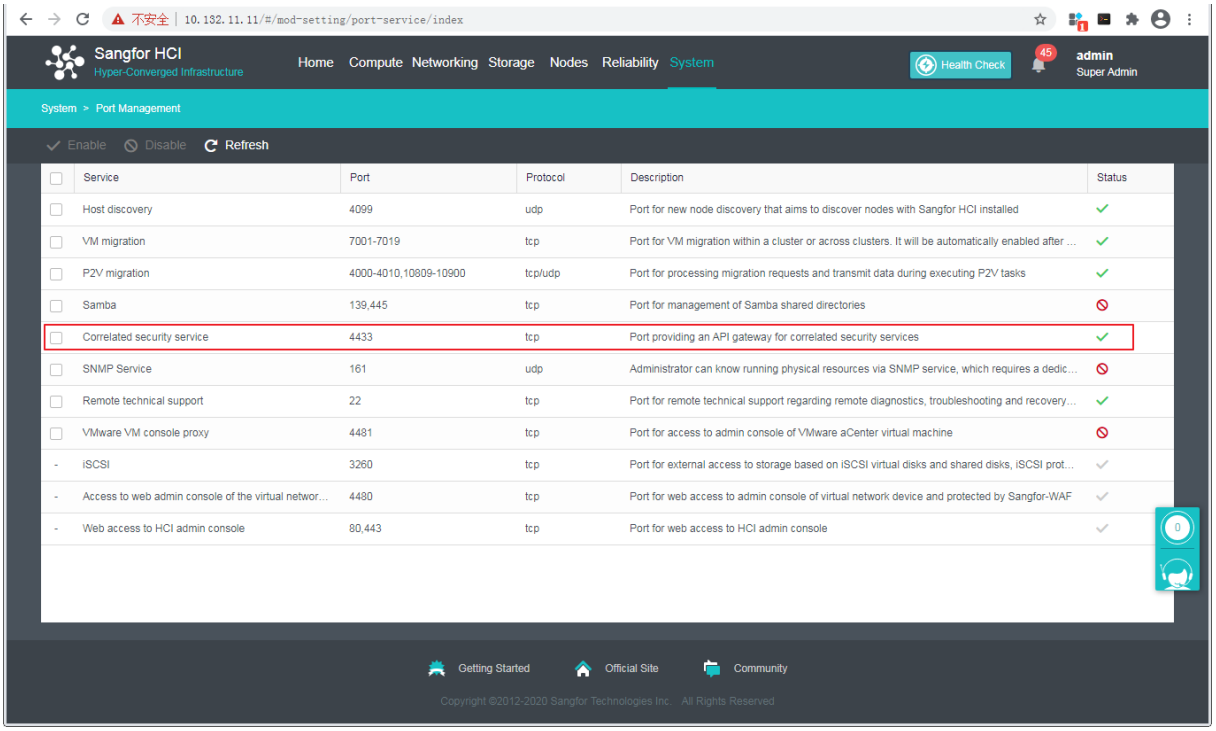
Step 1: On the "Serial Number" page of the SCP management interface, click "Cluster License" to edit the license of the cluster. Make sure that both the "Storage Virtualization (aSAN)" and "Security Correlation (aSI)" are configured.

[8]For more details, see Persistent Volumes, Storage Classes, and Dynamic Volume Provisioning on the official site.



4.2.2 aCloud Cluster Storage Configuration

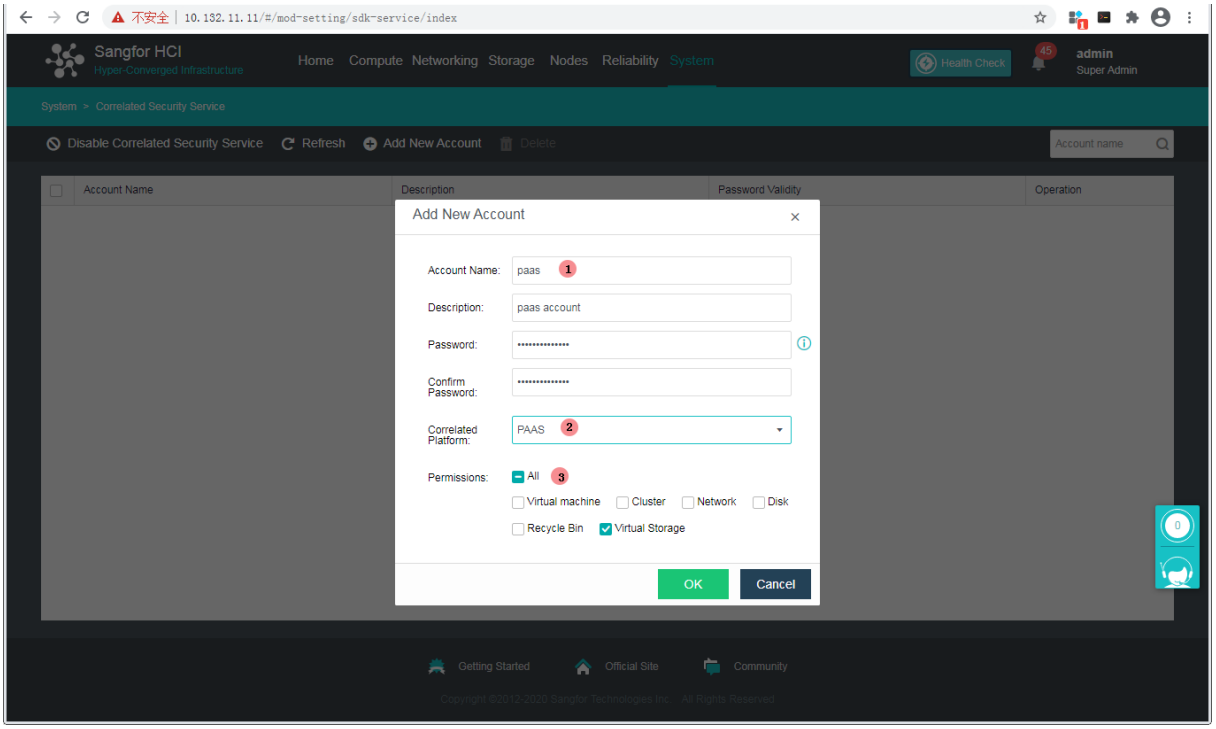
Step 1: In the SCP cluster, log in to the correct aCloud cluster. On the system management interface, click "Port Management" and enable port 4433.



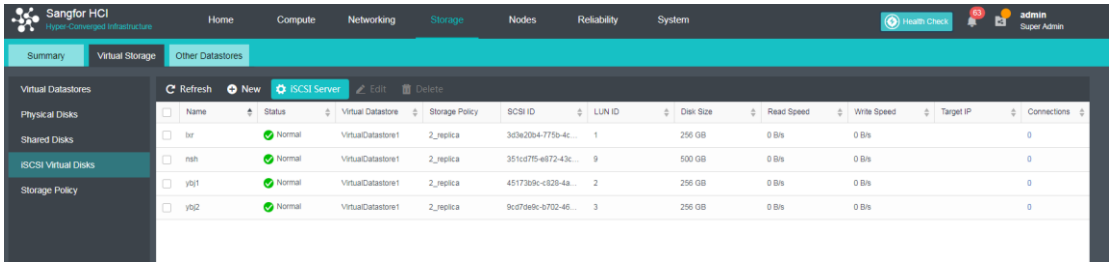
Step 2: In the aCloud cluster, click "Add Service Account" on the system management interface. Create a PaaS-correlated account and take down the username and password. You will configure the "storage server" for the K8s cluster using this account.

1. Account name
2. Choose **PAAS** for "Correlated product"

3. Only check **Storage** for "Enable services"

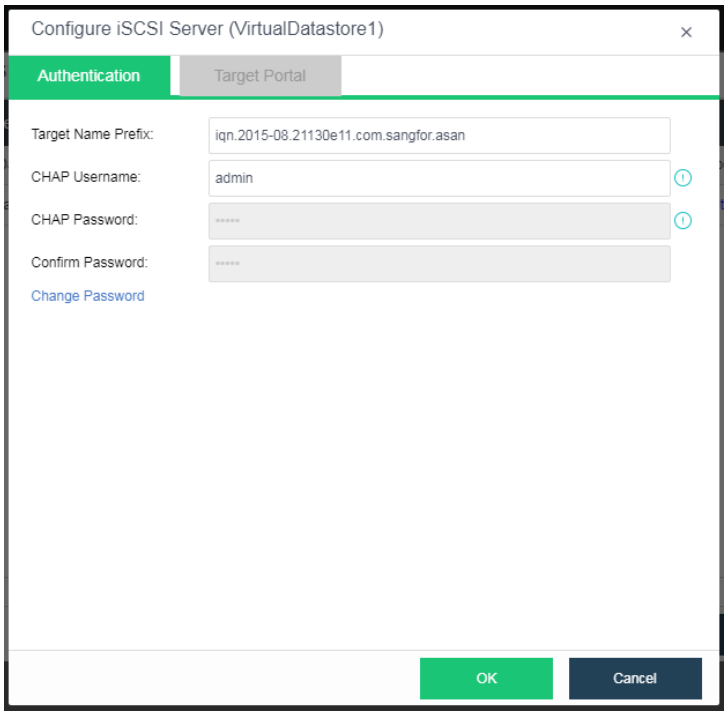


Step 3: In the aCloud cluster, on the Virtual Storage interface, click "iSCSI Virtual Disks" -> "iSCSI Server".



Step 4: Configure iSCSI authentication and the access IP address, and take down the following information:

1. Configure iSCSI Auth
2. Configure Access IP

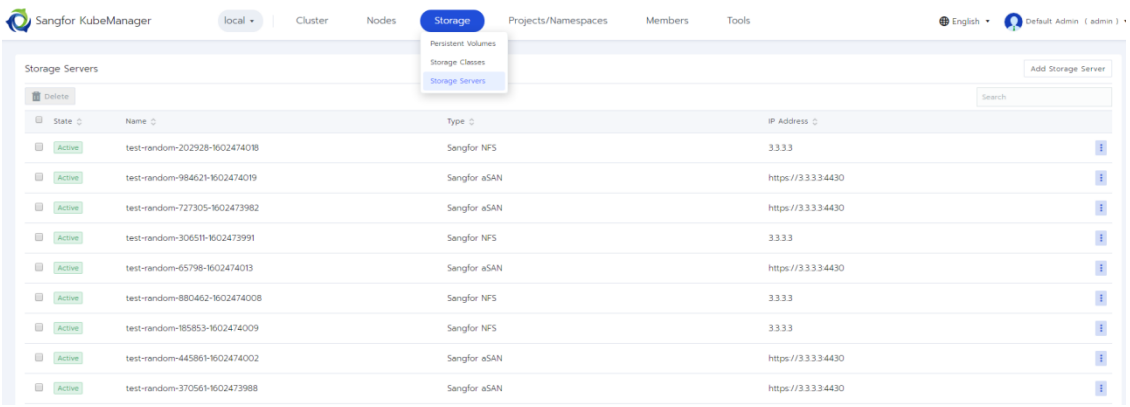


Config Item	Configuration	Description
Name of storage volume	Virtual Storage Volume 1	Name of the virtual storage volume, used to configure aSAN storage class
Target Prefix	-	"iqn", used to configure aSAN storage class
CHAP Username	admin	Username of iSCSI CHAP, used to configure aSAN storage class
CHAP Password	-	Password of iSCSI CHAP, used to configure aSAN storage class
External Network Port	-	Network port for iSCSI storage traffic, configured as management network or storage network (recommended)
Access IP	10.113.66.171	A K8s node accesses virtual iSCSI via the access IP address. Please ensure that the cluster node and the access IP[9] network are available for configuring the aSAN storage
Subnet Mask	-	Subnet mask of the access IP address
Virtual IP Pool	-	The system allocates a virtual IP address in the virtual IP pool for each host. After accessing via the access IP address, all external hosts will be evenly redirected to the respective virtual IP addresses of different hosts.

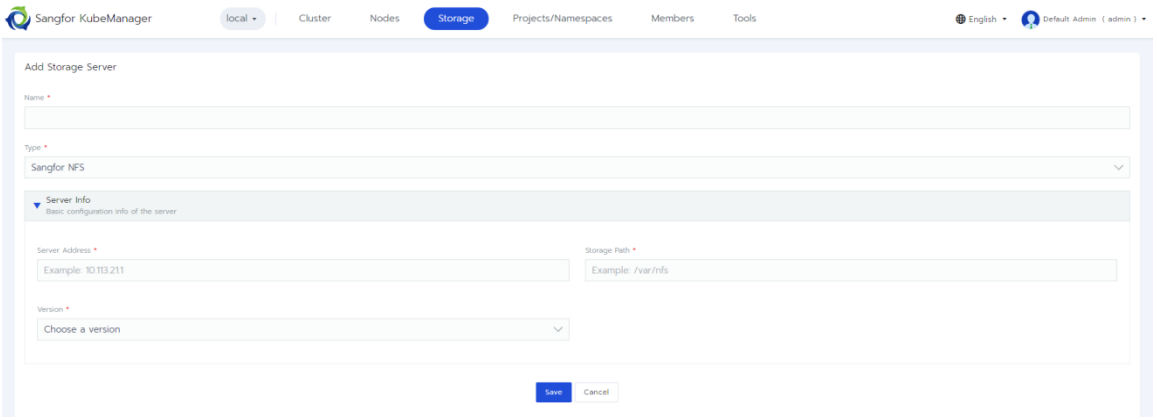
4.2.3 Configuration of K8s Cluster Storage Server

Step 1: Select the cluster and then click the "Storage Server" under "Storage" to go to the interface of the storage server list.

[9]The storage network is recommended for the production environment. The K8s cluster node creates a NIC separately for storage traffics and configures a storage IP address for communication with the access IP address.



Step 2: Click "Add Storage Server" and then configure the storage server according to Adding Service Account to aCloud Cluster in the previous section.

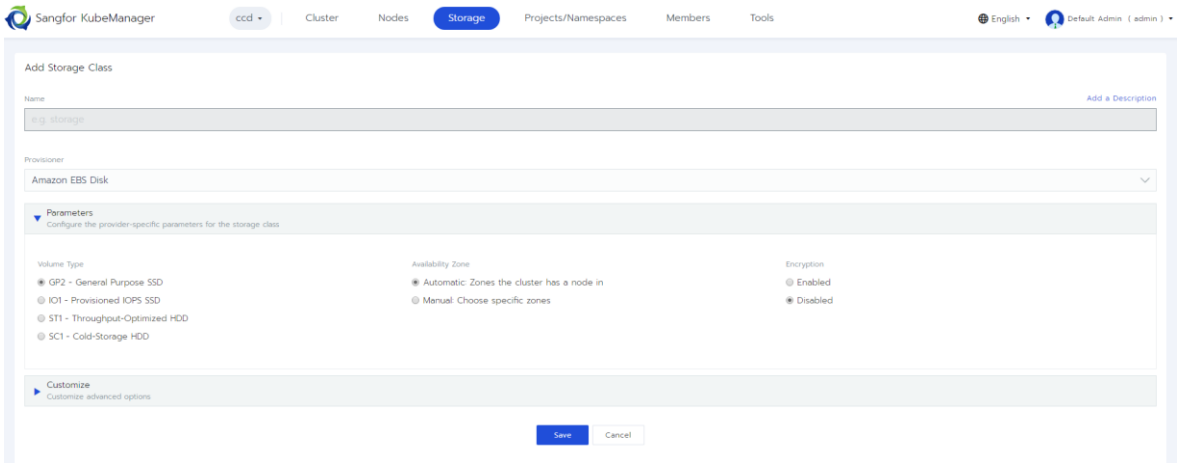


Config Item	Configuration	Description
Name	aSAN1	Configure the name of the storage server
Type	Sangfor aSAN	Select the type of storage service provider
Gateway IP	https://10.113.66.11:4433	The aCloud cluster IP address and the 4433 port
Authenticated User	paas	Partner service, PaaS correlation account
User Auth Password	-	Partner service, PaaS correlation password

4.2.4 Configuration of K8s Cluster Storage Class

Step 1: Select the cluster and then click the "Storage Class" under "Storage" to go to the interface of the storage class list.

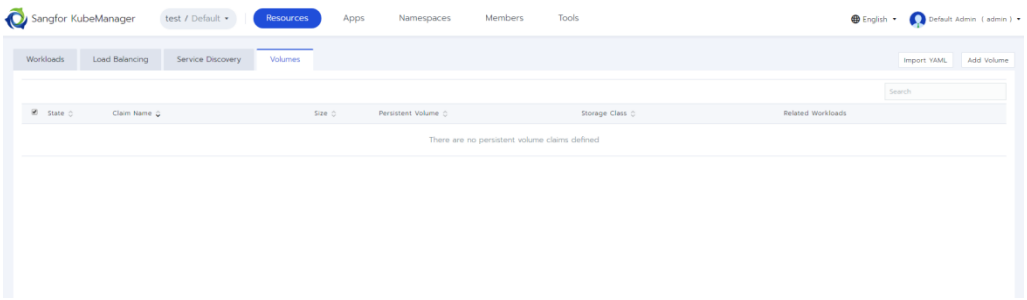
Step 2: Click "Add Storage Class" and then add a storage class according to Configuring aCloud Virtual iSCSI Server in the previous section.



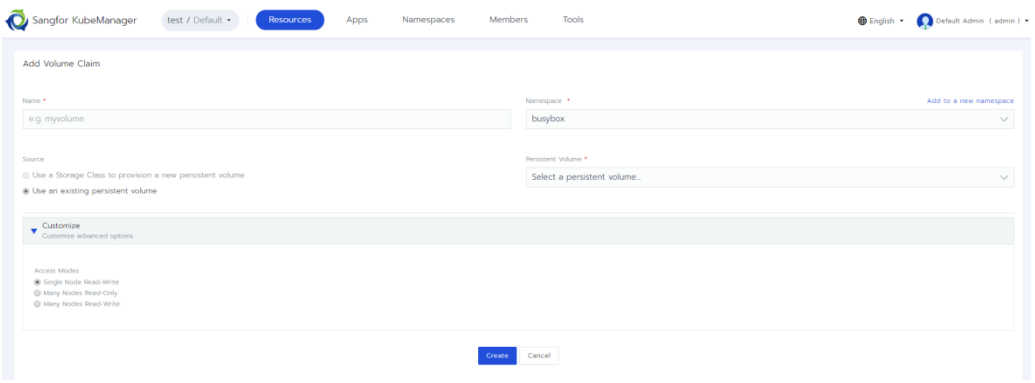
Config Item	Configuration	Description
Name	asan-sc1	Configure the storage class name
Description	-	A brief description of the storage class
Provider	Sangfor aSAN	Select the storage plug-in
Storage Server	aSAN1	Select the storage server configured in the previous section
Name of storage volume	Virtual Storage Volume 1	See the storage volume name in Configuring iSCSI Server
Access IP	10.113.66.171:3260	See the access IP address in Configuring iSCSI Server
IQN	-	See the Target prefix in Configuring iSCSI Server
File System Type	ext4	Currently, only ext4 is available
Storage Policy Name	-	Optional; can be left blank
Storage Space Pre-Allocation	Enabled	Pre-allocation will occupy more storage space while promoting the performance of the hard disk; if this item is disabled, space will be occupied based on the actual data. That is, space will be allocated in an on-demand manner.
Accessible Hosts	Allow all hosts to access	Configure the K8s hosts that are allowed to access the storage class
CHAP Auth Username	admin	See the CHAP user in Configuring iSCSI Server
CHAP Auth Password	-	See the CHAP password in Configuring iSCSI Server
Deallocation Policy	-	Deallocation policy of the K8s storage class
Mounting Options	-	Mounting options supported by the **mount** command

4.2.5 Use of K8s Application Storage

Step 1: Select "Clusters" and "Projects" in the upper left corner of the interface. Click "Project" to go to the Volumes page on the Resources interface and then click "Add Volume".

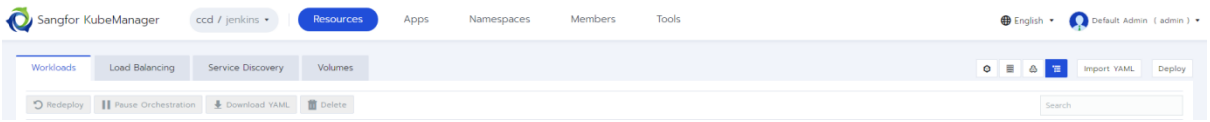


Step 2: Click "Create" to add a PVC.

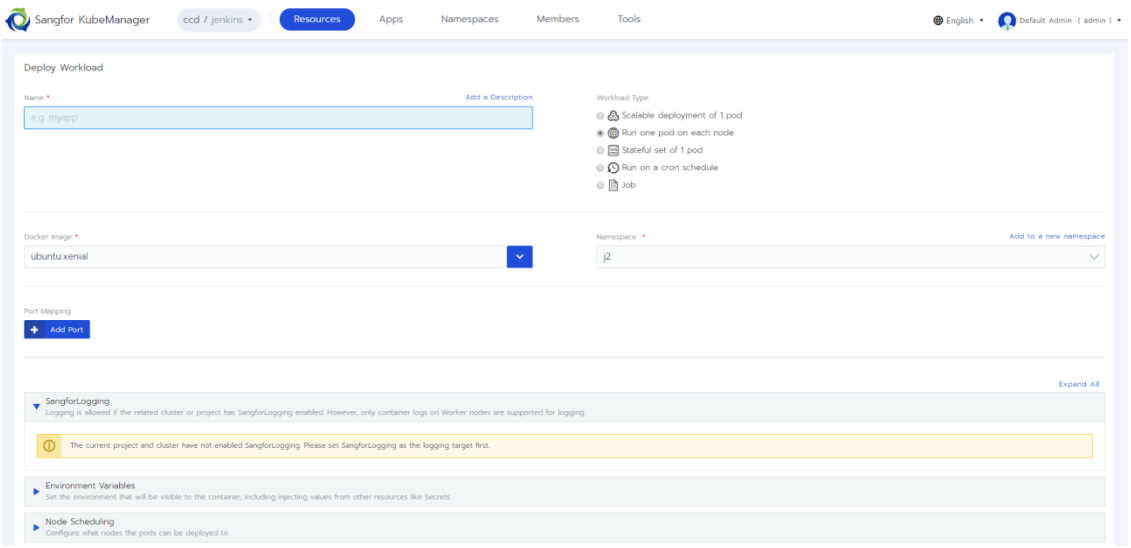


1. Configure PVC name
2. Select the namespace to create the PVC
3. Select the storage class used by the PVC
4. Configure the storage capacity of the PVC

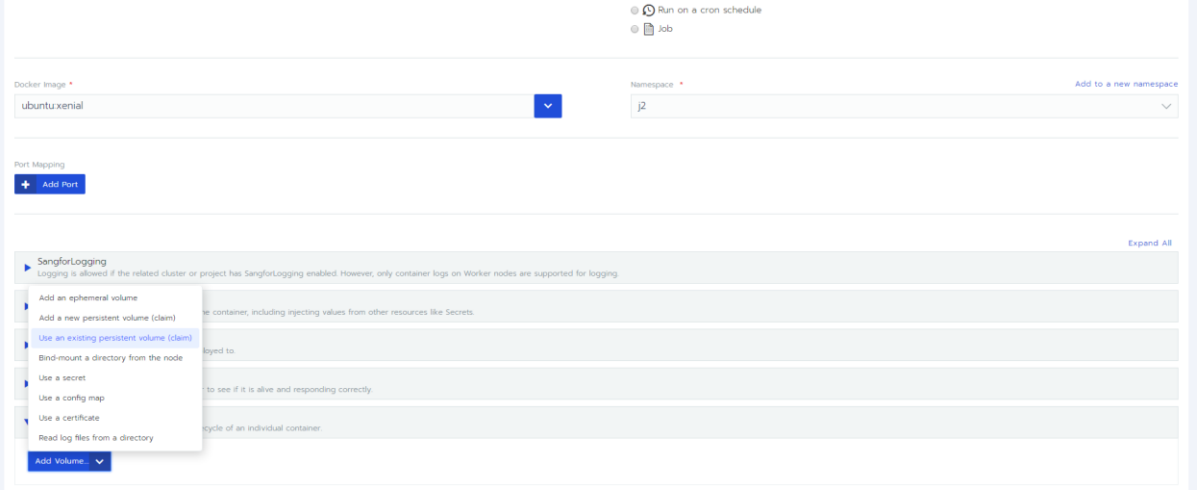
Step 3: Select "Clusters" and "Projects" in the upper left corner of the interface. Click "Project" to go to the Workloads page on the Resources interface and then click "Deploy".



Step 4: Basic workload configuration



1. Configure the name.
2. Select to configure the workload type of the K8s pod, i.e., pod controller type.
3. Docker Image: Configure the image address of the application, which can be searched in the Harbor VIP built-in registry.
4. Select the namespace of the project where the application will be deployed. You may also create a new namespace.
5. Configure the data volume



(a) Click "Add Volume" and select "Use an existing persistent volume(claim)".

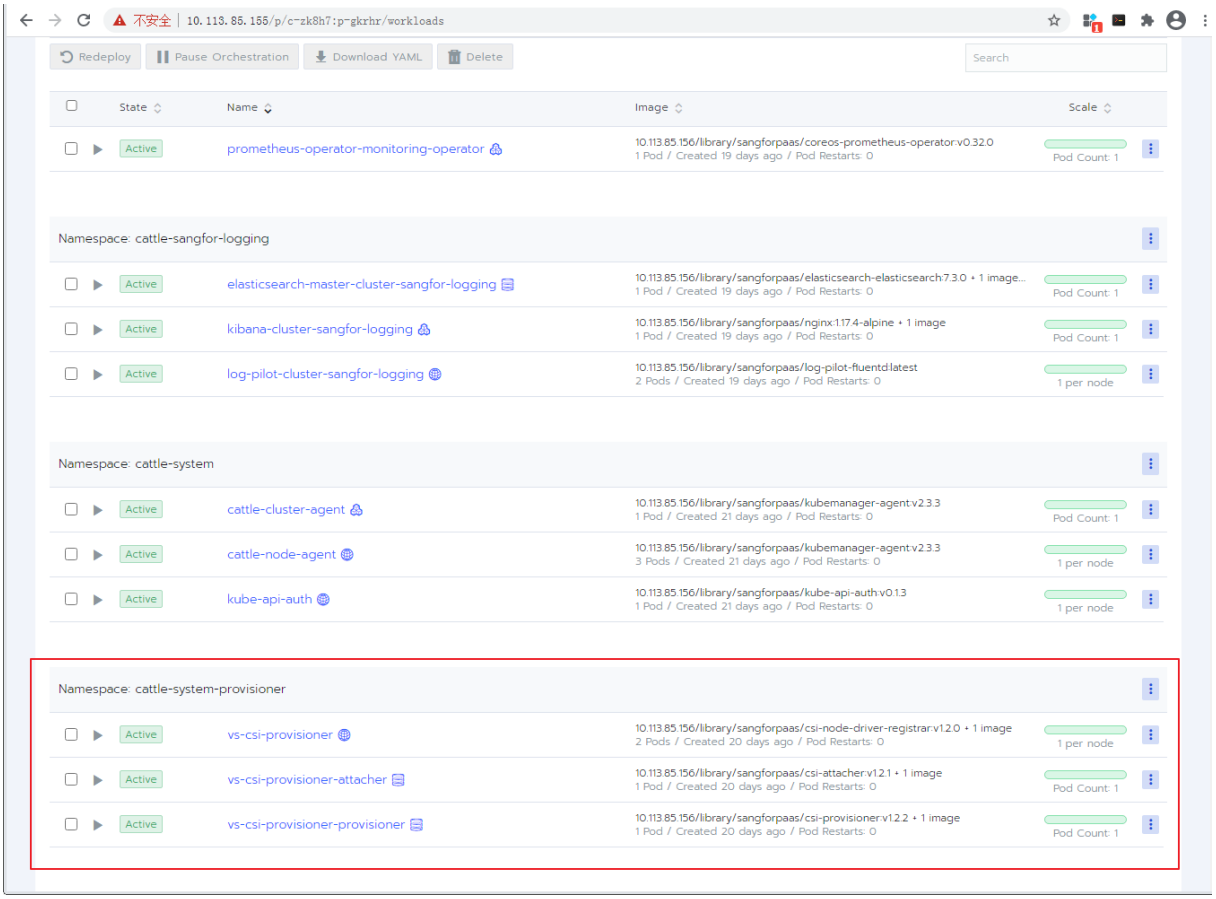
(b) Configure the volume name.

(c) Select the PVC created in Step 2.

(d) Configure the path for mounting the container.

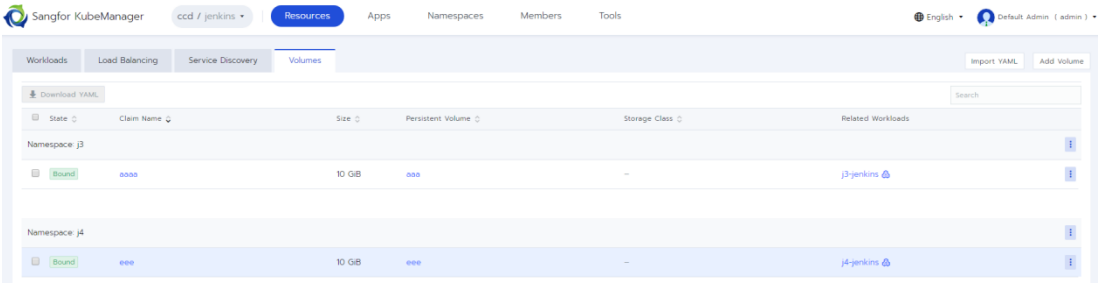
4.3 Verification

Step 1: On the interface, select "Cluster" and "System Project". On the "Resource" -> "Workload" interface, check whether the workload of namespace **cattle-system-provisioner** is working normally.

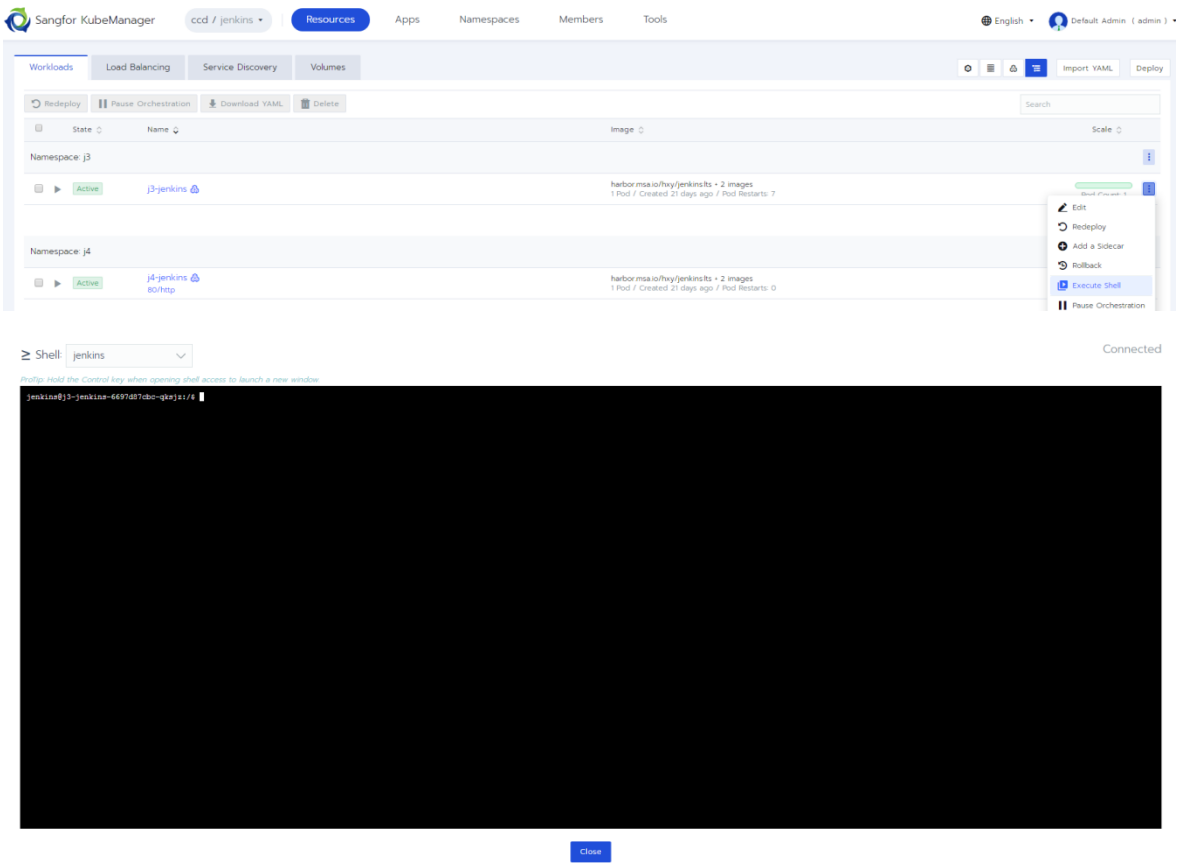


Step 2: Check PVC. By creating a PVC, the Sangfor aSAN storage plug-in calls the storage

volume to create a PV starting with "pvc" for dynamic provisioning.



Step 3: Click "Workloads" -> "Execute Shell" and run the **df** command, to check whether the workload has correctly mounted the storage to the container directory.



As shown in the figure, by running the **df** command, you can see the storage configured through the data volume. The storage plug-in automatically creates an iSCSI disk on the aCloud platform, formats the disk, and then mounts it into the K8s node. Finally, the storage plug-in mounts the disk into container path **/mydata** through a bind mount.

4.4 Questions for Discussion

When using the K8s cluster storage, think about:

1. What other storage plug-ins are supported, in addition to Sangfor aSAN?
2. The K8s applies container technology and the container only adopts local bind mounts to map and use directories. What is responsible for creating volumes when a storage plug-in is being used? Is the disk attached, formatted, mounted, and then bind-mounted to the container?
3. The workload of Sangfor aSAN in namespace **cattle-system-provisioner** in a cluster

is normally deployed but the related PVC for storage remains pending. Which storage class configuration and docking configuration items of Sangfor aSAN are abnormal?

4. PVs and SCs[10] are cluster resources, whereas PVCs are namespace resources. How does a PVC reference an SC to automatically create a PV and then use the storage?
5. Is the SC editable? If a PVC or PV reference an SC, can the SC be deleted?
6. Why is the option for setting the default storage class available for clusters?

[10] SC (storageClass) stands for storage class and PV for persistent volume.

5 Experiment 4: Enabling K8s Cluster Monitoring

5.1 Introduction

5.1.1 About the Experiment

This experiment enables monitoring in a K8s cluster to monitor and display real-time indicators of the cluster. Users can view real-time monitoring data and event messages from the cluster perspective.

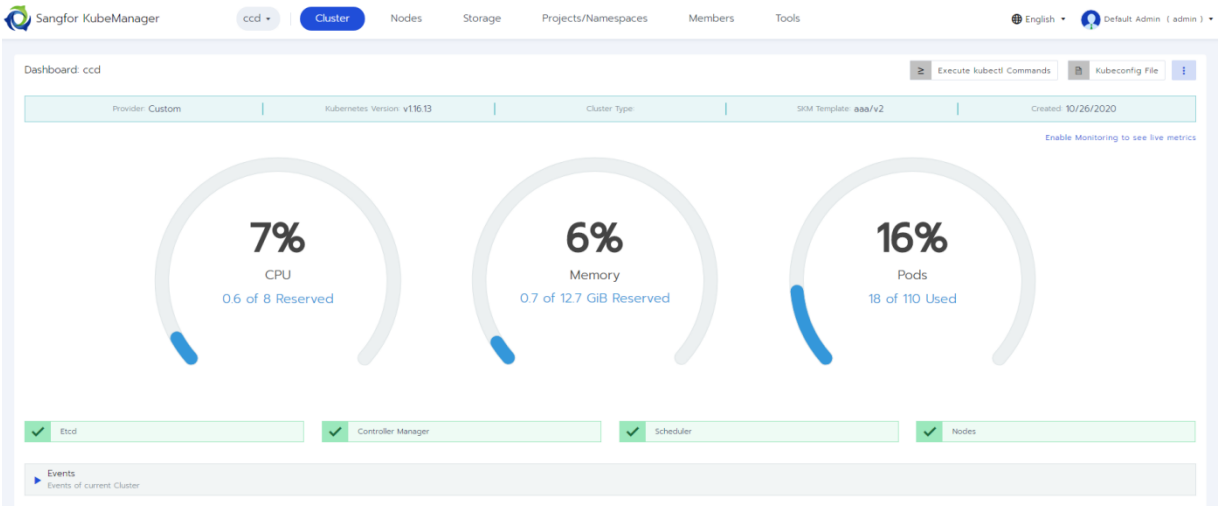
5.1.2 Purpose

- To find out how to view monitoring indicators of a cluster and workloads, which will provide a basis for procedure optimization and be used for analyzing real-time service workloads.

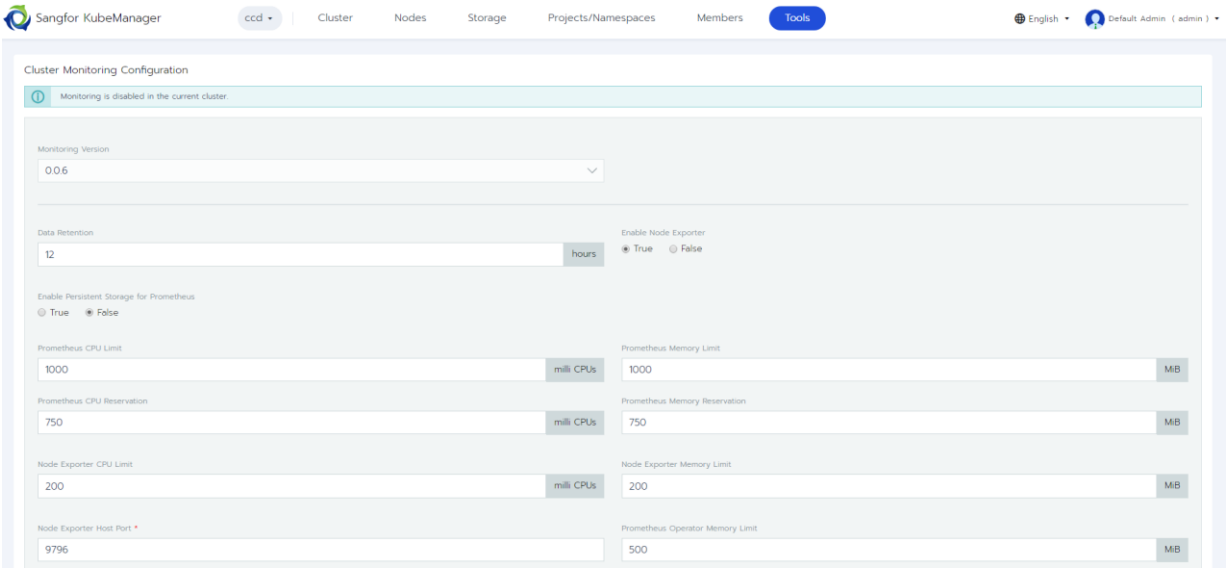
5.2 Steps

5.2.1 Enabling Cluster Monitoring

Step 1: On the cluster interface, click "Cluster Name" to go to the cluster dashboard. Click "Enable Monitoring to see live metrics" to go to the "Cluster Monitoring Configuration" interface.



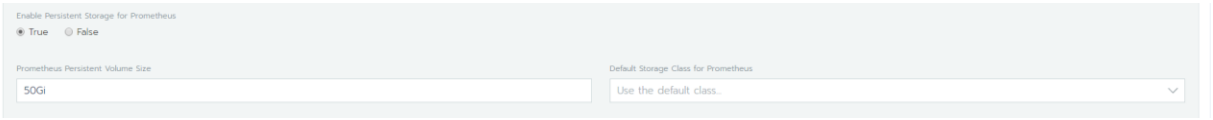
Step 2: Cluster Monitoring Configuration



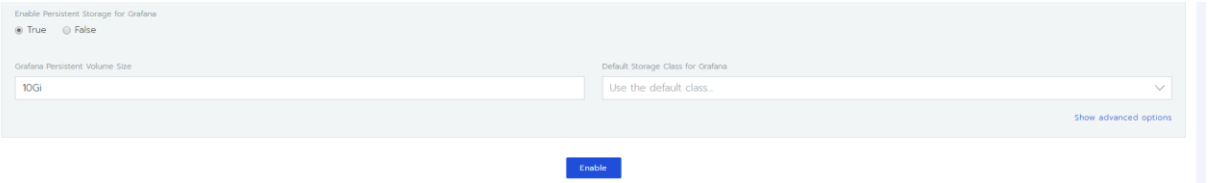
Config Item	Configuration	Description
Monitoring Component Version	0.0.6	Monitoring Component Version
Data Storage Duration	12 hours	Duration for monitoring data persistence, which can be set based on the compliance requirement
Enable Node Exporter	Yes	Whether to enable Node Exporter
Enable Persistent Storage for Prometheus	Yes	Whether to persist monitoring data to prevent data loss in case the monitoring pod crashed
Prometheus CPU Limit	1000m	CPU limit; more resources will be allocated if there are more carried services, to ensure stable operation
Prometheus CPU Preservation	750m	CPU request related to scheduling
Prometheus Mem Limit	1000M	mem limit
Prometheus Mem Preservation	750M	mem request
Node Exporter CPU Limit	200m	CPU limit
Node Exporter Mem Limit	200M	mem limit
Node Exporter Host Port	9796	Node Exporter Host Port
Prometheus Operator Mem Limit	500M	mem limit
Add Selector	-	
Add Scheduling Tolerance	-	
Enable Persistent Storage for Grafana	Yes	

5.2.2 Cluster Monitoring Data Persistence

1. Enable persistent storage for Prometheus

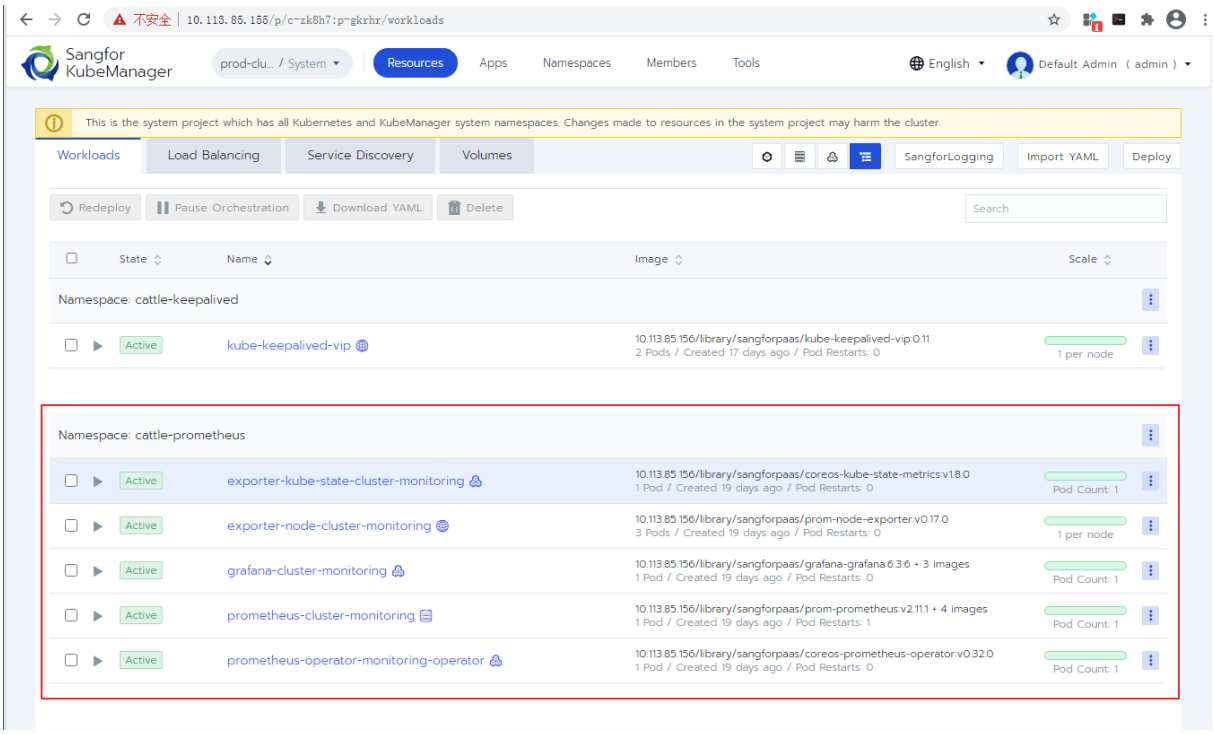


2. Enable persistent storage for Grafana

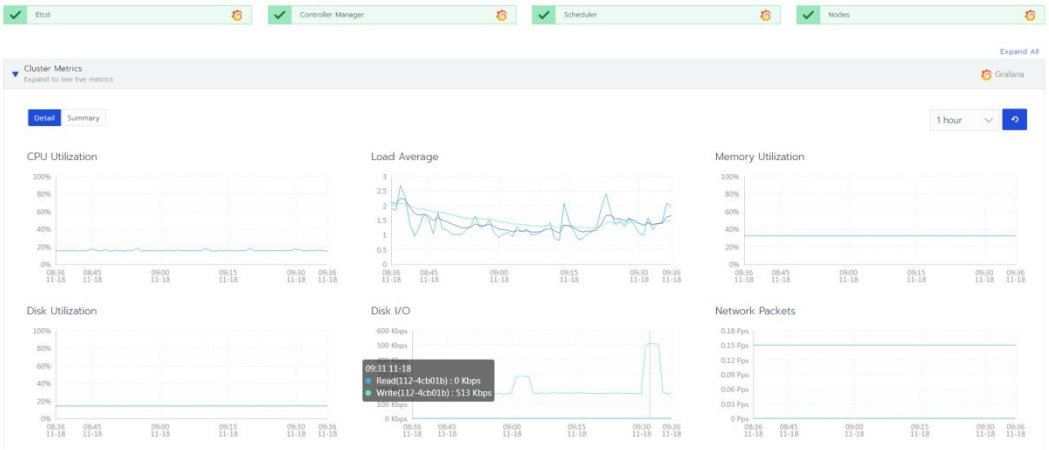


5.3 Verification

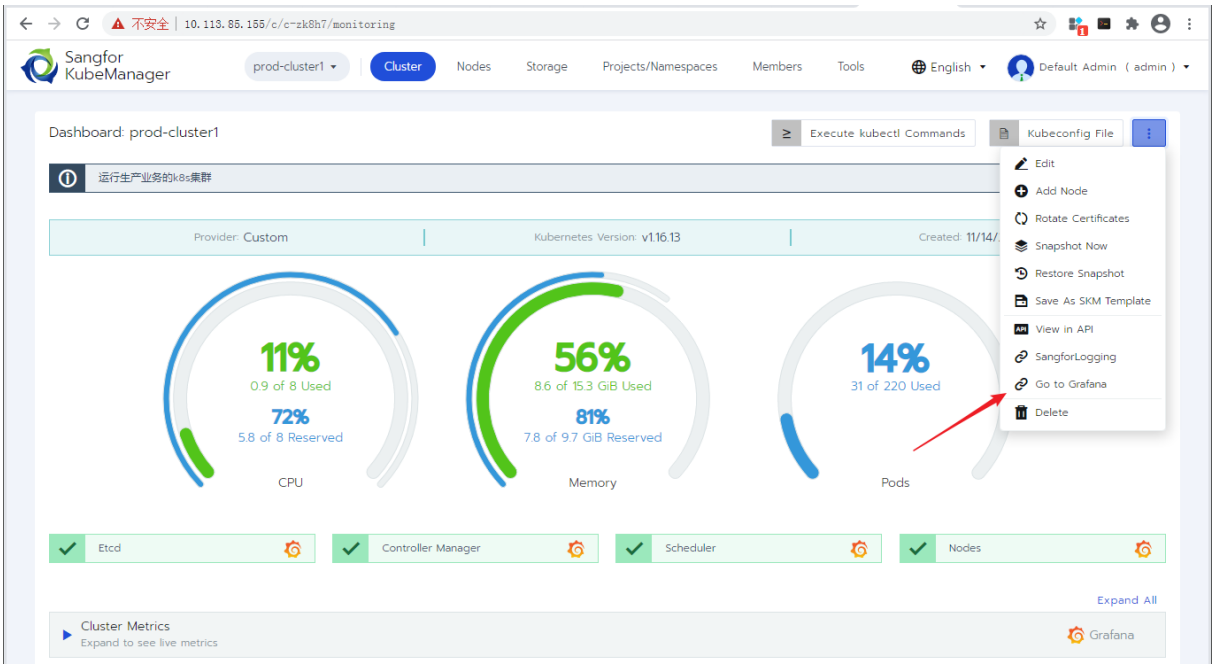
Step 1: On the interface, select "Cluster" and "System Project". On the "Resource" -> "Workload" interface, check whether the workload of namespace **cattle-prometheus** is working normally.



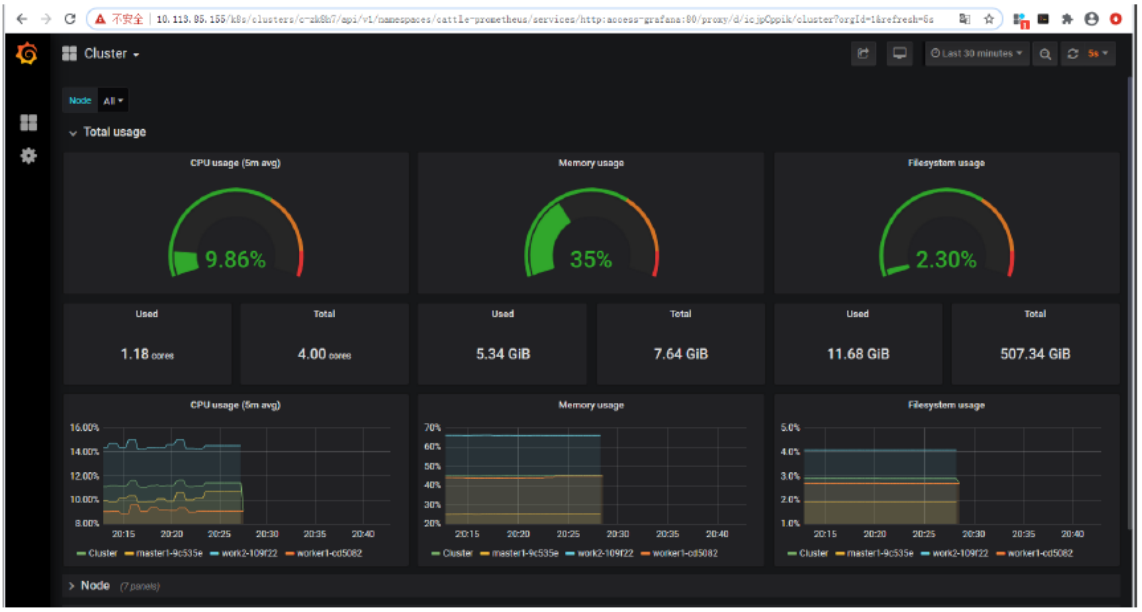
Step 2: After enabling monitoring, wait for 2–3 minutes and then click "Cluster Name" on the cluster interface to go to the cluster dashboard. On the dashboard, check whether there is real-time monitoring data.



Step 3: On the cluster interface, click "Cluster Name" to go to the cluster dashboard. Then click "More" on the upper right corner of the interface and see if there is "View Grafana" from which you can enter the Grafana Monitoring interface.



Step 4: Go to the Grafana interface to view the cluster's real-time indicators. Set the refresh time to 5 s and check whether the monitoring data is updated.



5.4 Questions for Discussion

When enabling the cluster monitoring, think about:

1. What other things need to be monitored, in addition to the cluster? What other workloads need to be monitored, in addition to the K8s cluster's basic components and VMs?
2. Are service monitoring and health inspection of workloads related to monitoring? How does K8s realize service monitoring?
3. Is it feasible to not configure storage for cluster monitoring? What are the risks of monitoring non-persistent storage in the production environment?
4. What can you do if monitoring data is available for workloads? HPA?

6 Experiment 5: Enabling K8s Cluster Logs

6.1 Introduction

6.1.1 About the Experiment

With cluster logs enabled, Load Balancing deployment can collect standard logs or file path logs, implement persistent data storage, and troubleshoot and analyze problems by log retrieval on the interface.

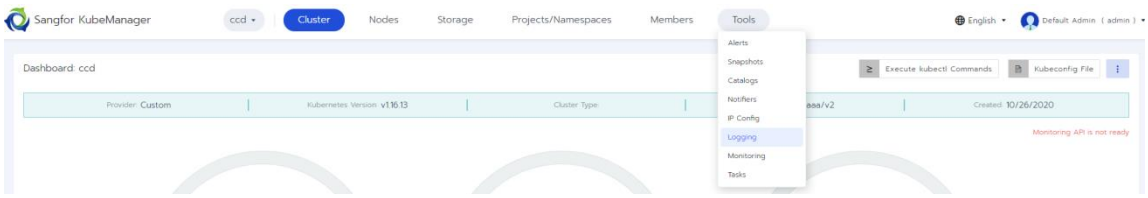
6.1.2 Purpose

- To find out the methods of collection, persistence configuration, and retrieval of workload logs

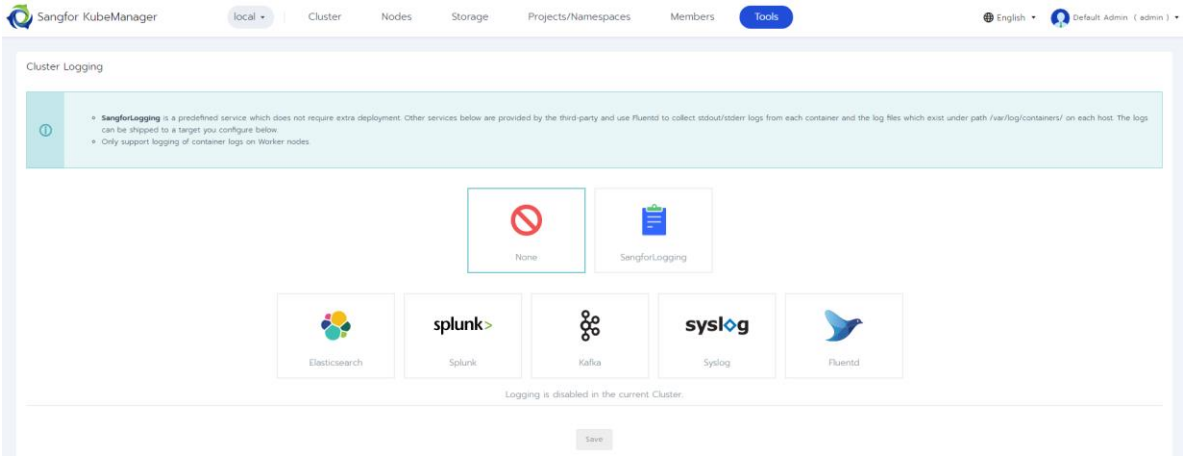
6.2 Steps

6.2.1 Enabling Cluster Logs

Step 1: On the cluster interface, click "Cluster Name", move the mouse cursor to "Tools" and then click "Logging" to go to the Cluster Log Collection interface.



Step 2: Click "SangforLogging", configure log collection, and enable persistent storage. Retrieval and troubleshooting functions are available on the interface.



Config Item	Configuration	Description
Log Component Version	7.3.0	Log Component Version
Kibana Language	Simplified Chinese	Default language on the log inspection interface
Enable Shared Storage for SangforLogging	Yes	To apply centralized storage for log persistence. If you select "No", a directory will be created locally and the log data saved on the application layer is highly available.
CPU Limit	2000m	CPU limit
CPU Preservation	1000m	CPU request
Mem Limit	3000M	Mem limit

Mem Preservation	2000M[11]	Mem request

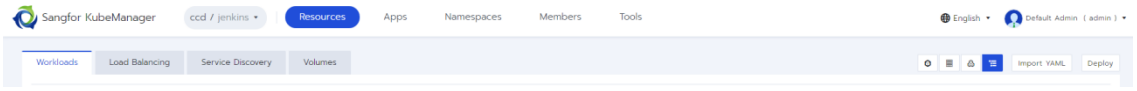
6.2.2 Cluster Log Data Persistence

1. Enable shared storage for SangforLogging

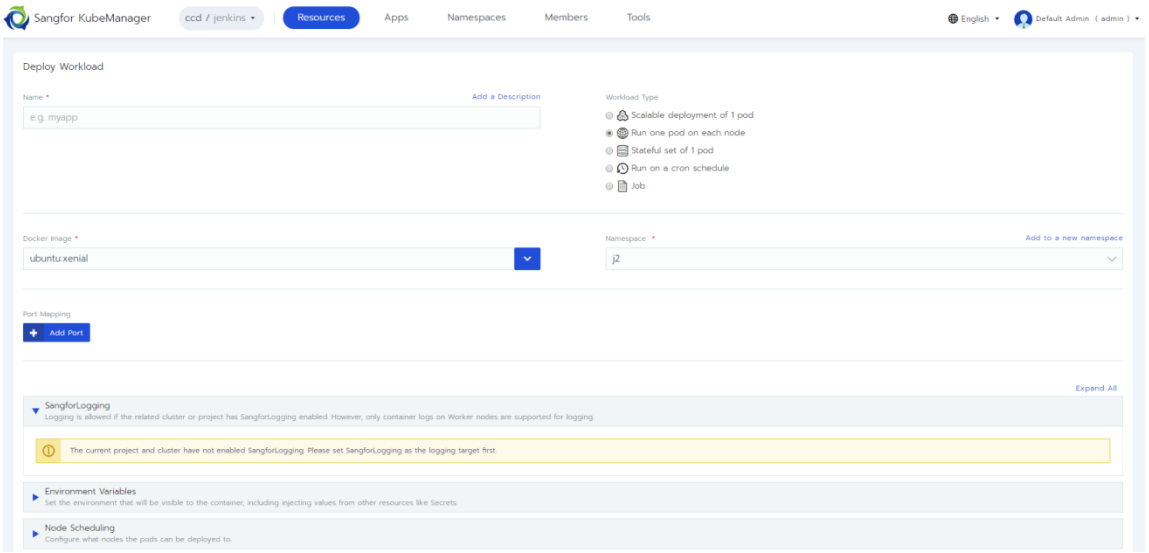
By default, if the shared storage is not enabled for SangforLogging logs, when there are three or more nodes in a cluster, SangforLogging will use the local storage on the application layer to ensure high data availability.

6.2.3 Configuring SangforLogging for Workload

Step 1: Select "Clusters" and "Projects" in the upper left corner of the interface. Click "Project" to go to the Workload page on the Resource interface and then click "Deploy".



Step 2: Set basic workload configuration and then click "Enable".

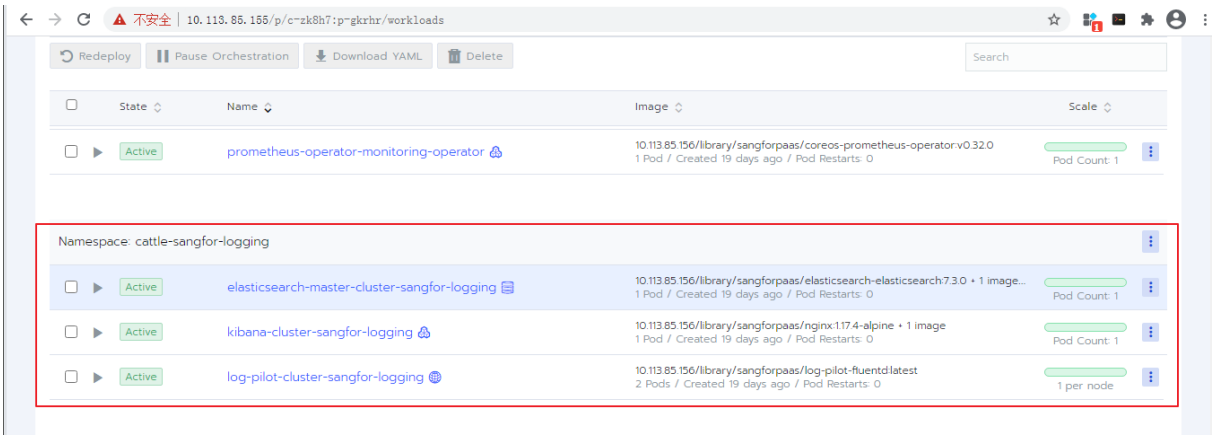


1. Configure the name.
2. Select to configure the workload type of the K8s pod, i.e., pod controller type.
3. Docker Image: Configure the image address of the application, which can be searched in the Harbor VIP built-in registry.
4. Select the namespace of the project where the application will be deployed. You may also create a new namespace.
5. SangforLogging Log: Enable Standard Log Collection and disable Container File Log Collection.

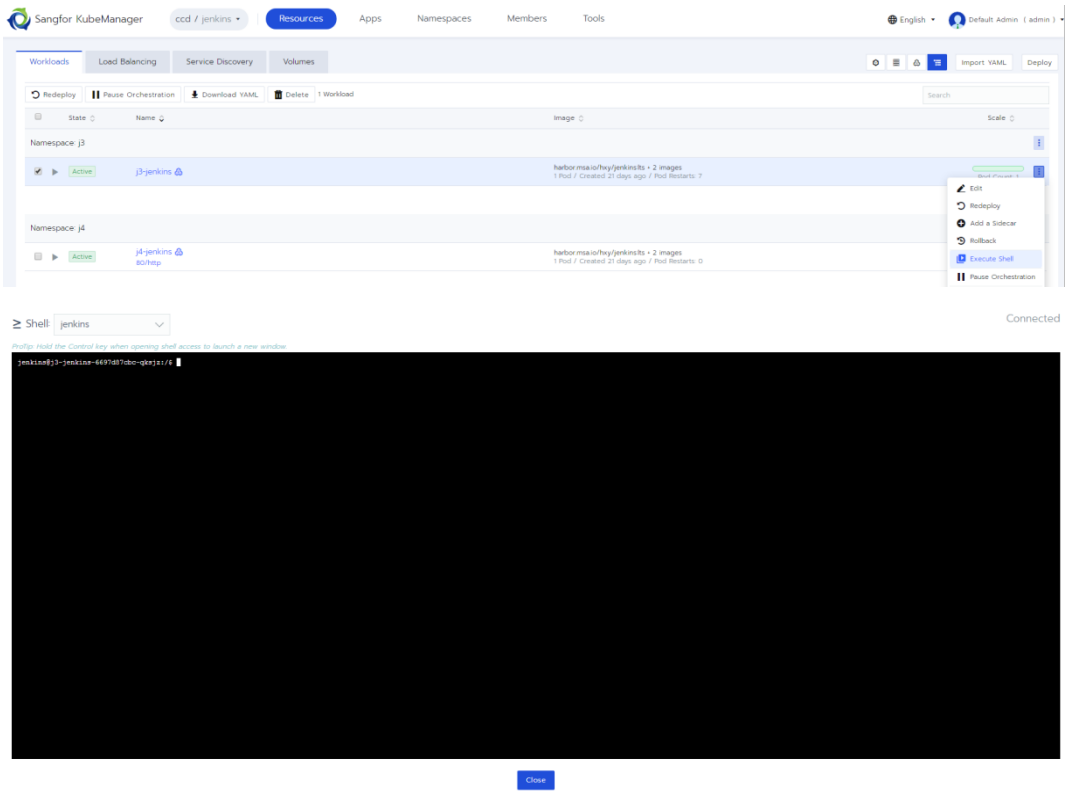
6.3 Verification

[11] The SangforLogging log component consumes relatively more memory, which is 2 GB by default. When enabling logs, make sure that there are surplus worker resources for running the log component.

Step 1: On the interface, select "Cluster" and "System Project". On the "Resource" -> "Workload" interface, check whether the workload of namespace **cattle-sangfor-logging** is working normally.



Step 2: Access workloads to generate access logs. Log collection by the system is available.

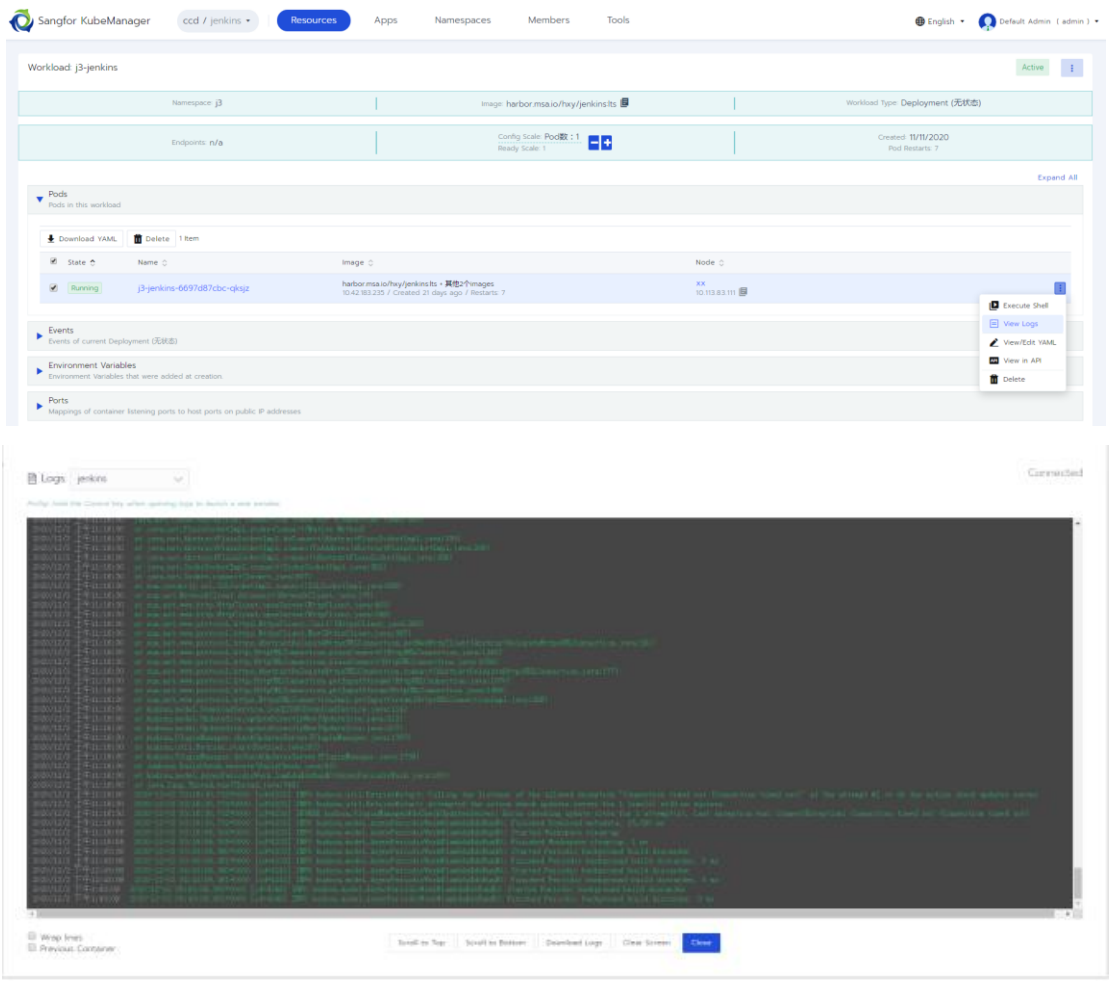


Access to workloads can provide external access to applications and then generate logs by publishing services.

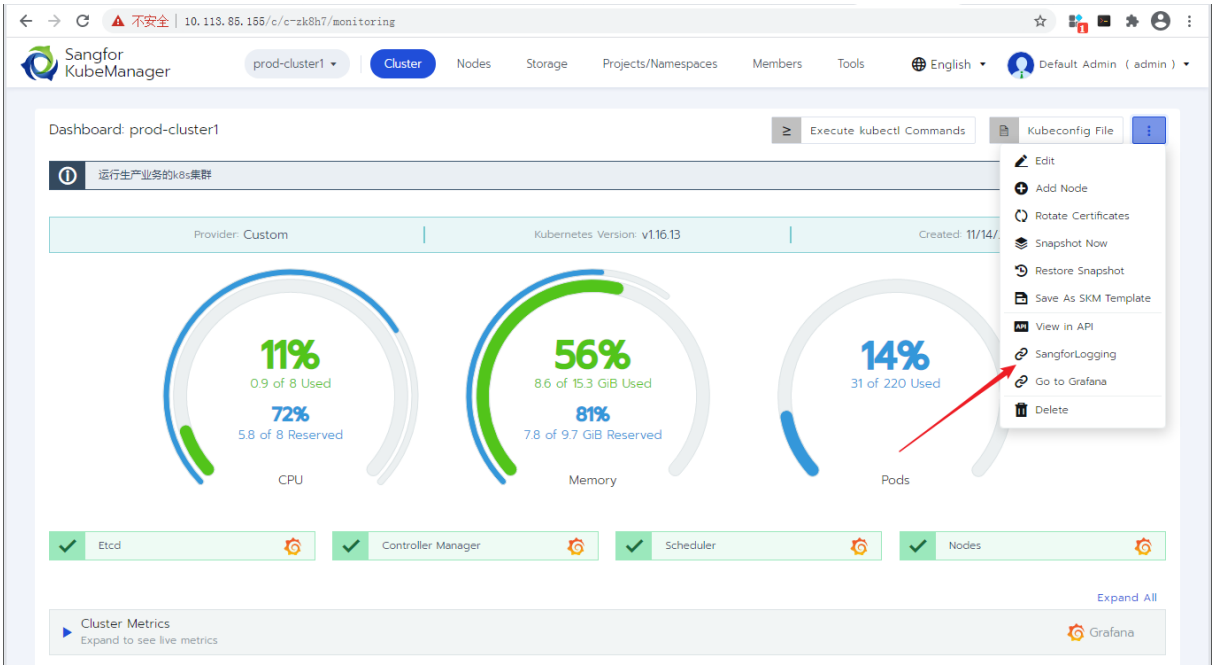
Here, we simply run a command line in the container to get access and generate container application logs.

1. Click "Workload" -> "Run Command Line".
2. Run the **wget -O- localhost** command to simulate access to workloads and the generation of application logs.

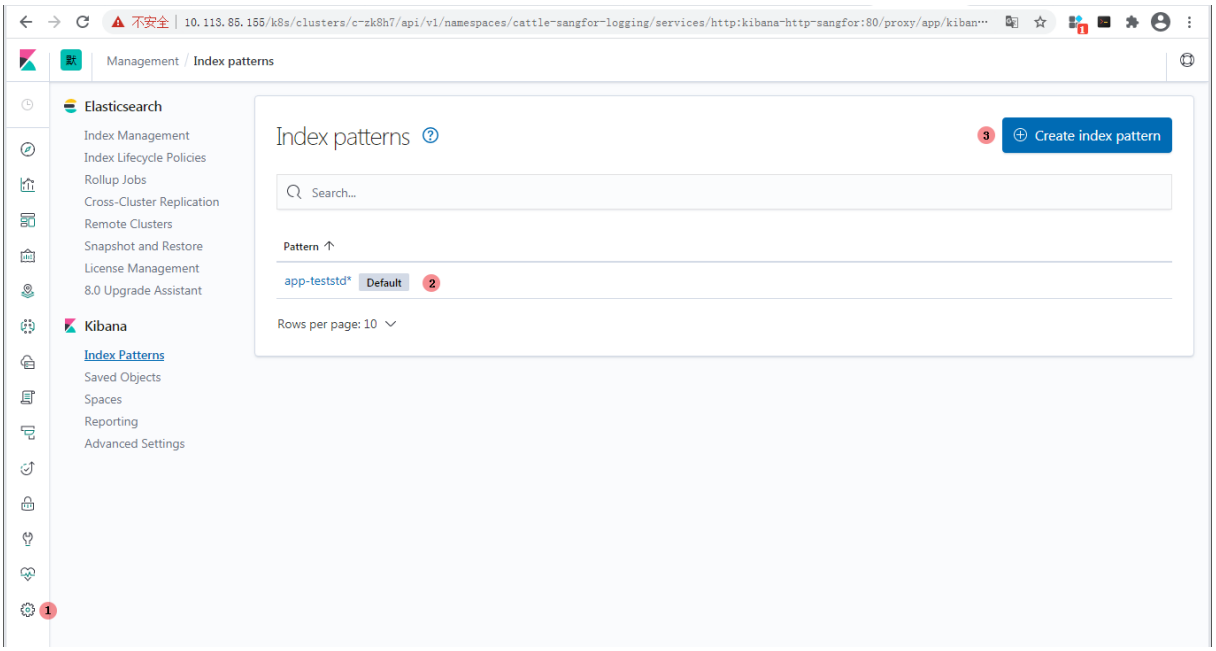
Step 3: Click "Workload" -> "View Logs", and the real-time container logs are displayed. Confirm the logs generated in Step 2.



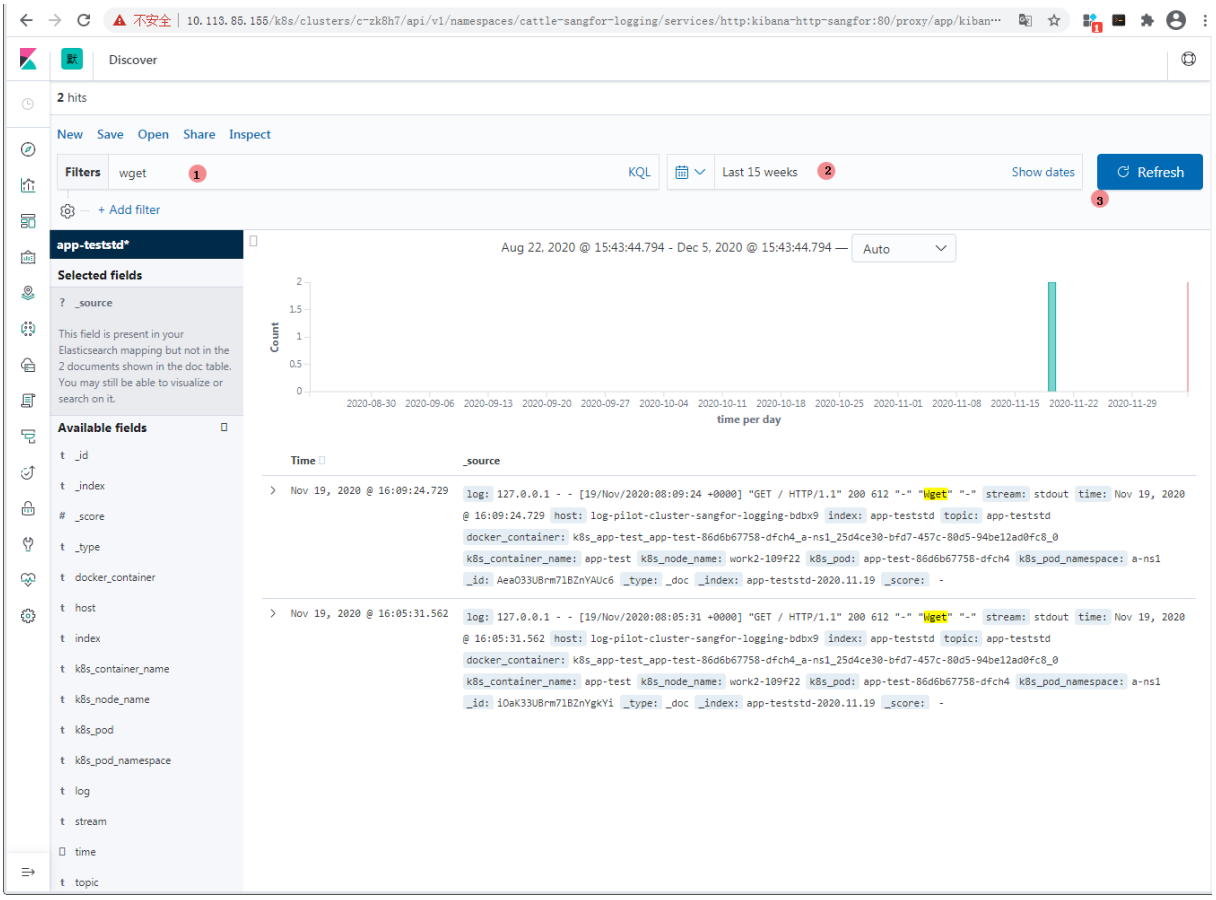
Step 3: On the cluster interface, click "Cluster Name" to go to the cluster dashboard. Then click "More" on the upper right corner of the interface and see if there is "SangforLogging" from which you can enter the Kibana Log Retrieval interface.



Step 4: Create the index mode on the Kibana interface. As shown in the figure, create app-test* to match logs collected by the log collection system.



Step 5: Click "Discover" on the Kibana interface to retrieve logs.



1. Search keywords.
2. Set time range of logs.
3. Click "Refresh" to retrieve logs already persisted. Log analysis and troubleshooting measures are available.

6.4 Questions for Discussion

When enabling cluster logs, think about:

1. Is the log data reliable, if the SangforLogging log collection system does not use shared storage? How many worker nodes are needed to achieve reliability?
2. What are the two log types the log system generally collects? How many file paths can be configured for a workload?
3. Can application logs be viewed through KubeManager when the SangforLogging log collection system is not enabled? Can logs be persisted and searched?
4. Are there any differences between the logs available from "Workload" -> "View Logs" and those collected by SangforLogging log collection? Persistence? Retrieval?

7 Experiment 6: Creating a K8s Application

7.1 Introduction

7.1.1 About the Experiment

1. By creating a project, a namespace, and a workload, as well as using the fine-grained configuration, on the interface, this experiment successfully deploys a workload and realizes real-time full-lifecycle management of workloads on the interface.
2. By configuring load balancing, this experiment publishes services of applications and allows external access to K8s applications.

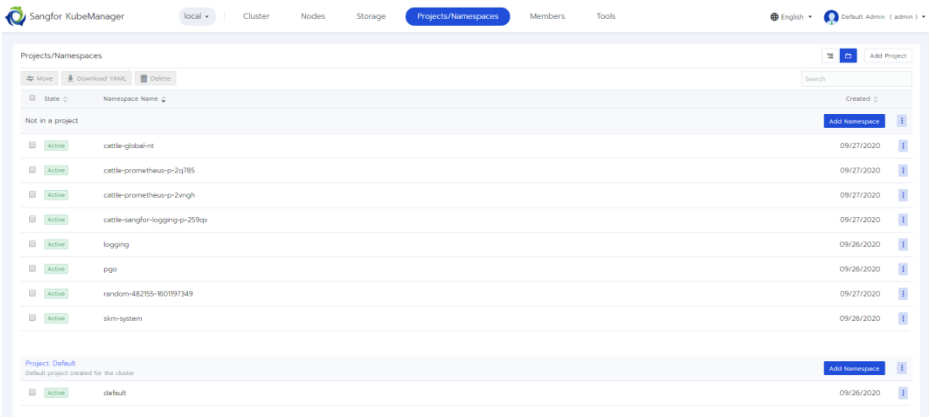
7.1.2 Purpose

1. Workload deployment
   - To find out the method for KubeManager to deploy K8s workloads
   - To know and understand how KubeManager deploys container applications provided by open-source communities
   - To understand KubeManager's management over the workload lifecycle and the O&M it provides
   - To find out the YAML configuration standard of K8s Load Balancing, including K8s parameters and the specific parameters related to the container[12], and to find out the configuration conversion for migration from a traditional container to the K8s
2. Service publishing
   - To find out the method to configure network ports
   - To find out how to configure service publishing for L4 load balancing
   - To find out the service types in the K8s and how to use L7 or L4 to publish services, thereby realizing access to north-south traffics of services

7.2 Steps

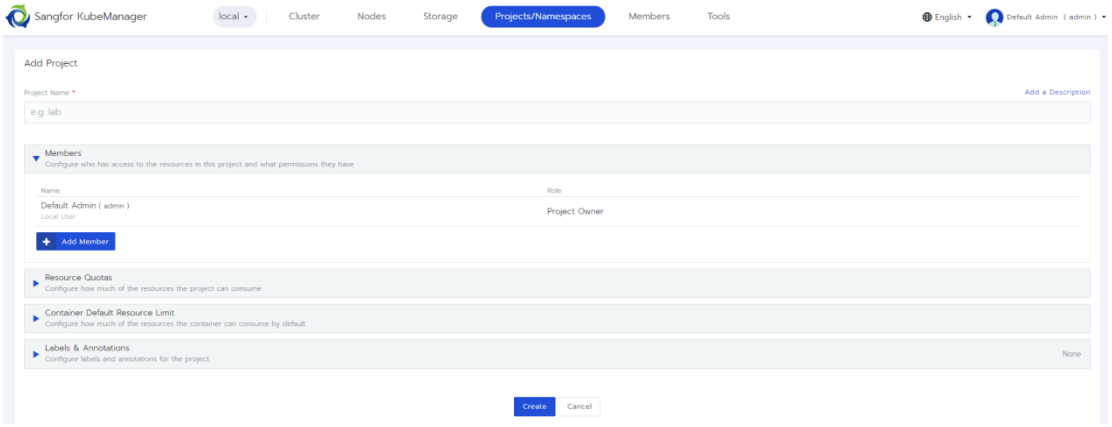
7.2.1 Project Creation

Step 1: On the cluster interface, click "Cluster Name" to go to the cluster dashboard. Click "Projects/Namespaces" to go to the "Projects/Namespaces List" interface and then click "Add Project".



[12] OCI Runtime Specification

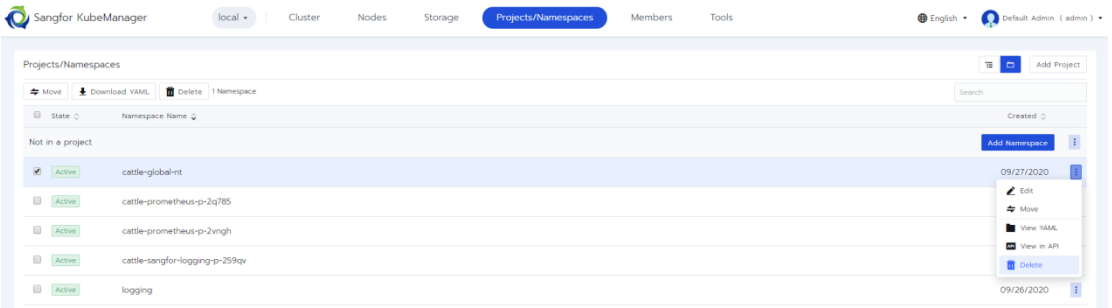
Step 2: Add a project.



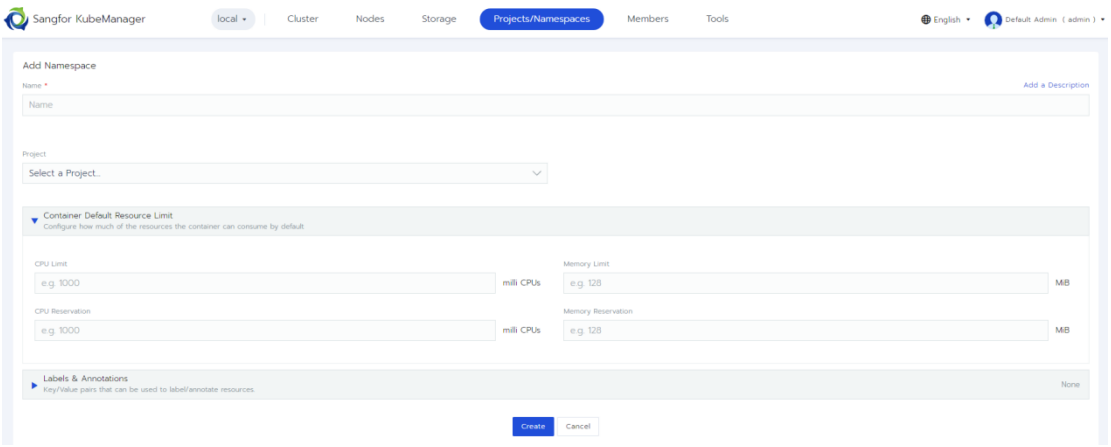
Config Item	Description
Project Name	Name of the project
Description	A brief description of the project
Add Member	Configure users allowed to access resources in the project and user permissions, to provide users the permission to use the project
Add Quota	Configure the amount of resources the project can use. This item allows for the configuration of resource allocation and limit; there will be no limit if it is not configured
Default CPU Limit for Container	Default value of the CPU limit for applications deployed under this item; there will be no limit if it is not configured
Default CPU Preservation for Container	Default value of CPU requests for applications deployed under this item; there will be no limit if it is not configured
Default Mem Limit for Container	Default value of mem limit for applications deployed under this item; there will be no limit if it is not configured
Default Mem Preservation for Container	Default value of mem requests for applications deployed under this item; there will be no limit if it is not configured
Comment/Tag	Projects are also a type of resources; you may add tags or comments to projects

7.2.2 Adding a Namespace

Step 1: On the cluster interface, click "Cluster Name" to go to the cluster dashboard. Click "Projects/Namespaces" to go to the "Projects/Namespaces List" interface, find the project, and then click "Add Namespace".



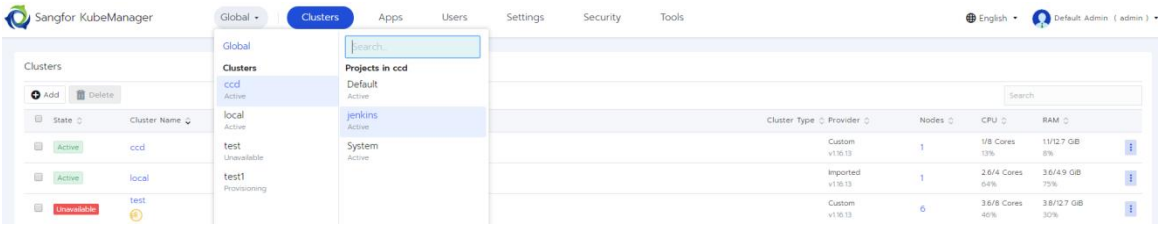
Step 2: Add a namespace.



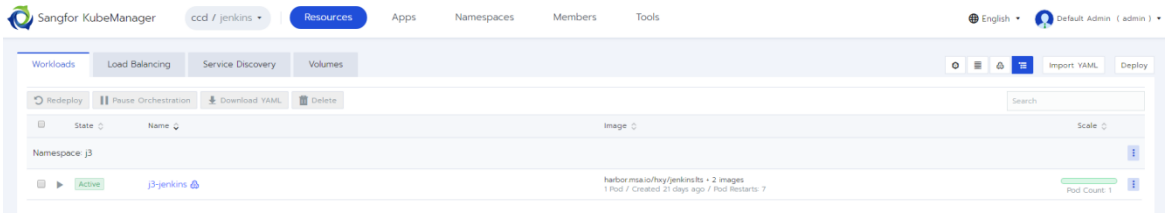
Config Item	Description
Name	Name of the namespace
Description	A brief description of the namespace
Project	The project to which the namespace belongs
Default Limit for Container	Default resource limit for the container deployed under the namespace; there will be no limit if it is not configured
Tag/Comment	Namespaces are also a type of K8s resources; adding tags or comments to resources is supported

7.2.3 Creating Workload

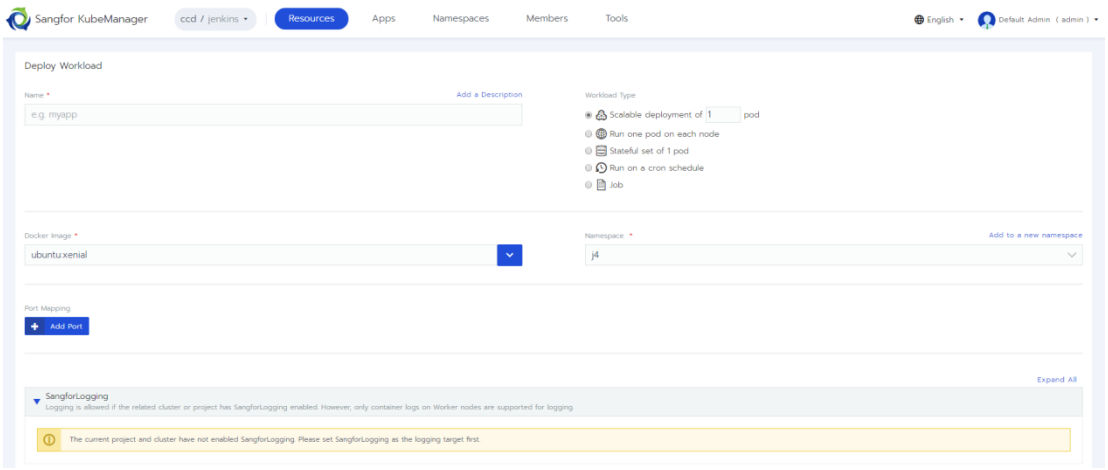
Step 1: Select the "Clusters" and "Projects" in the upper left corner of the interface. Click "Project". The "Workloads" on the Resource interface is displayed.



Step 2: Click "Deploy".



7.2.4 Basic Workload Configuration



Step 1: Configure the name.

Step 2: Add a short description of the configuration application (optional).

Step 3: Select to configure the workload type of the K8s pod, i.e., pod controller type.

- Deployment

  Introduction: Deployment is most commonly used by K8s to deploy stateless applications. Combined with other parameters and copies configured on the interface, K8s describes the target state in the Deployment. The Deployment controller changes the actual state of deployed applications into the expected state at the controlled rate, i.e., making them run in the expected state.

  Application scenario: You may search for various types of application images available on the market, including those provided by open-source communities, on https://hub.docker.com/. For typical images, such as DNS service image, OpenLDAP image, and Gitlab image, if K8s deployment mode is not available, they can generally run container applications by means of Deployment, which is a common method.

- DaemonSet

  Introduction: It is used to deploy a pod on each host. DaemonSet ensures that the copies of only one Pod are running on all (or some) nodes. A new pod will be created for each new node added into a cluster, and whenever a node is removed from the cluster, the corresponding pod will be deallocated. Deleting DaemonSet will delete all Pods it has created. Using the DaemonSet controller to create resources will run a Pod on each Node, such as log collection. Since the Pod is running on different nodes, each node must have a Pod for collecting logs. This is when DaemonSet can be used.

  Application scenario:

  – To run a cluster Daemon on each node. For example, run **glusterd** or **ceph** on each node.

  – To run a log collection Daemon on each node, such as **fluentd** or **logstash**.

  – To run a monitoring Daemon on each node, such as **Prometheus Node Exporter** or **collectd**.

- StatefulSet

  Introduction: It is used for the deployment of stateful applications. StatefulSet is

used to manage the workload API objects of such applications. The Deployment and DaemonSet noted previously are for stateless applications. In reality, however, many services are stateful, such as MySQL cluster, MongoDB cluster, and ZK cluster. Most of them have the following characteristics:

- Every node has a fixed ID, through which members in a cluster can find and communicate with each other.
- The cluster scale is relatively fixed and cannot be changed arbitrarily.
- Each node in the cluster is stateful and typically persists data to the permanent storage.
- If the disk crashes, a node in the cluster cannot run normally and cluster functions will be impaired.

- Job

  Introduction: It is used for running Pods simultaneously. The Job creates one or multiple Pods and ensures that a specified number of Pods are successfully terminated. When the Pods are successfully terminated, the Job tracks and records the number of successful Pods. The Job ends once the threshold of successful Pods is reached. Deleting the Job will clear all Pods it has created.

  Application scenario: A simple application scenario is to create a Job object to keep running a Pod in a reliable manner until the Pod is completed. When a Pod fails or is deleted (for example, because of a node hardware malfunction or reboot), the Job object will enable a new Pod. The Job can also run multiple Pods in a parallel way.

- CronJob

  Introduction: It is used for running Pods at a specified time. A CronJob object is like a line in a crontab (cron table) file. It is written in the Cron format and periodically implements Job at a given scheduling time.

  Application scenario: CronJobs are useful for creating periodic and repetitive tasks, such as data backup or email sending. They can also be used to schedule the implementation of independent tasks at a specified time, for example, implementing a Job when a cluster seems to be very idle.

Step 4: Select the namespace in which the application will be deployed in the project. You may also create a new namespace. In K8s, the namespace is used for application isolation management.

Step 5: Configure the application image address for "Docker Image", following this format: REPOSITORY[:TAG|@DIGEST] (for example, 10.113.85.156/library/sangforpaas/nginx:1.17.4@sha256:f3f1...5458). Here, TAG and @DIGEST are optional. For example, you may just configure the address as 10.113.85.156/library/sangforpaas/nginx. The default tag is "latest" and digest is optional.

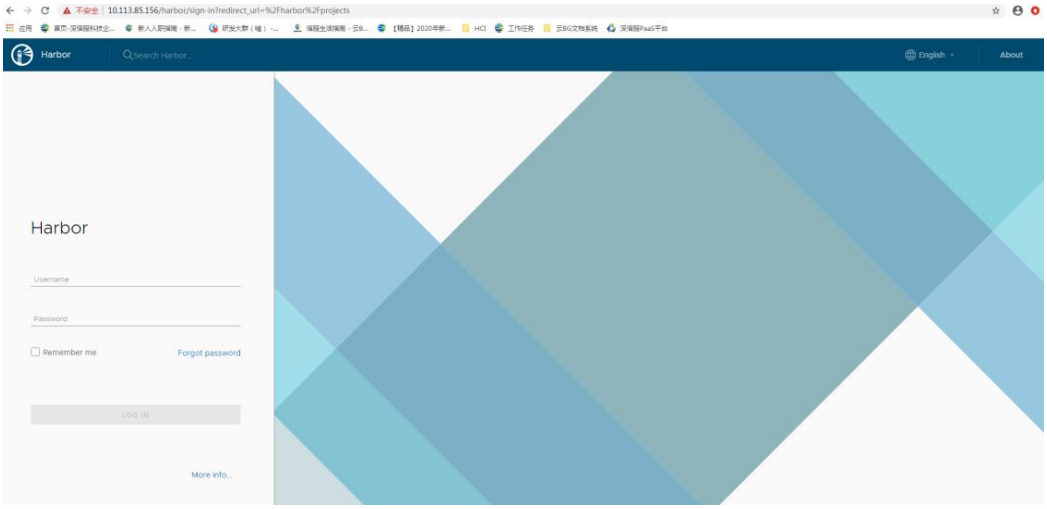
"Docker Image" is an image configuration entrance. Any image address consistent with the above format is supported[13]. By default, Docker Hub is used. In a private cloud scenario, the

[13] For the Docker Image address, it should be noted that by default, the docker pull image requires the registry to provide REPOSITORY, an HTTPS carrying an authentic SSL license. If the Docker Image is a self-built registry, the address may be an HTTP or may use HTTPS' self-signed SSL license. It is required to ensure that the docker configuration of all nodes in the K8s cluster can meet the docker's insecure configuration requirements.

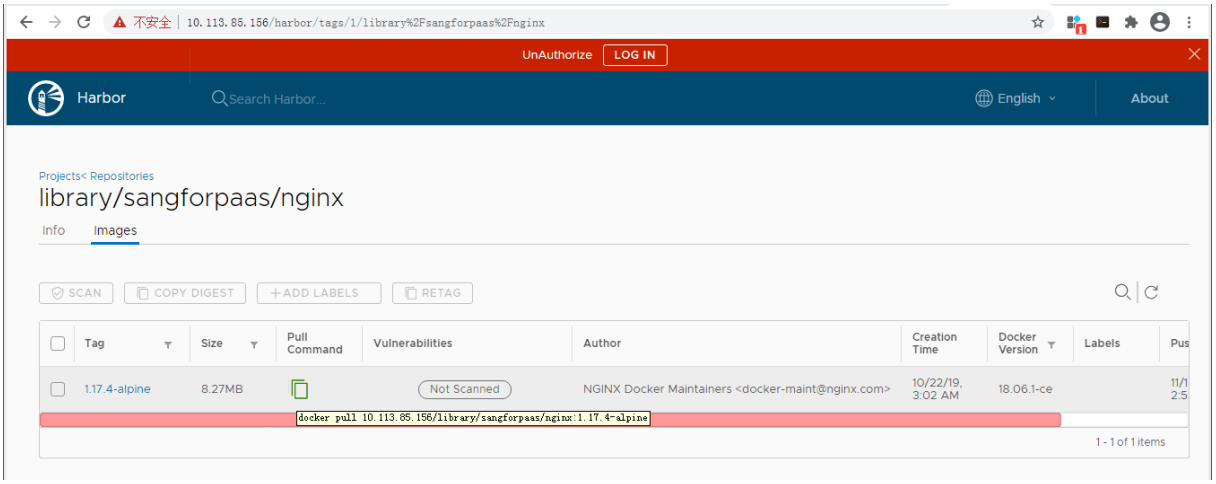
internal registry[14] is typically used for LAN communication. For application creation, K8s automatically pulls images according to the docker image address, before enabling workloads.

If the docker image uses the built-in registry, the steps for retrieving images are as follows:

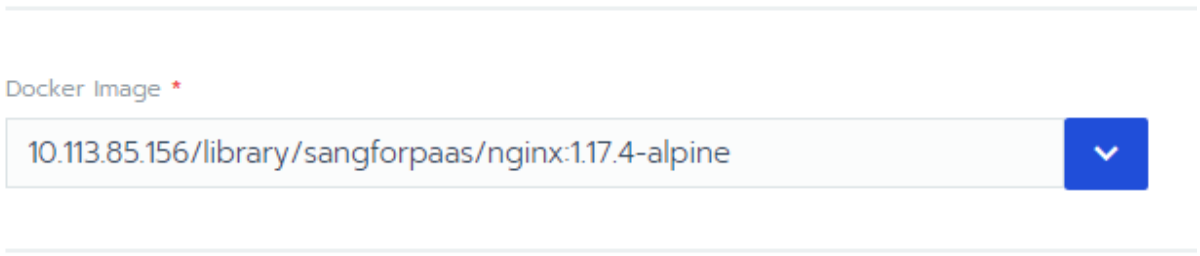
1. Enter HARBOR VIP in the browser and search for the image.



2. Copy the image address.



3. Paste the image address and configure the docker image.



The basic workload configurations include the required configuration items for deploying workloads. Others are optional configuration items. Load Balancing supports fully fine-

[14] When the KubeManager is deployed, there will be a highly available built-in registry. Application deployment on the LAN generally pushes the image of the application to be deployed to the built-in registry and then configures the internal Docker Image address. The default address of the built-in registry is **Harbor VIP**.

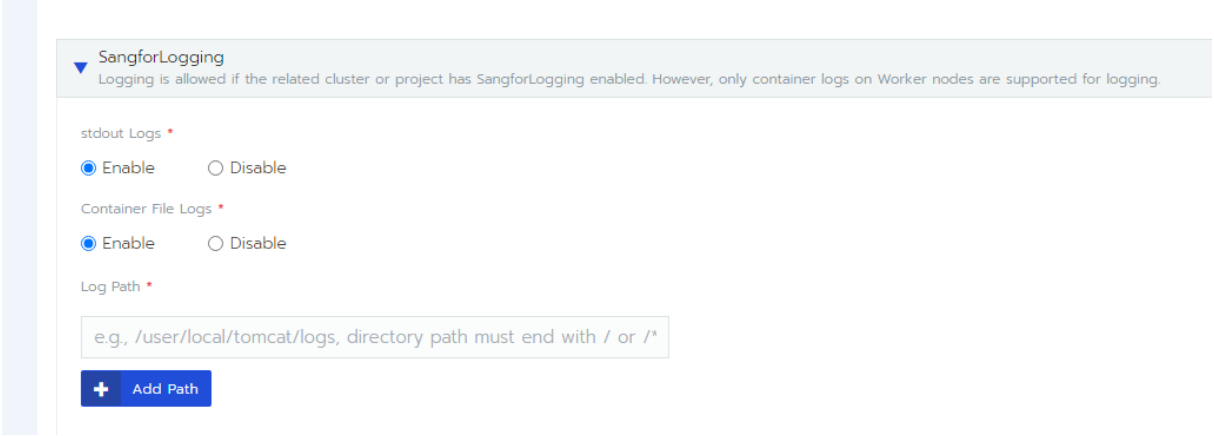
grained K8s configuration. Most other configurations can be customized according to the application, such as environment variables, log configuration, and storage configuration. This is also critical for considering how to configure original special image configurations on the platform during application migration.

7.2.5 Configuring Port Mapping

The port mapping is configured with the K8s service. By the service or port mapping, applications in a K8s cluster or external access applications can communicate with each other, and the network service abstract method provided by K8s for applications is available. Port mapping is an optional configuration. Application deployment and service publishing can be decoupled. Configuring service publishing after Load Balancing deployment can realize access to applications (see 3.2.5).

7.2.6 Configuring SangforLogging

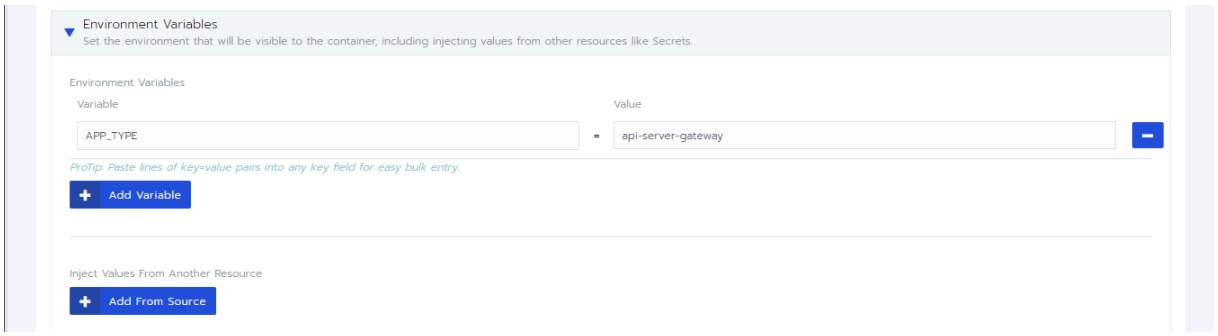
When SangforLogging enables log collection, Load Balancing can configure log collection. The log service only supports collecting container logs on the Worker role node.



Config Item	Configuration	Description
Standard Log Collection	Enabled	The standard output of the container, which is automatically collected and persisted by the log collection system for retrieval
Container File Log Collection	Disabled	Can be enabled when a container application has file logs. By configuring the log path, the collection system automatically collects and persists logs for retrieval.

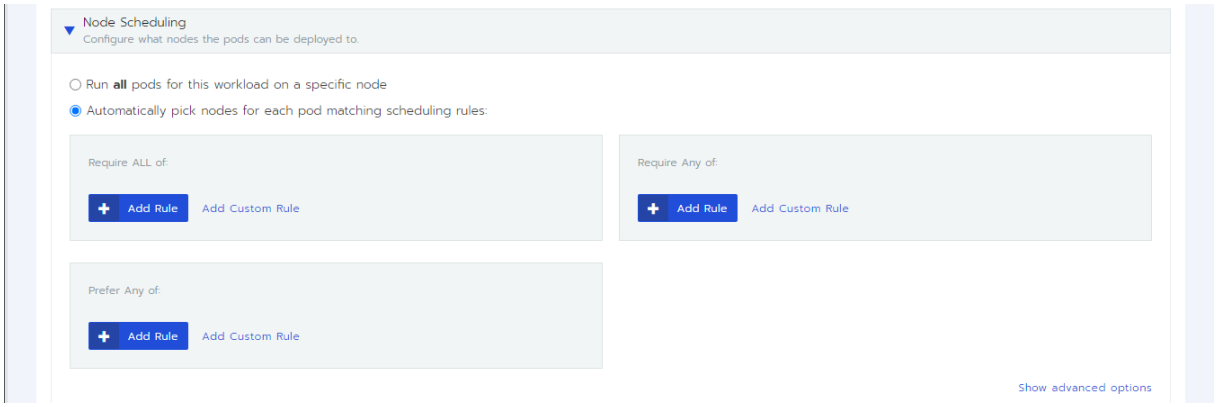
7.2.7 Configuring Environment Variables

The setting of environment variables visible in the container, including values injected by other resources (e.g., ciphertext); a way for applications to inject data. Container applications can configure environment variables by means of key values to complete the application environment configuration. The environment variables configured on the K8s will ultimately construct operating container environment variables by enabling kubelet, thereby injecting environment variables.



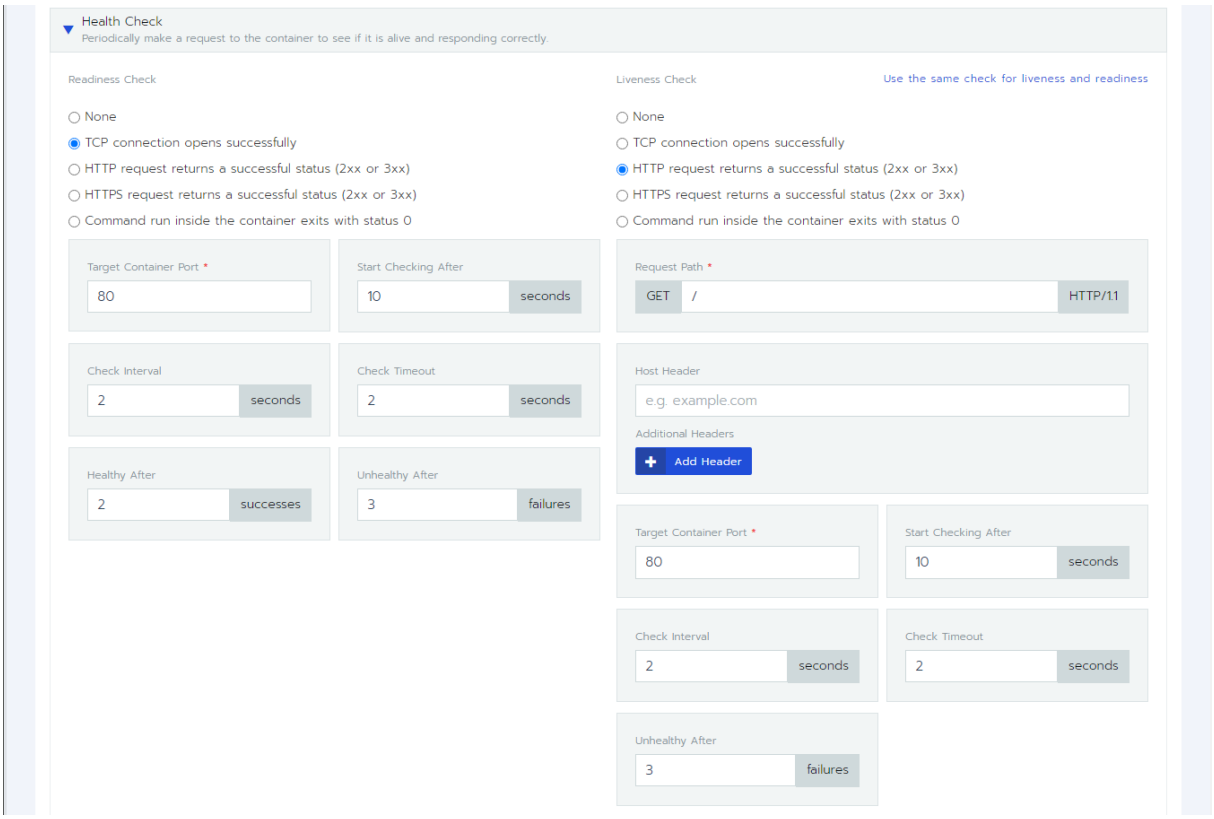
7.2.8 Configuring Host Scheduling

The configuration of the host scheduling rule corresponding to Pods. When Load Balancing is running in a distributed K8s cluster, the pod host scheduling can be done based on the selected host or flexibly based on the tag matching rule of K8s by conducting scheduler computation.



7.2.9 Configuring Health Inspection

The configuration of periodic requests from the kubelet to the container, in order to inspect the latter's health. The kubelet conducts a health inspection (survival detector) to know when to reboot the container. A probe is a regular diagnosis implemented by the kubelet for the container.

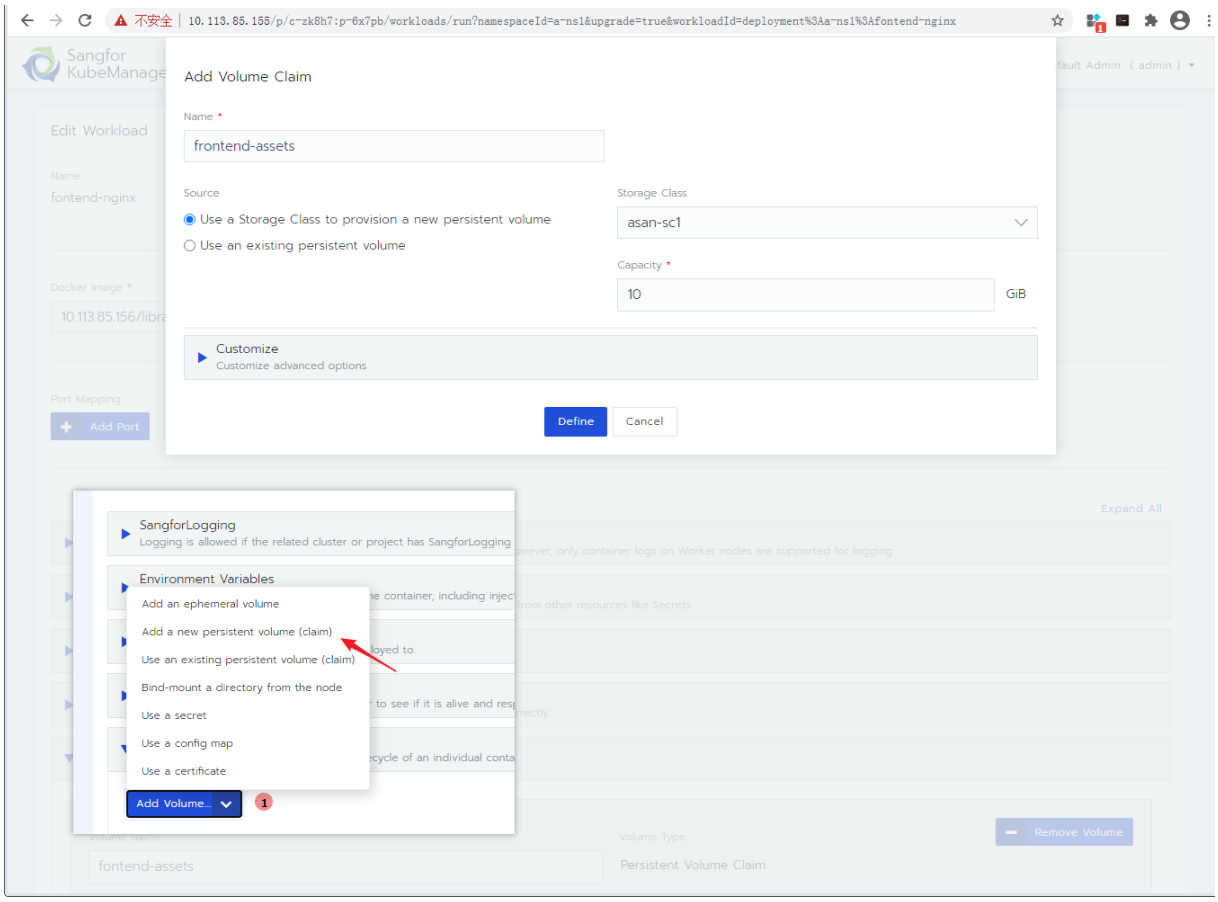


7.2.10 Configuring Data Volume

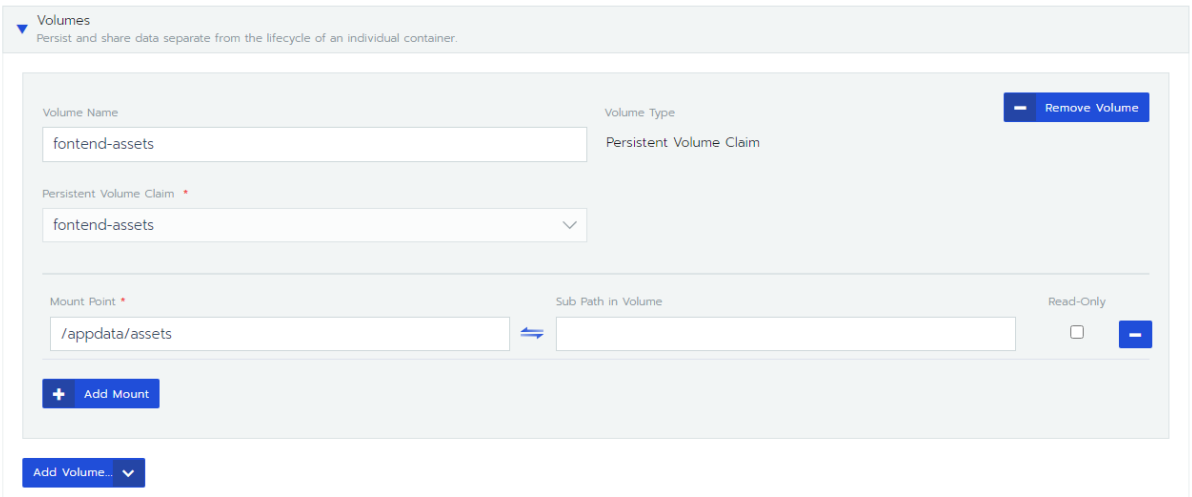
Load Balancing's storage configuration, persistence, data sharing, and lifecycle separation from independent containers; a way for applications to inject data. Supported storage types include:

1. hostPath
2. secret
3. configmap
4. License volume

Step 1: Add a new PVC, which can either use a storage class to dynamically provision to the new storage or be docked with an existing PV.



Step 2: Configure the path for mounting the container.



7.2.11 Configuring Commands

The configuration of the executable file to be run during the container startup and the parameters, which is the process configuration of the container runtime-spec.

7.2.12 Configuring Networks

The configuration of network namespaces, which supports the following customized configurations for applications.

1. Whether to use the host network
2. DNS policy to configure whether the cluster DNS server is used on a regular pod or on

the host network

3. Hostname configuration
4. The configuration of the **/etc/hosts** record for container applications (see Adding entries to Pod /etc/hosts with HostAliases).
5. Adding the DNS server, DNS search domain, and DNS resolution options for the container (see Pod's DNS Config).

7.2.13 Tag/Comment

You can add tags and comments to all K8s resources for K8s framework management.

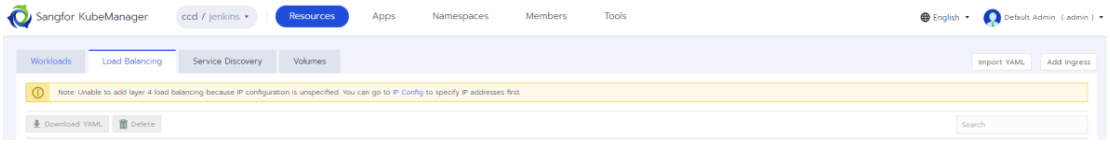
7.2.14 Security/Host Settings

To grant or limit the capability of the container to affect the running host (see Pod Security Standards - Capabilities).

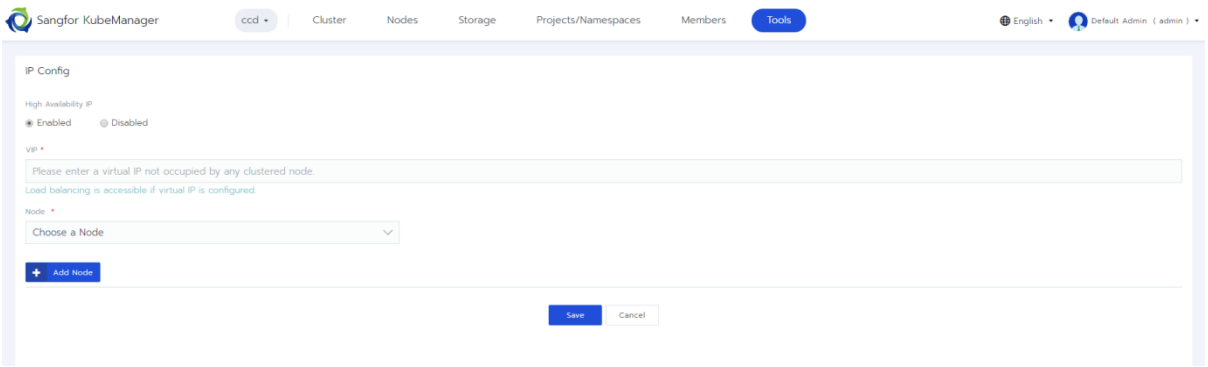
7.2.15 Network Interface Settings

To configure service publishing for applications in order to provide the capacity of external access to applications, when workload deployment is complete.

Step 1: Select the Resource interface of the project and click "Load Balancing". Currently, you cannot add L4 load balancing since you have not configured the network port. Click "IP Config" to configure it.



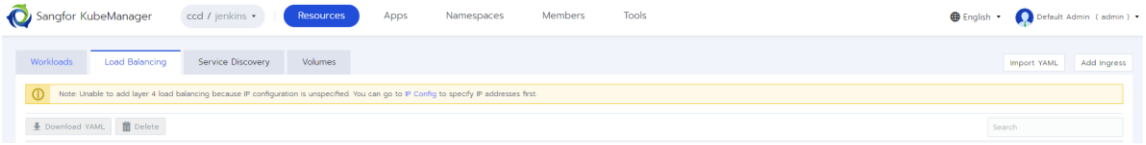
Step 2: On the "IP Config" interface, check "Enable" for "High-Availability= IP" and configure "VIP".



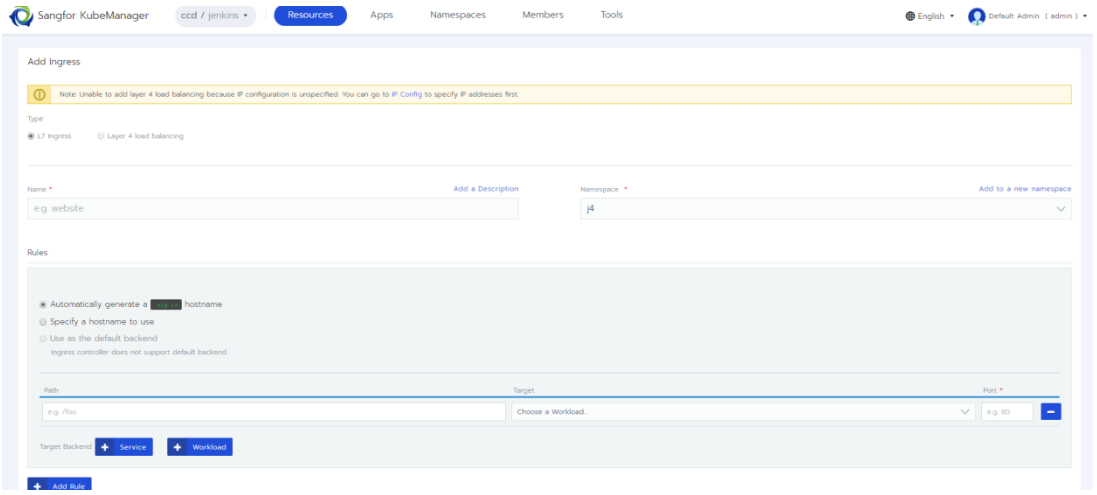
1. Configure the network port VIP.
2. Add the hosts and select two worker nodes. Click "Save".

7.2.16 L4 Service Publishing

Step 1: Select the Resource interface of the project. Click "Load Balancing" and click "Add Rule".



Step 2: Configure L4 load balancing.

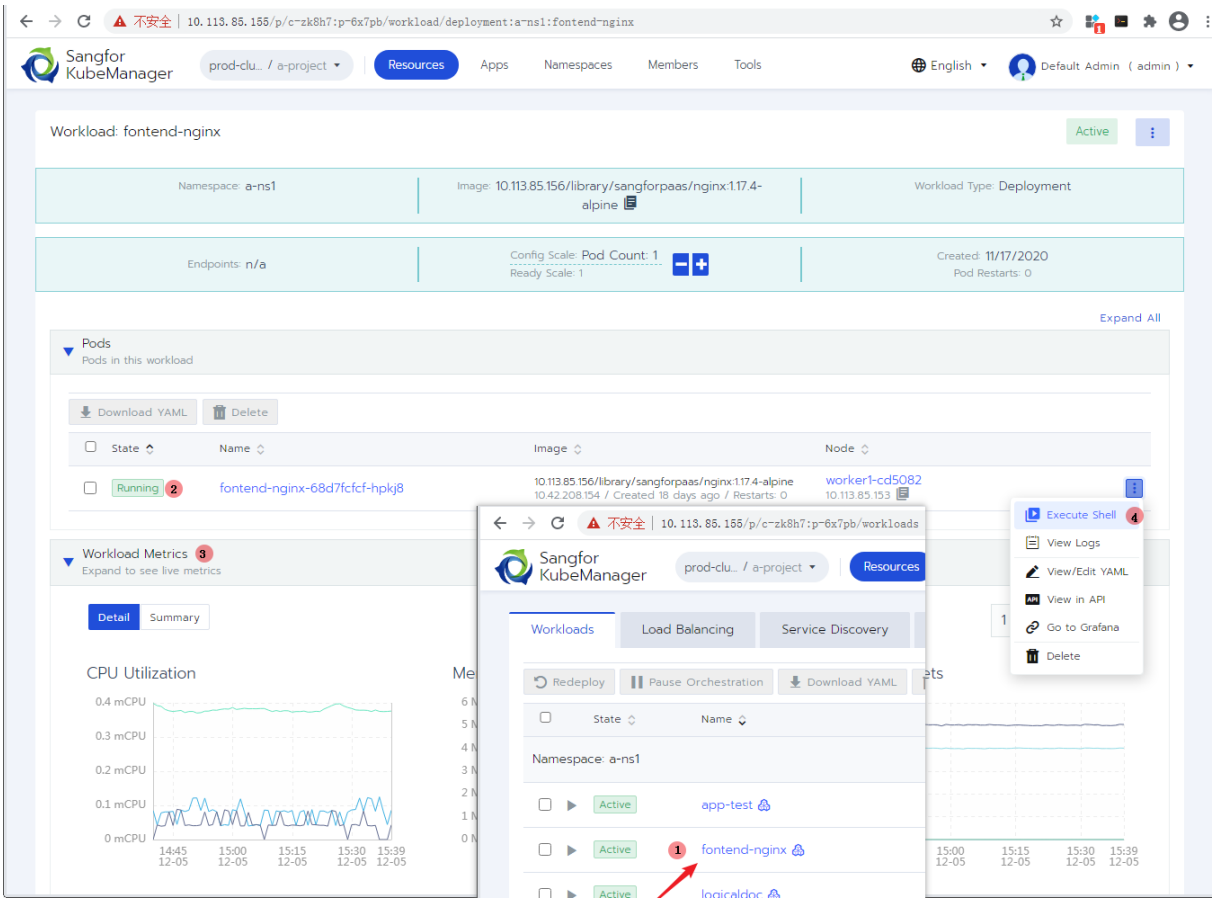


Configuration	Description
Name	Name of L4 load balancing
Type	Choose L4 type (TCP or UDP)
Monitoring Port	The port used to monitor the host
Service/Workload	If you select Workload, you need to manually configure the container port; if you select Service, just select a port.
Container Port	The port used to monitor actual container applications

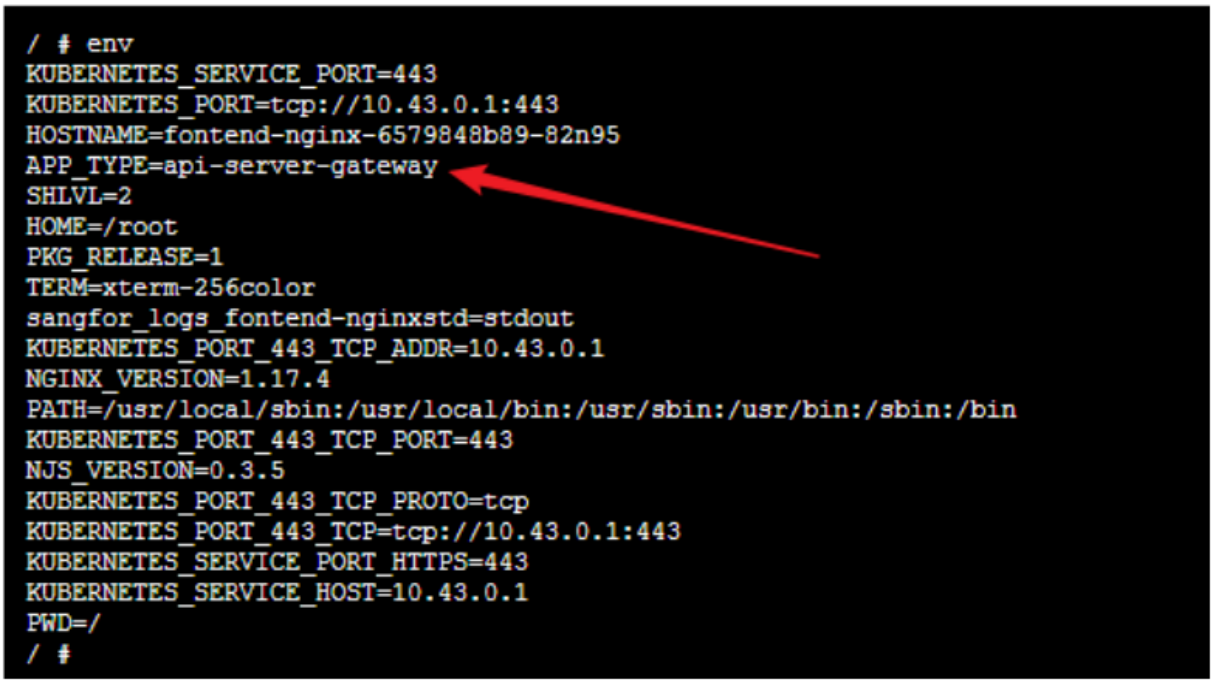
7.3 Verification

7.3.1 Deploying Workloads

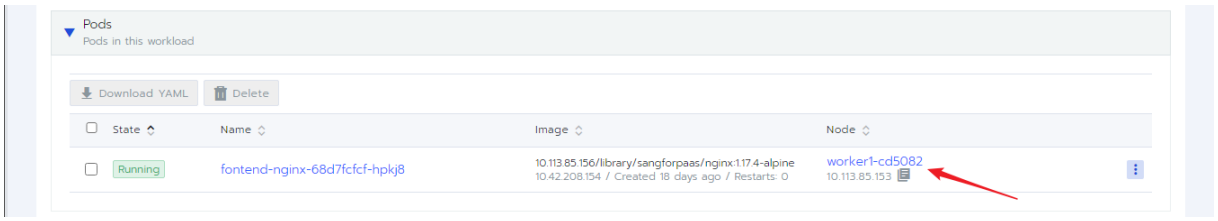
1. In "Resource" -> "Workload", click "Workload Name" to check whether the pod status is "Running" and whether there is real-time monitoring data of workload monitoring. Then click the "More" option of the pod and open "Run Command Line".



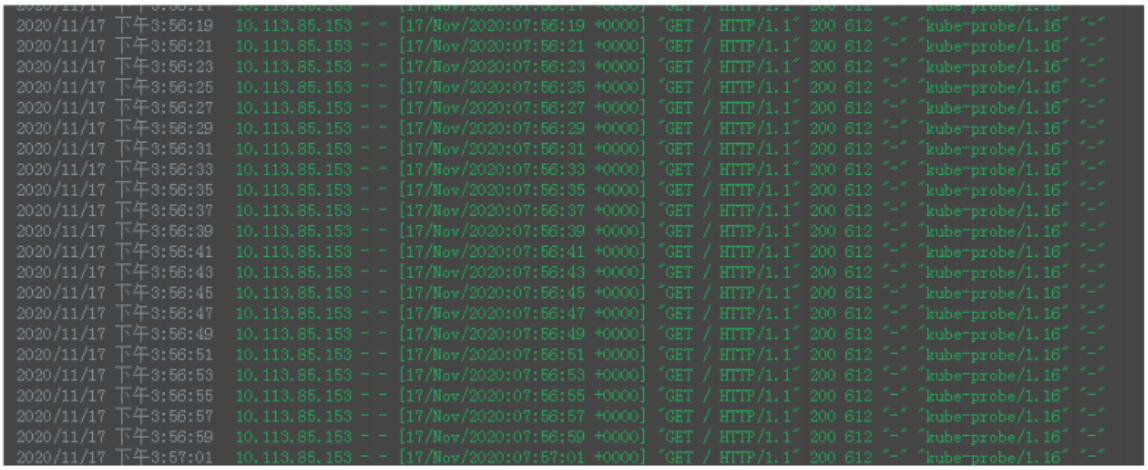
2. To configure environment variables, open "Run Command Line" and enter the **env** command to check whether environment variables are injected.



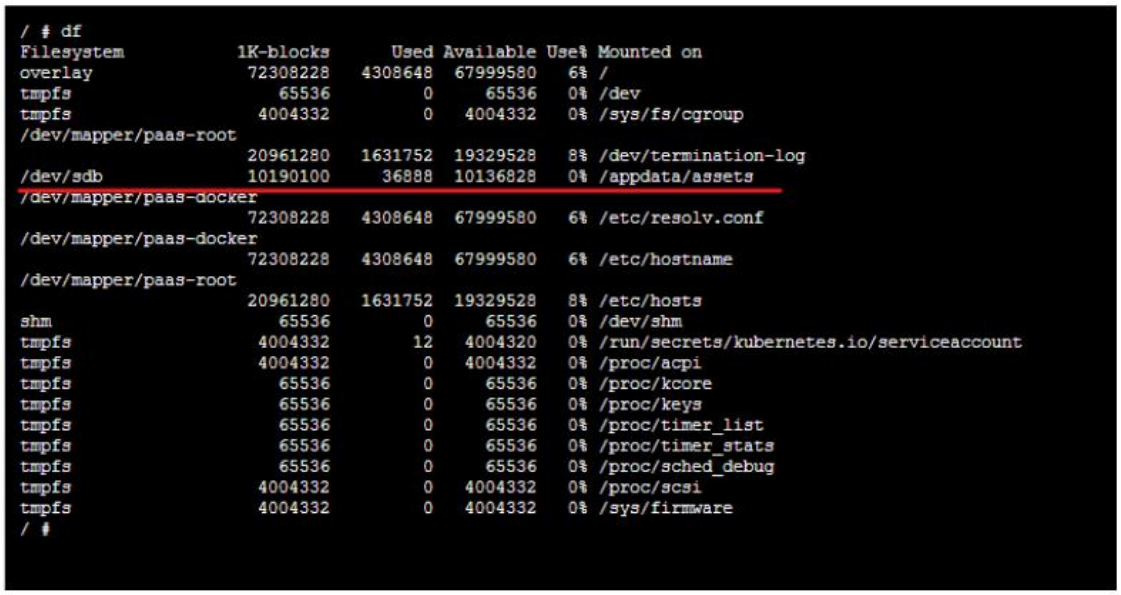
3. To configure the host scheduling, check whether the pod has been scheduled to the correct K8s host.



4. To configure the health inspection, click the "More" option of the pod and then open "View Logs" to check whether there are logs generated as a result of the probe's periodic access.

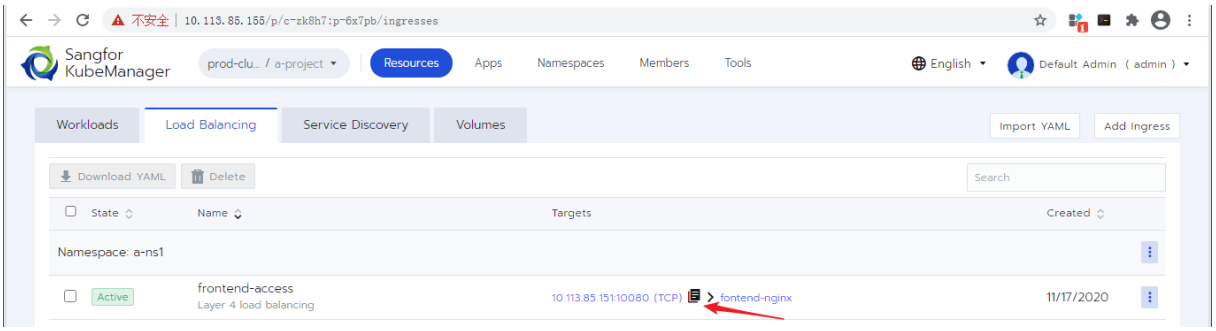


5. To configure the data volume, click the "More" option of the pod, open "Run Command Line", and then enter the **df** command to check whether the data volume has been mounted correctly.

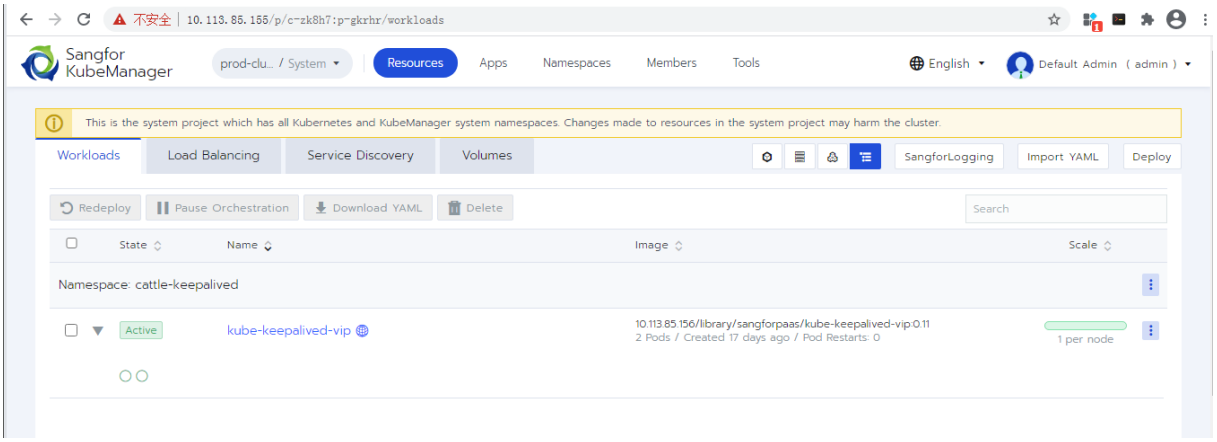


7.3.2 Service Publishing

Step 1: On the Load Balancing interface, click the copy button of the target. Check whether the application can be accessed through the browser.



Step 2: After enabling the network port, check whether it is deployed normally. Go to the Workload interface of the system project to check whether workload **cattle-keepalived** is running normally.



7.4 Questions for Discussion

When deploying Load Balancing, think about:

1. What are other ways to deploy K8s or container applications, other than clicking "Workload" on the interface?
2. What to do with the deployed application image address, if the K8s cluster environment is not connected to the Internet? How do you use the docker push to share images to the built-in Harbor registry and reference these images?
3. Does the "Workload" provided on the interface support the deployment of all K8s applications?
4. What will change in the updating policy, after a **Deployment** configures a data volume?
5. In docker image address configuration, what are the insecure configuration requirements for all node dockers in a K8s cluster, in order to configure a private registry address?
6. What are the methods for applications to inject data?
7. Can the YAML configuration file for applications that you can view and edit on the interface be exported to run on another K8s?
8. What does the concept of "network port" mean? What does it use to realize VIP? What conflicts will the keepalived's VRRP in a physical network cause?
9. What applications are suitable for using L7 for publishing services? What applications are suitable for using L4 for publishing services?

8 Experiment 7: Use of App Store

8.1 Introduction

8.1.1 About the Experiment

In this experiment, we upload a tomcat application image and the corresponding helm chart package to the built-in registry through docker push and deploy K8s applications in a way similar to software package management using the app store from the KubeManager interface.

8.1.2 Purpose

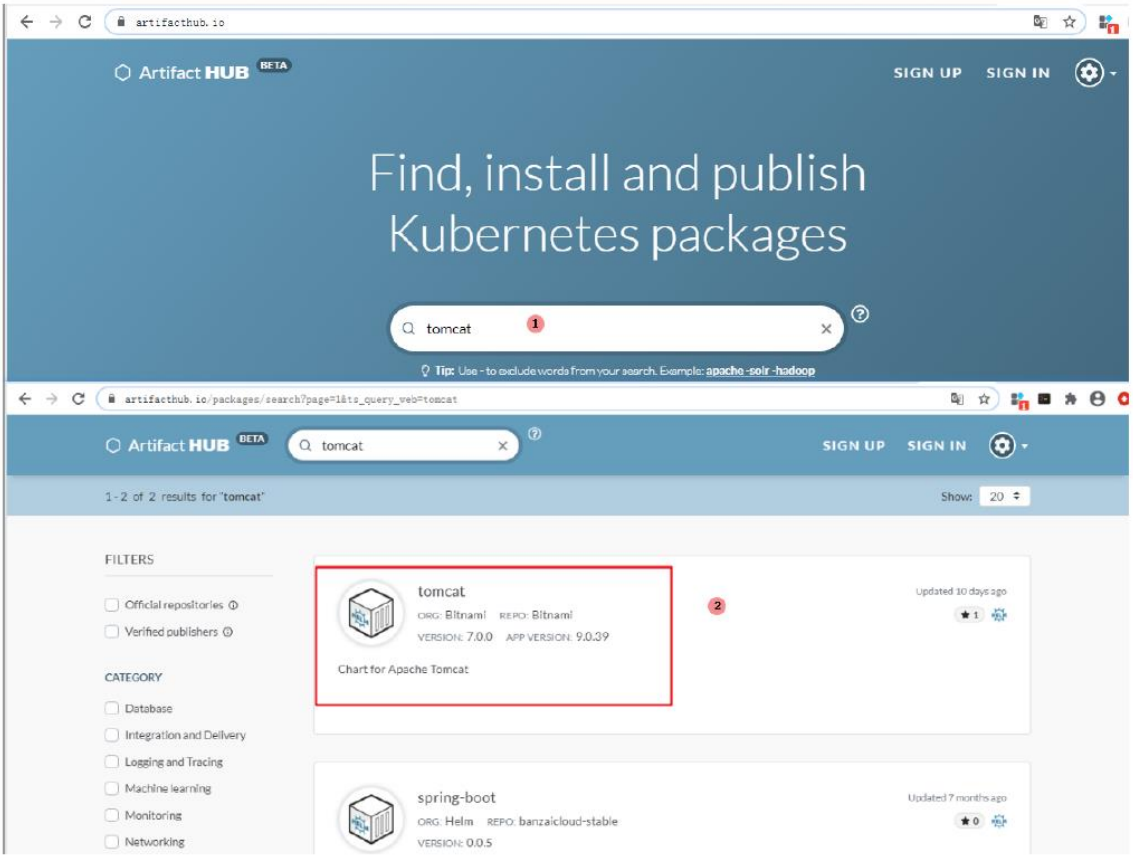
- To find out how to push images to the built-in registry
- To find out how to upload application chart packages to the built-in registry
- To find out how to configure the parameter customization of a helm chart package on the app store interface
- To find out how to deploy and configure helm applications by using the app store
- To understand that the app store also provides a more standardized way of application deployment by helm chart packages, in addition to the workload for application deployment provided by the platform

8.2 Steps

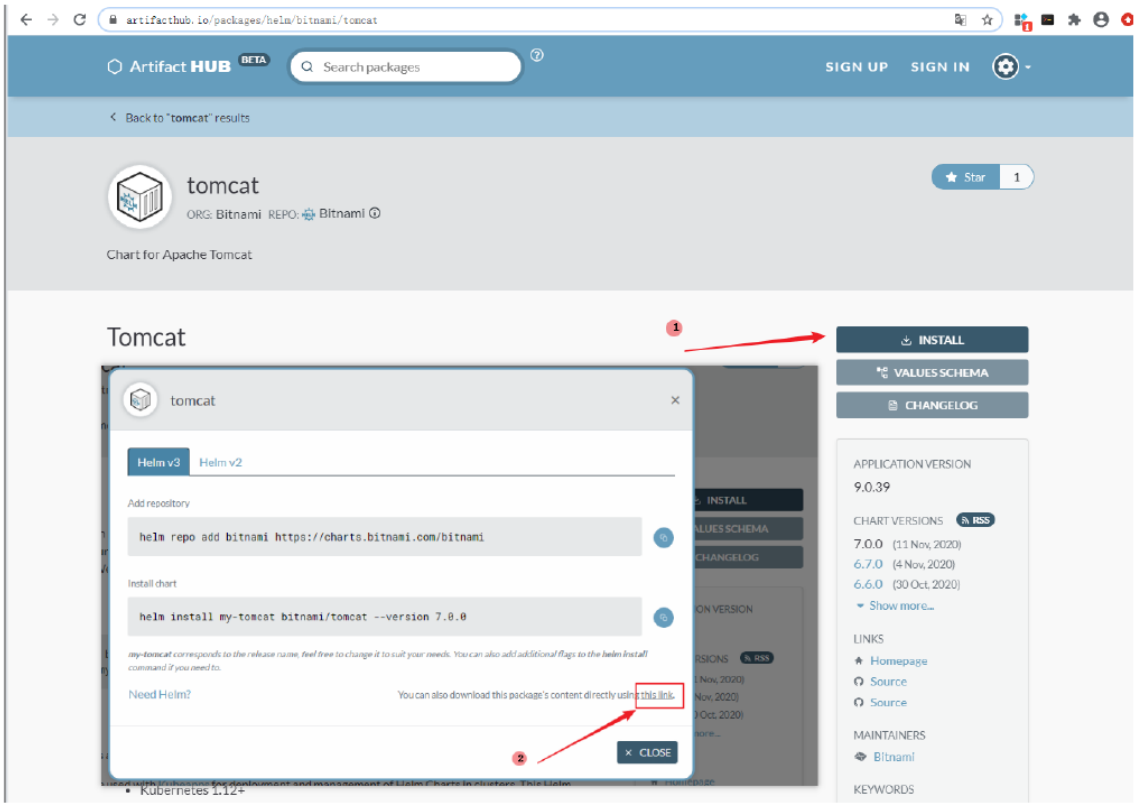
8.2.1 Preparation of Chart Packages

1. Preparation of Helm charts

Step 1: Open https://artifacthub.io in the browser, search for the image **tomcat** on the site, and then select the chart package provided by bitnami.

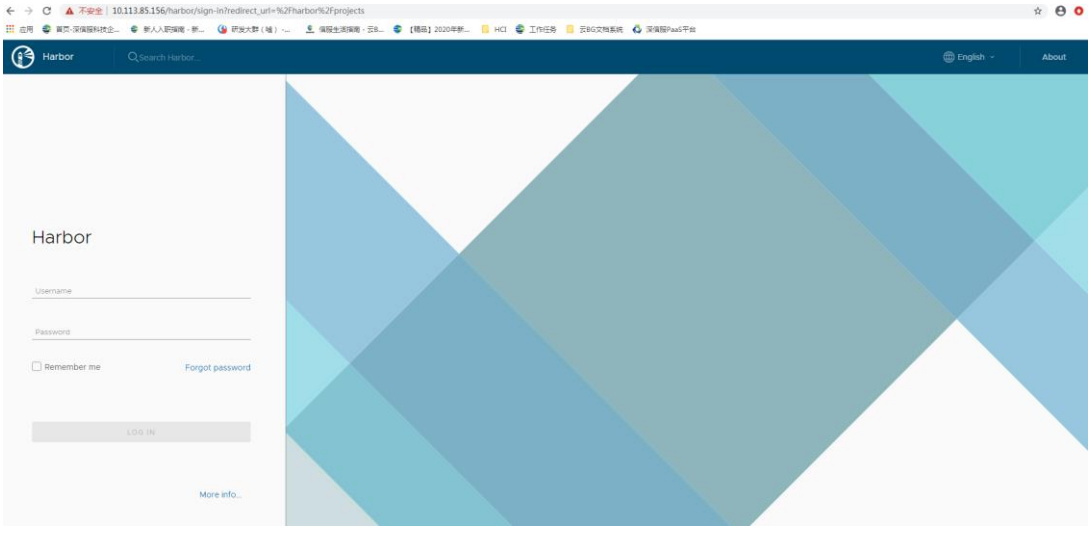


Step 2: Click "INSTALL" and then the download link on the pop-up page to download the chart package.



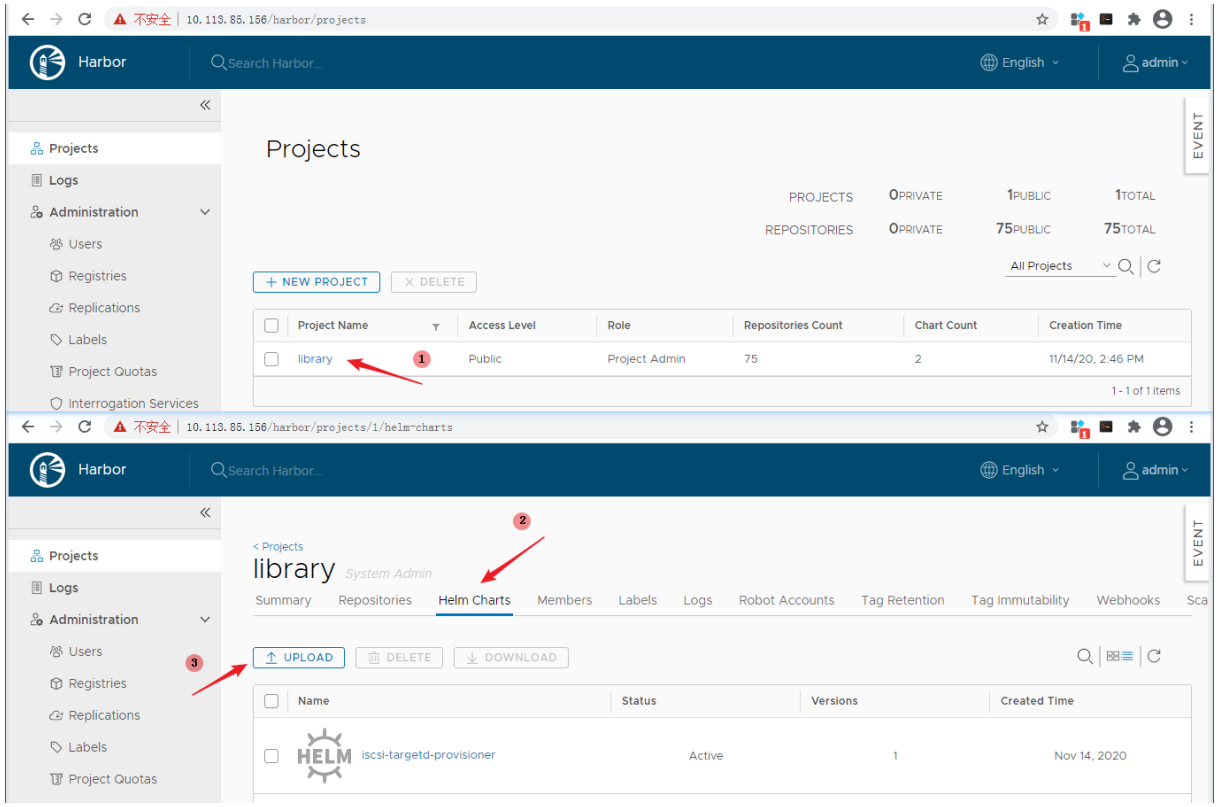
1. Upload the helm chart package to the built-in registry

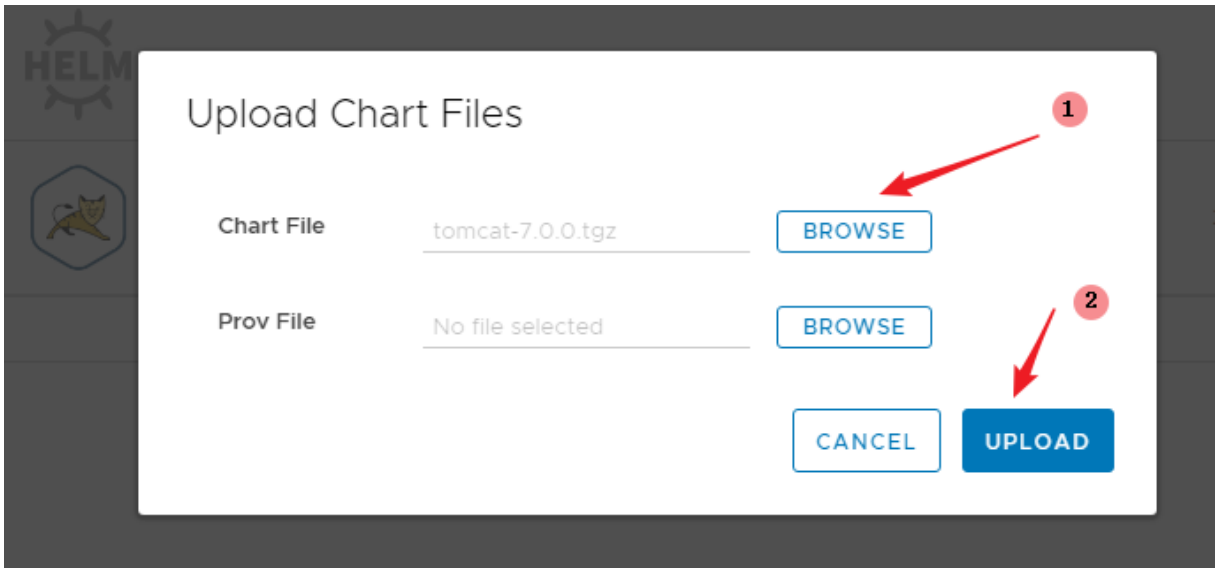
Step 1: Open HARBOR VIP in the browser and log in using your username and password[15].



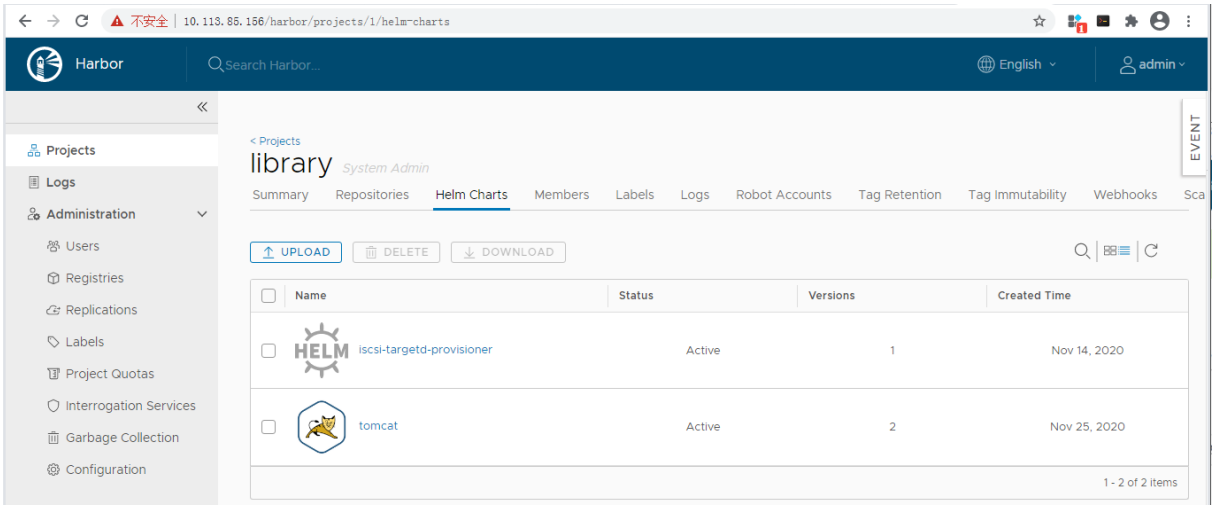
Step 2: Select a project, click "Helm Charts" and then "UPLOAD", and select tomcat-7.0.0.tgz.

[15] The default username and password of Harbor are: a****/H********4*. Contact relevant staff for the password.





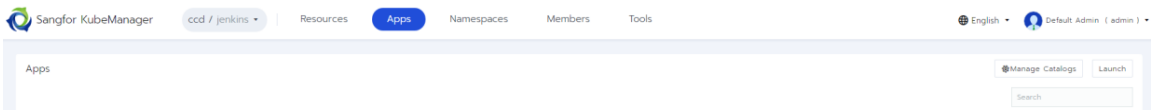
Step 3: A message appears to indicate that the application has been uploaded.



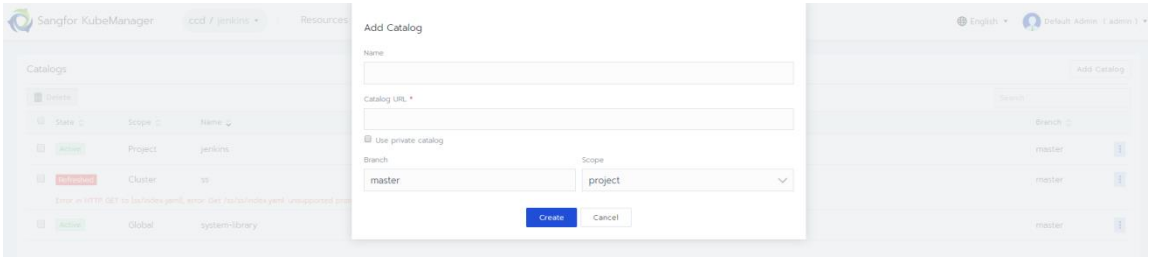
8.2.2 Deploying App Store's Helm Application

1. Add the built-in app store.

   Step 1: Select the "Apps" page of the project and click "Manage Catalogs" to go to the store setting at the project level. Click "Add Catalog".



   Step 2: Click "Add Catalog".

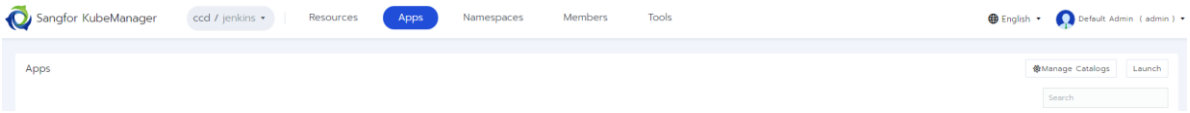


   (a) Configure the name.

   (b) Configure app store address as the built-in registry's address for storing the project for helm charts[16]. The format is **https://built-in registry IP/chartrepo/uploaded project name**.

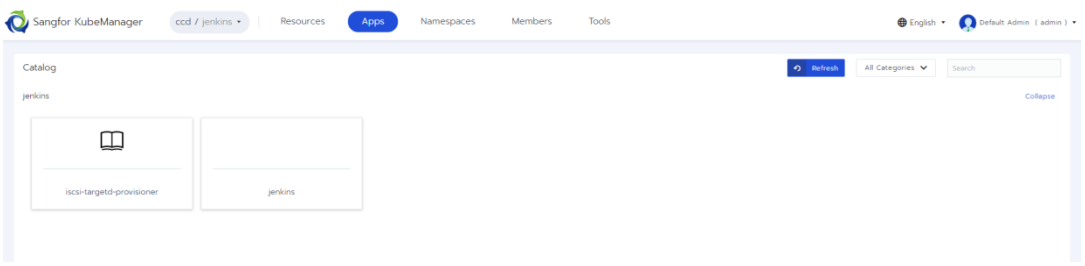
2. Deploy applications provided by the app store.

   Step 1: Click "App Store" of the project to go to the application list. Then click "Launch".



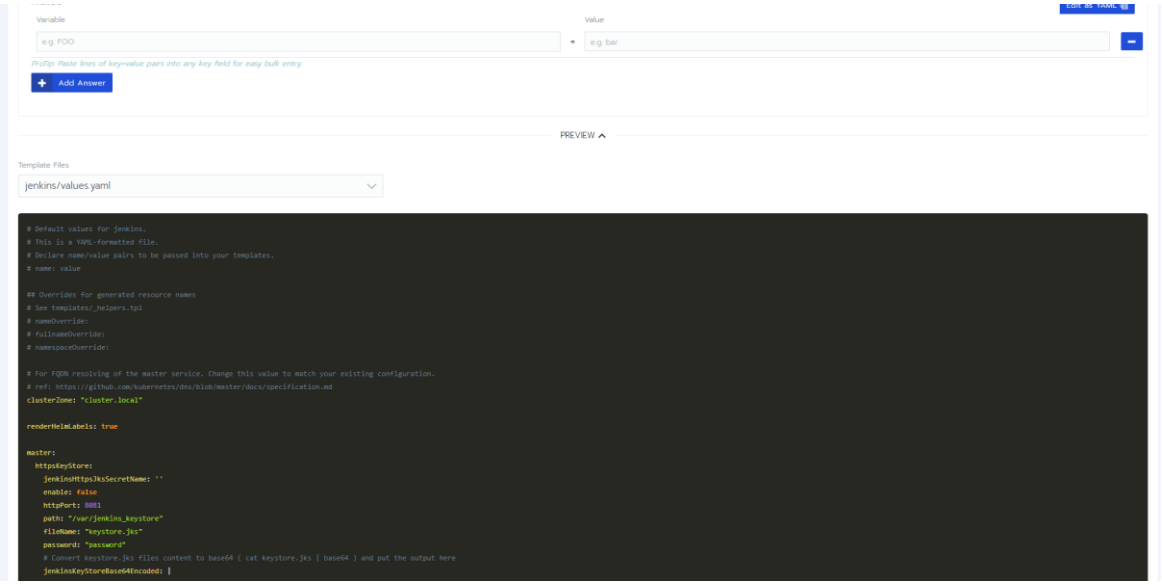
   Step 2: Now you can see the uploaded tomcat helm chart application. Click the tomcat application for deployment.

[16] For details about the method to upload helm charts via Harbor, see Managing Helm Charts on the official site.



3. Preparation of images

   (a) Click "PREVIEW" and find the image on which the tomcat package relies. Download the application image and then push it to the built-in registry. This frees you from reliance on the Internet for application deployment.



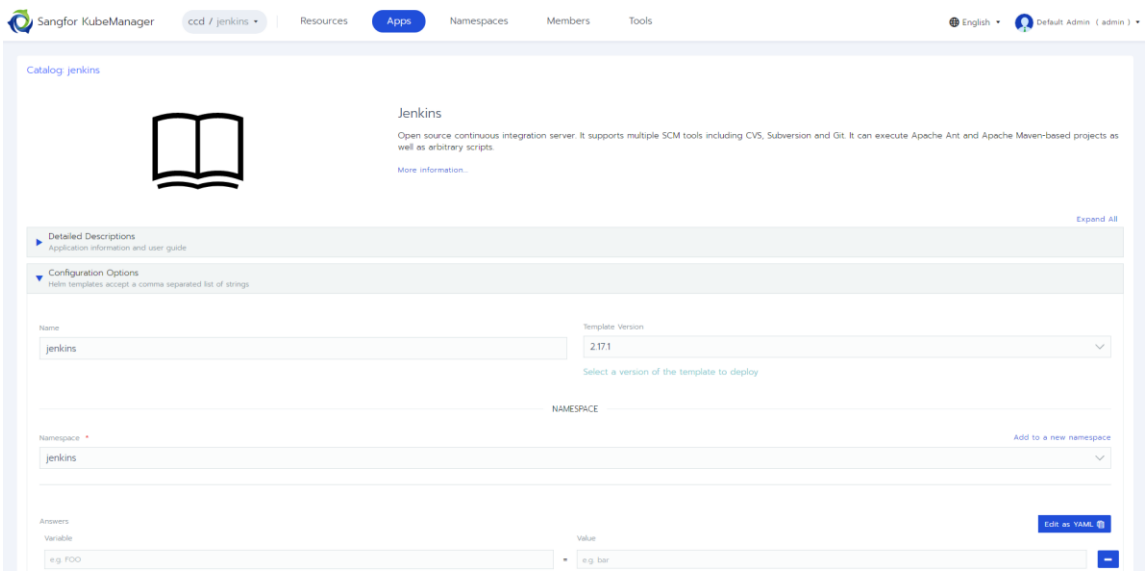
   (b) Download the image using a Linux host that can be connected to the Internet.

   docker pull bitnami/tomcat:9.0.39-debian-10-r26

   (c) Upload the container image on which the helm relies into the built-in registry.

   docker tag bitnami/tomcat:9.0.39-debian-10-r26 \ 10.113.85.156/library/tomcat:9.0.39-debian-10-r26 docker push 10.113.85.156/library/tomcat:9.0.39-debian-10-r26

4. Application settings. The helm configuration option is available on the interface, as shown in the figure.



(a) A detailed description, which means that the **helm** command line reads the chart's description "helm show readme". Read the description carefully to know how to configure the corresponding helm chart application.

(b) Configure the application name

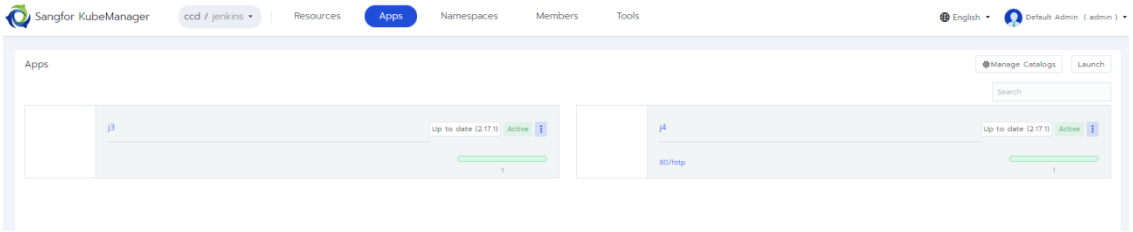
(c) If the helm chart is available in multiple versions, select "Version Deployment" -> "App Store". There will be a prompt for an update on the interface if a new helm chart version is available.

(d) Select the namespace for deploying the application.

(e) Answers configuration, which is a customization application for configuring helm's key value (**--set key=vale**).

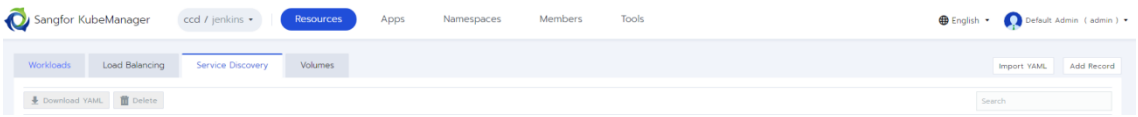
8.3 Verification

Step 1: Click "App Store" and check whether the application has been deployed.

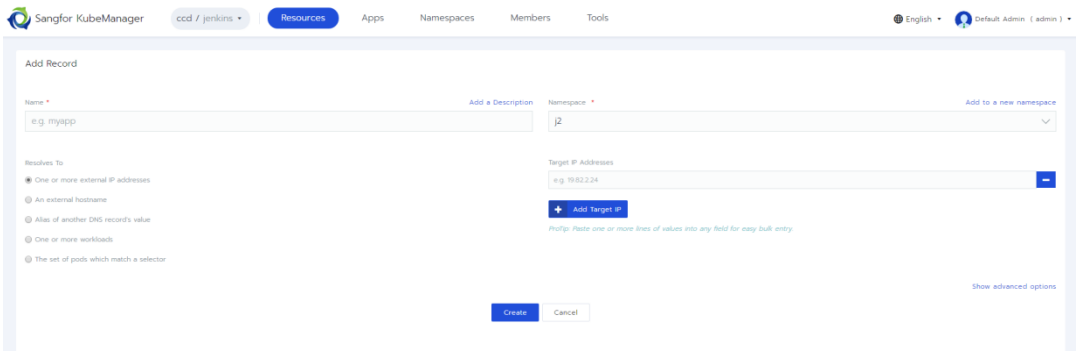


Step 2: Configure L4 load balancing to publish the service access application.

Click the Resource page of the project and then "Service Discovery" to add DNS records.

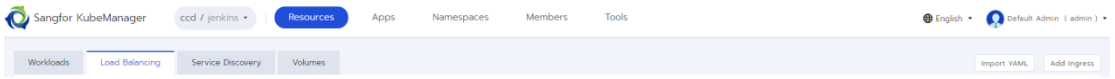


Configure DNS records.

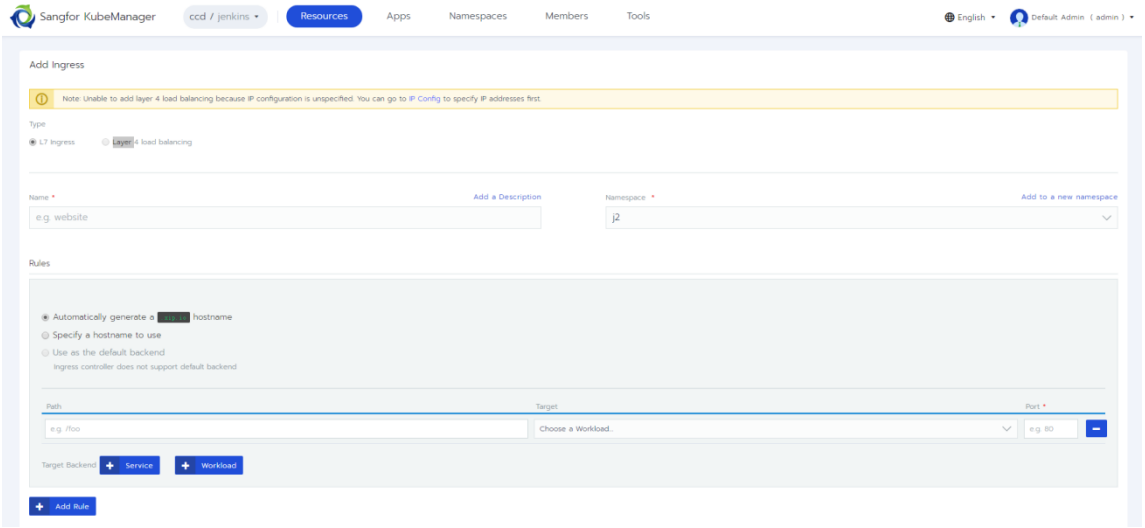


1. Configure the DNS name.
2. Namespace: Select the namespace where the tomcat is in.
3. Select to resolve to "Workload".
4. Select the tomcat workload
5. Select "Cluster IP" for service type
6. Configure the port for the service
   (a) Port Name
   (b) Service Port: port of the service IP address
   (c) Dst Port: monitoring port for the container workload; the tomcat monitors the 8080 port

To configure load balancing, click the "Resource" page of the project. Click "Load Balancing" to add the rule.



Configure service publishing for L4 workloads.



1. Rule Type: L4
2. Configure the name
3. Namespace: Select the namespace where the tomcat is in.
4. Configure the rule

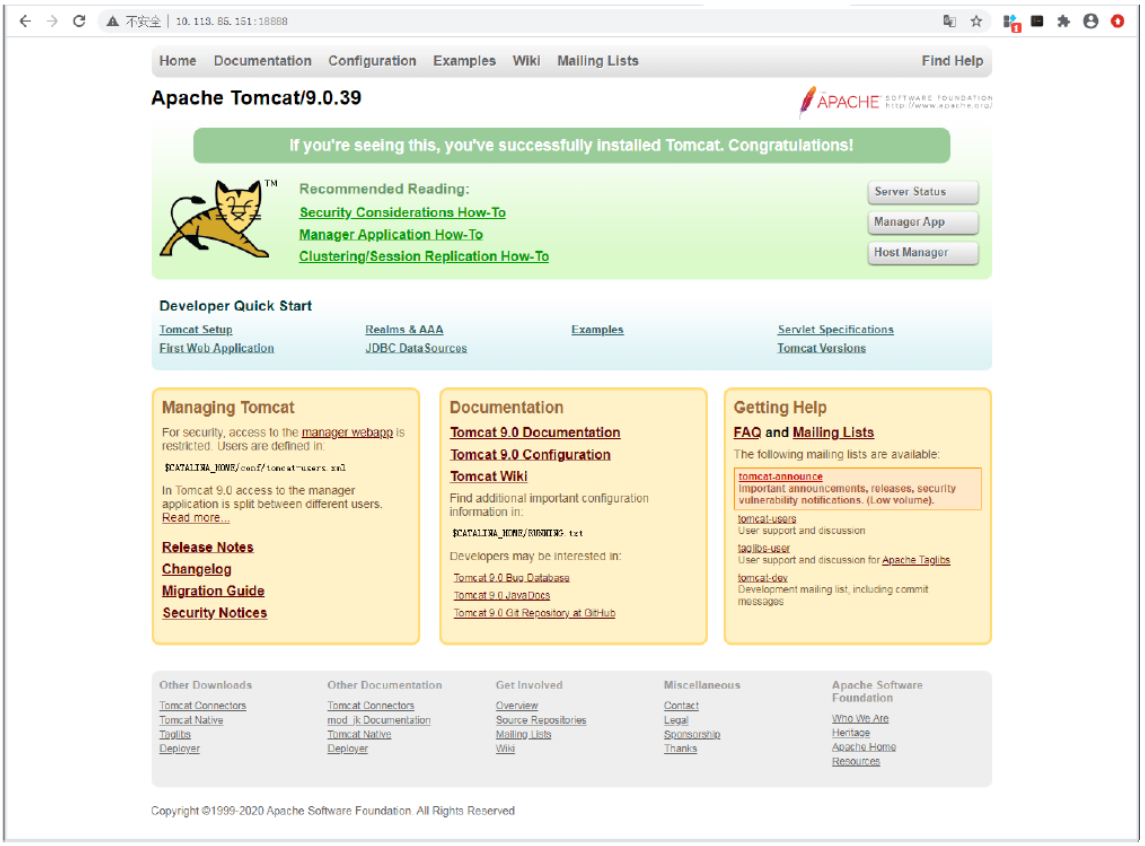
(a) Type: TCP

(b) Network port IP address (Monitoring Port): 18888

(c) Service: tomcat-workload

(d) Container Port: port 80 of the tomcat-workload service

Access the application to check whether the service has been published.



8.4 Questions for Discussion

1. Can helm chart packages be uploaded by using a command line, other than the built-in registry's upload function?
2. How can an application be deployed by using the **helm** command line? It is feasible for the command line to only use the helm template rendering function and then import the application into K8s?
3. How is a helm-charts package made?
4. Which storage classes in a K8s cluster will be used by default if a chart package is to use storage classes? Is the setting of the default storage class available?

9 Experiment 8: Use of App Store

9.1 Introduction

9.1.1 About the Experiment

In this experiment, Jenkins is installed and configured through the App Store to realize continuous integration of software development. The KubeManager CLI is used in Jenkins pipeline for continuous delivery.

9.1.2 Environment Specification

- All nodes in KubeManager must be able to access the Internet.

9.1.3 Purpose

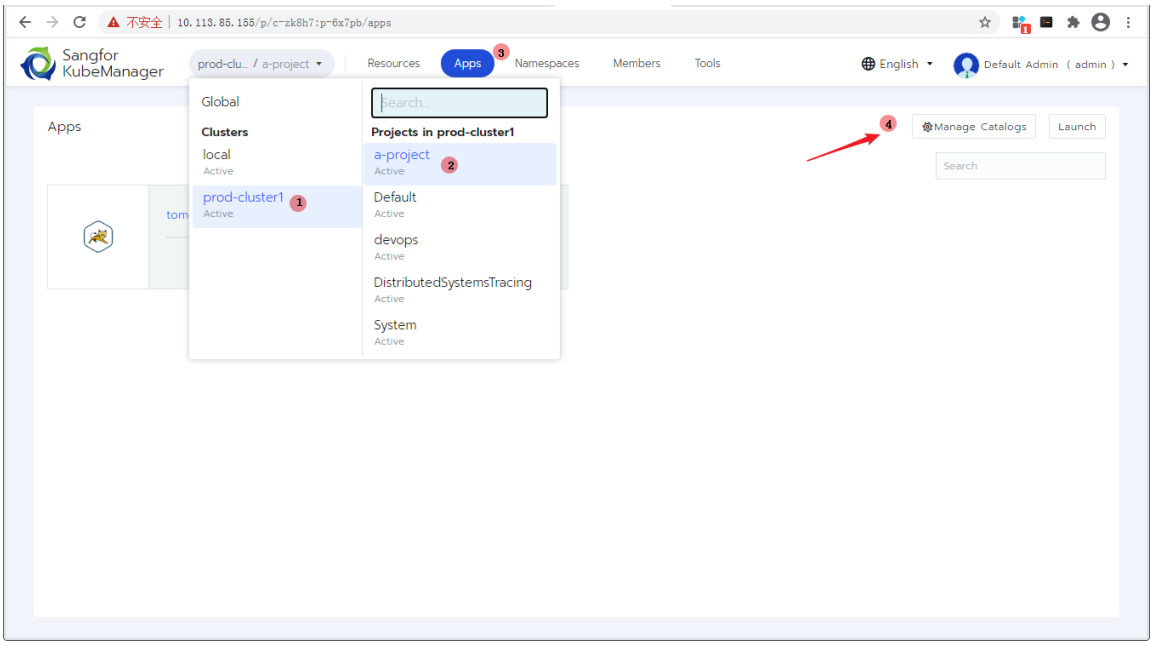
- Master the use of the app store.
- Master using Jenkins for app store deployments.
- Master configuring Jenkins to use Kubernetes resources.
- Master using Docker to build images in the Jenkins pipeline.
- Master using Docker to push images to the built-in image repository in Jenkins pipeline.
- Master using KubeManager CLI program in Jenkins pipeline to deploy images to Kubernetes environment for continuous delivery.
- Basic understanding of Jenkins declarative pipeline syntax.

9.2 Steps

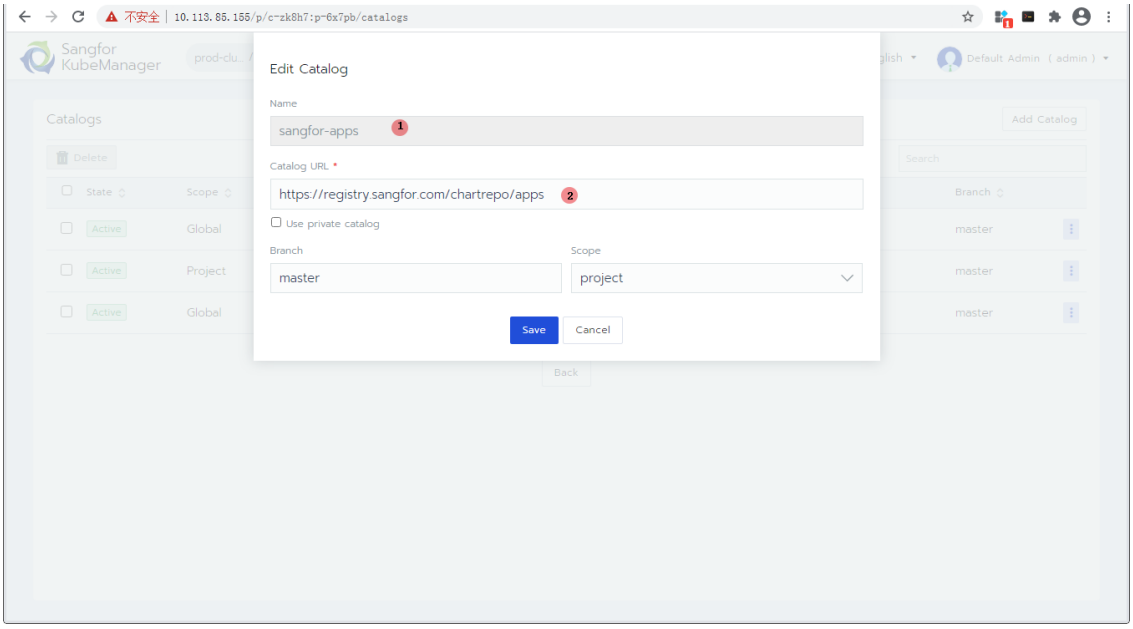
9.2.1 Install Jenkins in the App Store

1. Add the built-in app store.

   Step 1: Select the "Apps" page of the project and click "Manage Catalogs" to go to the store setting at the project level. Click "Add Catalog".



   Step 2: Click "Add Catalog".

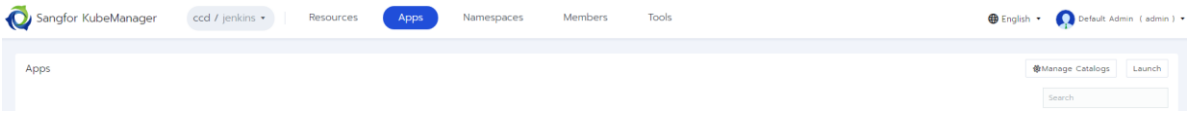


(a) Configure the name which is **sangfor-paas**.

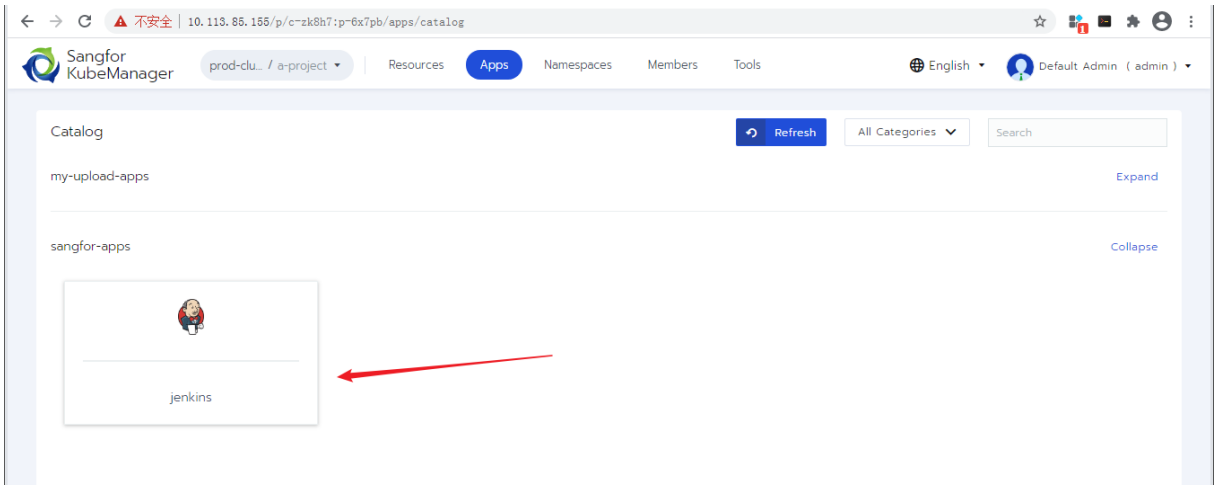
(b) Configure app store address as the built-in registry's address for storing the project for helm charts[17]. The format is **https://registry.sangfor.com/chartrepo/apps**.

2. Deploy the app provided by the App Store.

Step 1: Click "App Store" of the project to go to the application list. Then click "Launch".

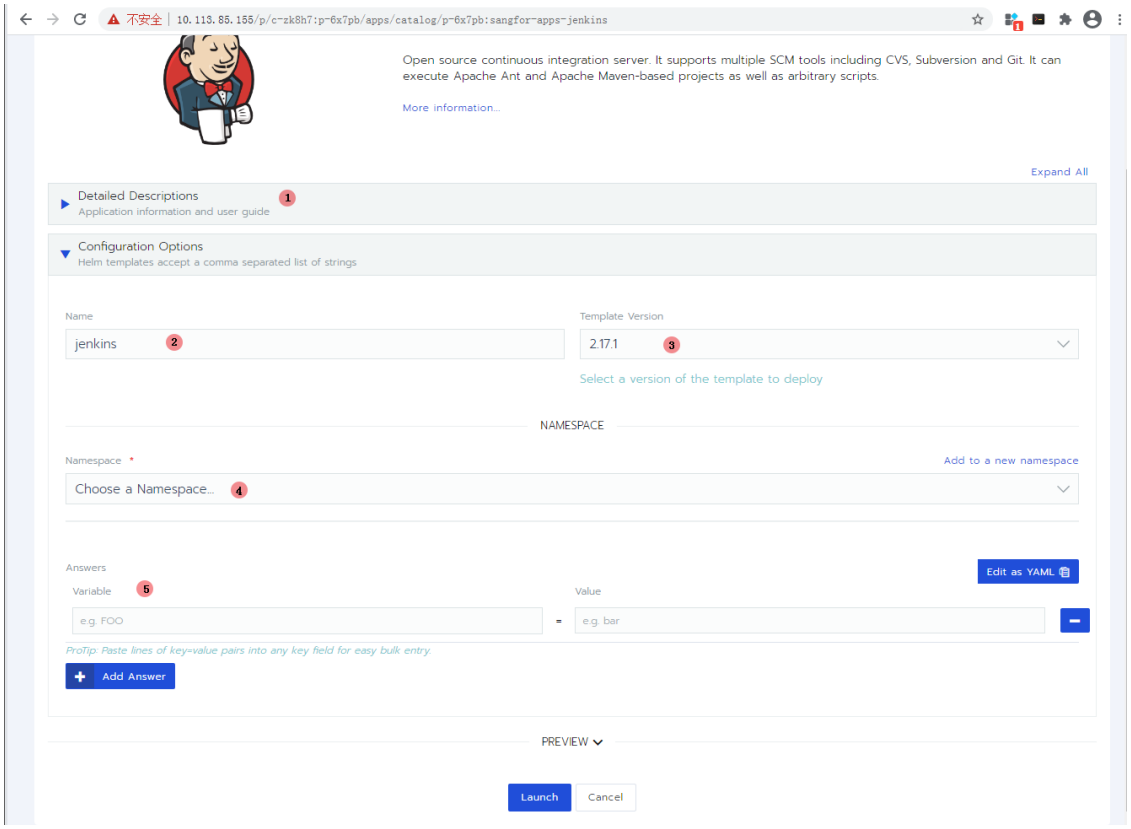


Step 2: Click "jenkins" for deployment.



Step 3: Application configuration.

[17] For details about the method to upload helm charts via Harbor, see Managing Helm Charts on the official site.



(a) Configure the application name.

(b) If the helm chart is available in multiple versions, select "Version Deployment" -> "App Store". There will be a prompt for an update on the interface if a new helm chart version is available.

(c) Select the namespace in which the application is deployed.

(d) Configure blank in Answers, which is a customization application for configuring helm's key value (--set key=vale).

Notice:

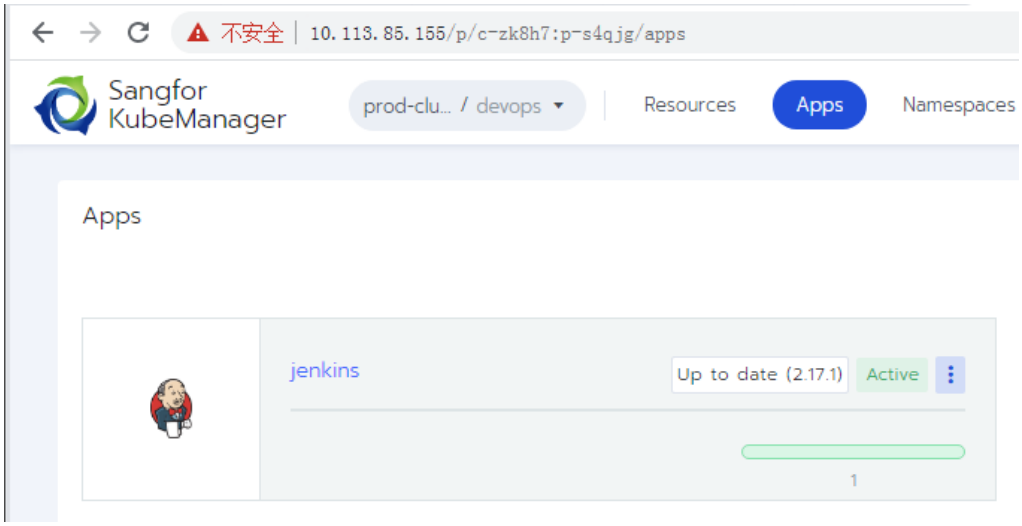
i:For an intranet environment, you need to push all dependent images to the built-in image repository. And then use Answers configuration to change the image address to the built-in image address.

ii:The Jenkins persistent store is not configured by default for production environments that need to be configured.

iii:Jenkins account security settings should be configured in production environment.

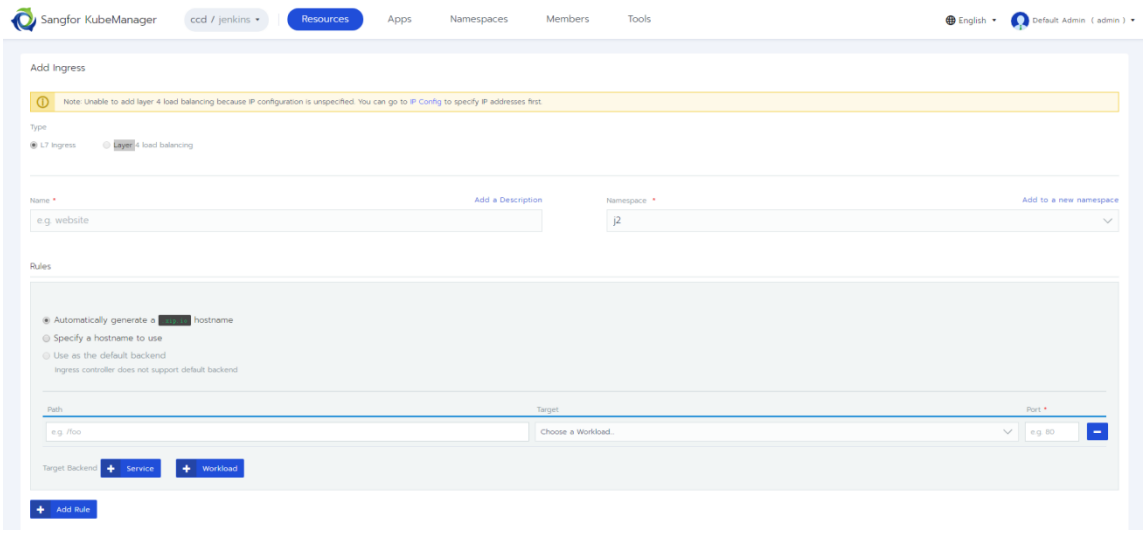
Step 4: Click "Create".

Step 5: Ensure that the application is Active.



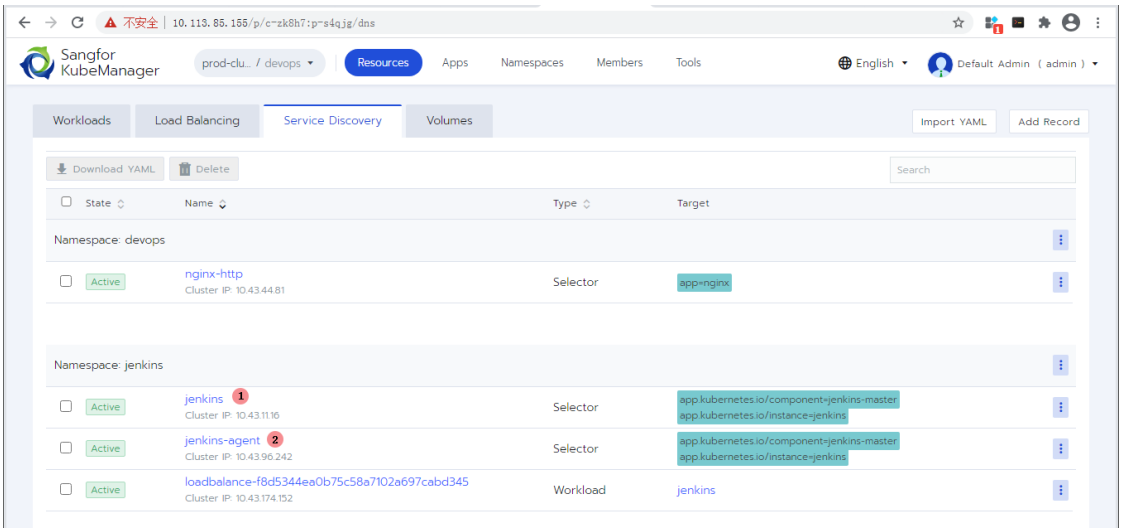
9.2.1 Install Jenkins in the App Store

1. Configure L4 or L7 of load balancing.



As an example of L4 load configuration, you should configure port 18080 access from the network exit IP to port 8080 of Jenkins workload.

2. Note down Jenkins and Jenkins-Agent service names.

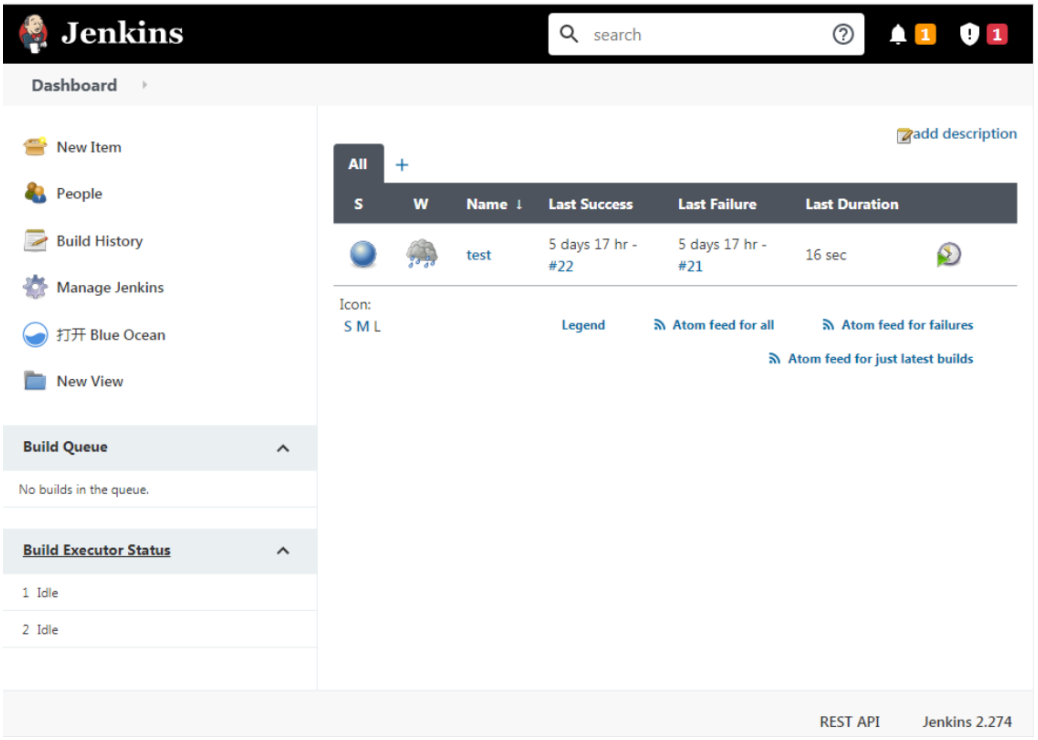


(a) Record the internal load domain name as Jenkins.

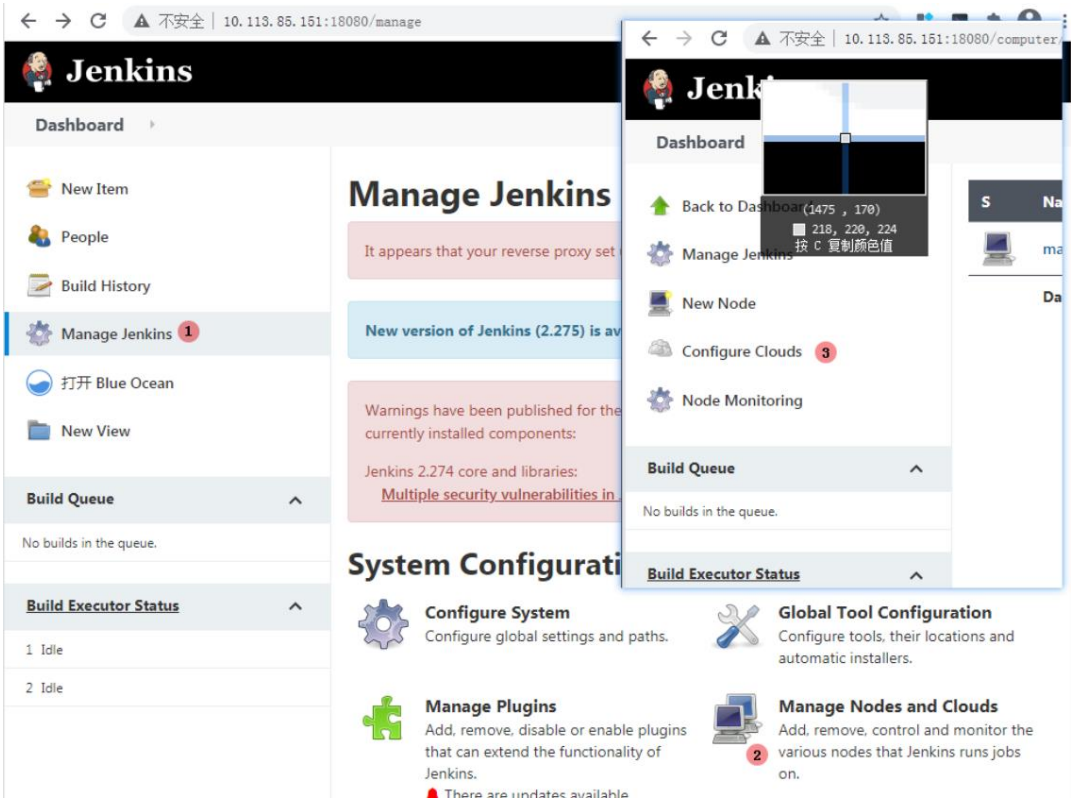
(b) Record the internal load domain name as Jenkins-agent.

9.2.3 Configure Jenkins

1. Visit Jenkins.



2. Configure Clouds.

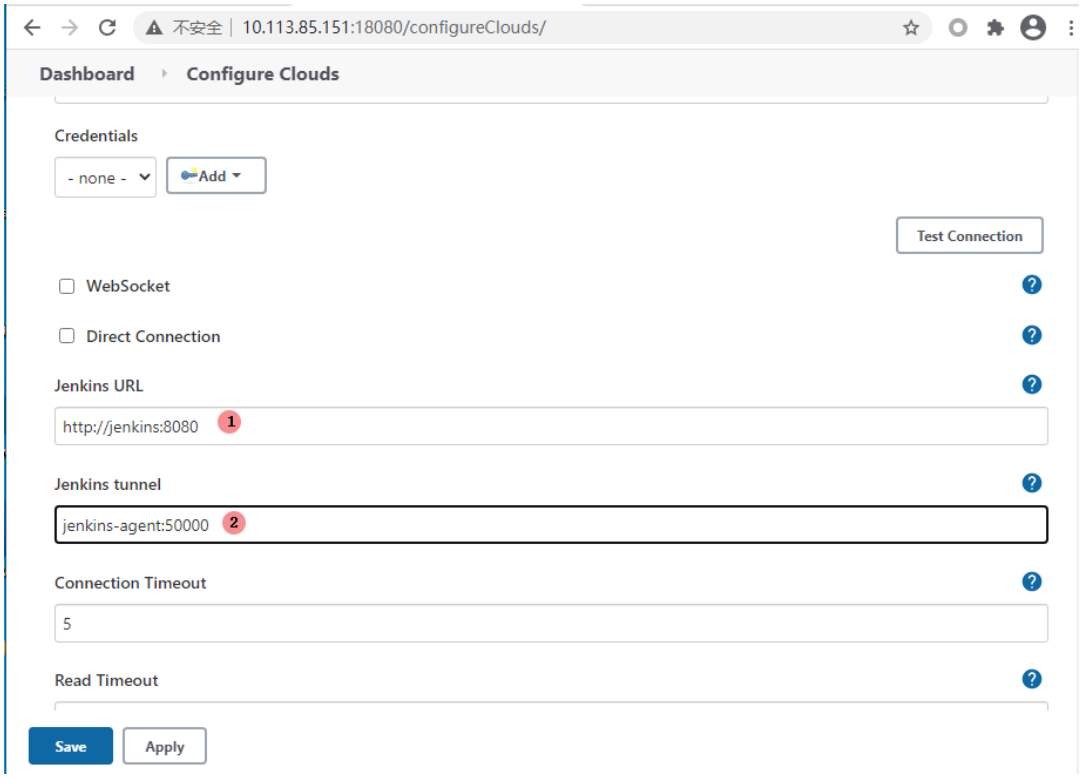


(a) Click “Manage Jenkins”.

(b) Click “Manage Nodes and Clouds”.

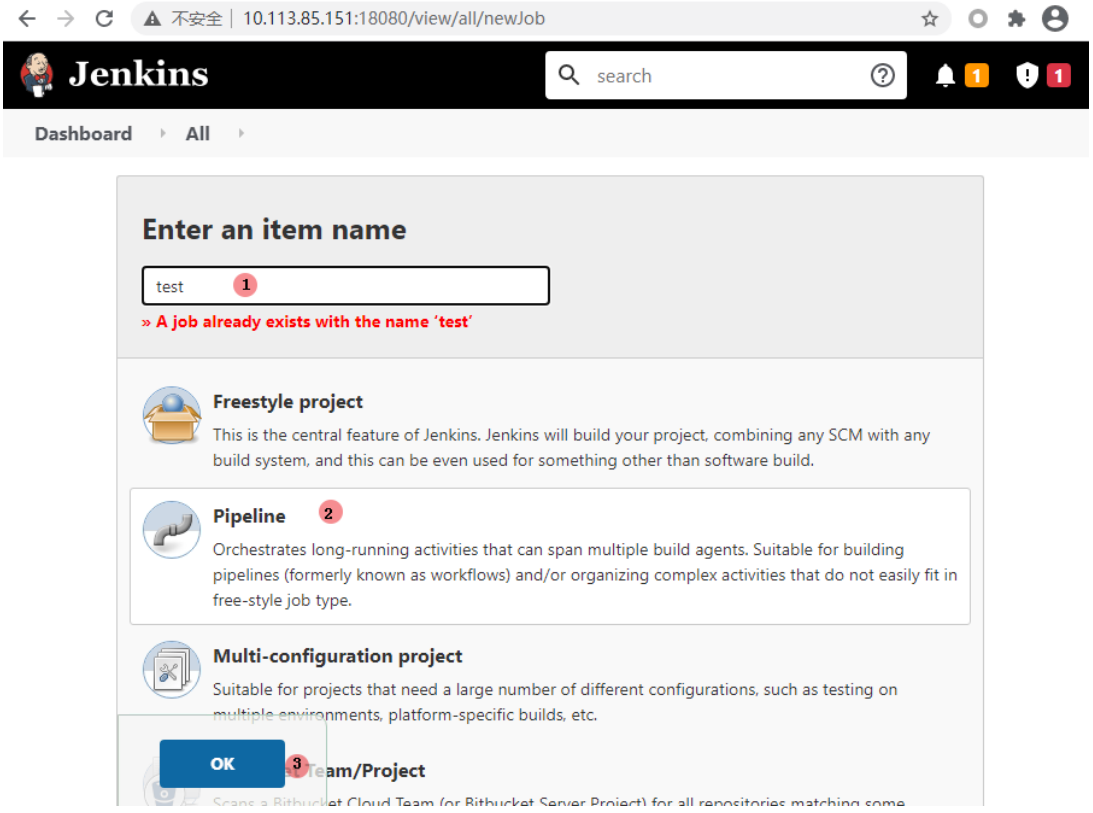
(c) Click “Configure Clouds” to add kubernetes cluster.

3. Configure and save Jenkins address and Jenkins channel.

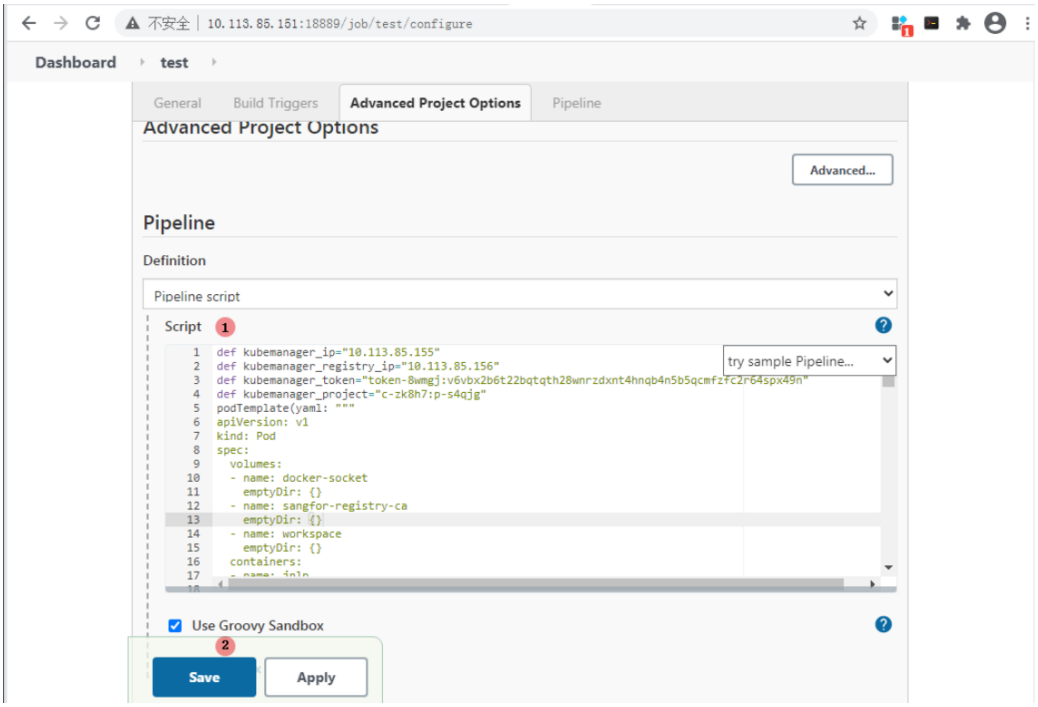


9.2.4 Create a pipeline in Jenkins

1. New Item.



2. Definition Pipeline.



Refer to pipeline configuration in kubernetes:

```
def variable=foo
podTemplate(...) {
    node(POD_LABEL) {
        //some steps
    }
}
```

Sample DevOps code:

```
// define const
// kubemanager IP address
def kubemanager_ip="10.113.85.155"
// kubemanager registry ip address
def kubemanager_registry_ip="10.113.85.156"
// kubemanager api token
def kubemanager_token="token-8wmgj:v6vbx2b6t22bqtqth28wnrzdxnt4hnqb4n5b5qcmfzfc2r64spx49n"
// kubemanager project name
def kubemanager_project="c-zk8h7:p-s4qjg"
podTemplate(yaml: """
apiVersion: v1
kind: Pod
spec:
  volumes:
  - name: docker-socket
    emptyDir: {}
  - name: sangfor-registry-ca
    emptyDir: {}
  - name: workspace
    emptyDir: {}
  containers:
  - name: jnlp
    image: registry.sangfor.com/apps/inbound-agent:4.6-1
    args: ['\$(JENKINS_SECRET)', '\$(JENKINS_NAME)']
    volumeMounts:
    - name: workspace
      mountPath: /workspace
  - name: docker
    image: registry.sangfor.com/apps/docker:20.10.2
    command:
    - cat
    tty: true
    volumeMounts:
    - name: docker-socket
      mountPath: /var/run
    - name: workspace
      mountPath: /workspace
  - name: docker-daemon
```

```
    image: registry.sangfor.com/apps/docker:20.10.2-dind
    securityContext:
      privileged: true
    volumeMounts:
    - name: docker-socket
      mountPath: /var/run
    - name: sangfor-registry-ca
      mountPath: /etc/docker/certs.d
    - name: workspace
      mountPath: /workspace
  - name: cd-tools
    image: registry.sangfor.com/apps/kubemanager-cd:sangfor
    imagePullPolicy: Always
    command:
    - cat
    tty: true
    volumeMounts:
    - name: sangfor-registry-ca
      mountPath: /etc/docker/certs.d
    - name: workspace
      mountPath: /workspace
""") {
    node(POD_LABEL) {
        stage('Clone') {
            container('jnlp') {
                sh """
            git clone -b modify-repo-url https://github.com/ctlaltlaltc/docker-nginx.git /workspace/docker-nginx
            """
            }
        }
        stage('Setting') {
            container('cd-tools') {
                sh """
            # Setting sangfor registry private certificate
            # Setting sangfor kubemanager registry ip
            mkdir -p /etc/docker/certs.d/$kubemanager_registry_ip
            wget --no-check-certificate -q -O- https://$kubemanager_registry_ip/api/systeminfo/getcert >\
            /etc/docker/certs.d/$kubemanager_registry_ip/ca.crt
            """

            }
        }
        stage('Build') {
            container('docker') {
                sh """
            docker version
            export DOCKER_BUILDKIT=1
            docker build --progress plain -t $kubemanager_registry_ip/library/nginx-devops:latest \
            /workspace/docker-nginx/stable/alpine
            """
            }
        }
        stage('Publish') {
            container('docker') {
                sh """
            docker login -u admin -p Harbor12345 $kubemanager_registry_ip
            docker push $kubemanager_registry_ip/library/nginx-devops:latest
            """

            }
        }
        stage('Deploy') {
            container('cd-tools'){
                sh """
            kubemanager login https://$kubemanager_ip --skip-verify --token $kubemanager_token --context $kubemanager_project
            if ! kubemanager kubectl get namespace devops; then
                kubemanager kubectl create namespace devops
            fi
            # rollout update
            if ! kubemanager kubectl -n devops get deploy nginx; then
              kubemanager kubectl -n devops create deployment nginx --image=$kubemanager_registry_ip/library/nginx-devops:latest
            else
              kubemanager kubectl -n devops rollout restart deployment/nginx
            fi
            # expose service
            if ! kubemanager kubectl -n devops get service nginx-http; then
              kubemanager kubectl -n devops expose deployment nginx --port=80 --target-port=80 --name=nginx-http
            fi
            kubemanager kubectl -n devops  wait --for=condition=available --timeout=600s deployment/nginx
            """
            }
        }
        stage('Test') {
            container('cd-tools') {
                sh """
            true
            """
            }
        }
        stage('Teardown') {
            container('jnlp') {
                sh """
            true
            """
            }
        }
    }
}
```

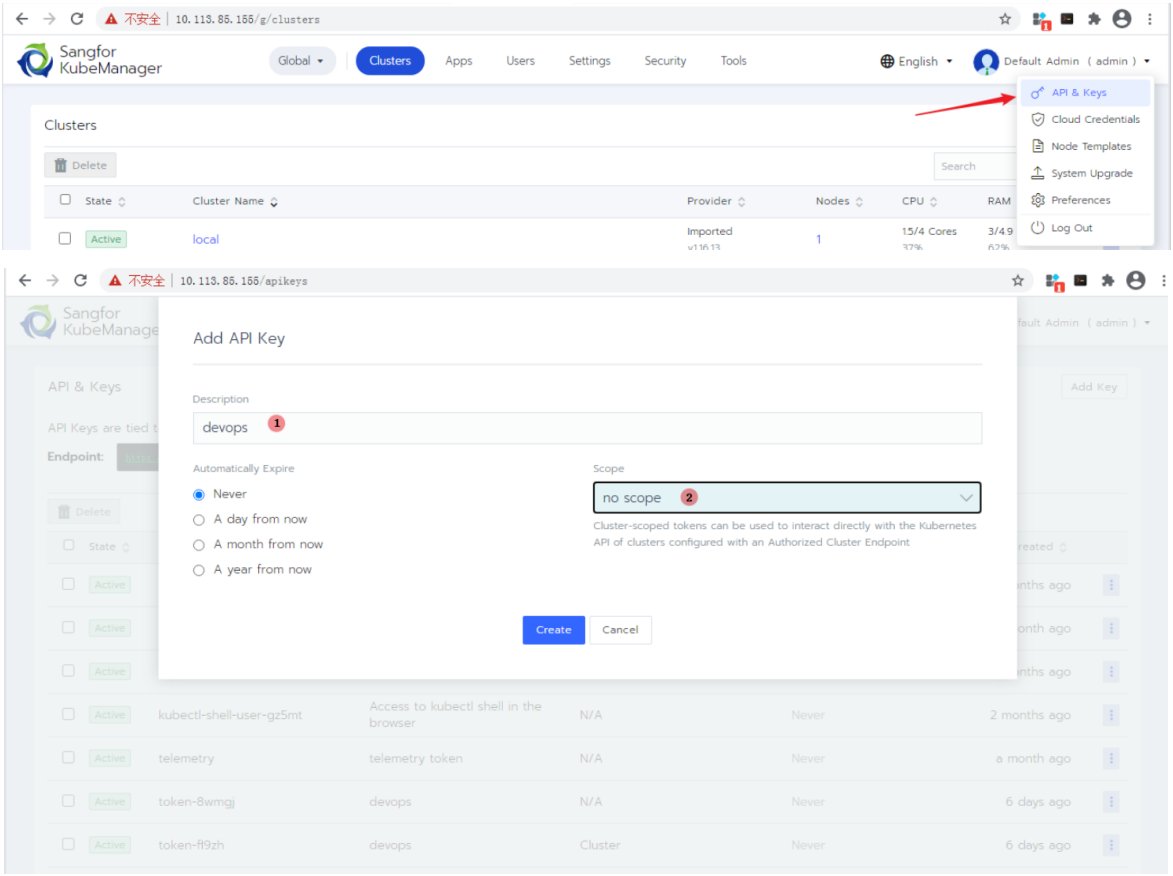
Pipeline description:

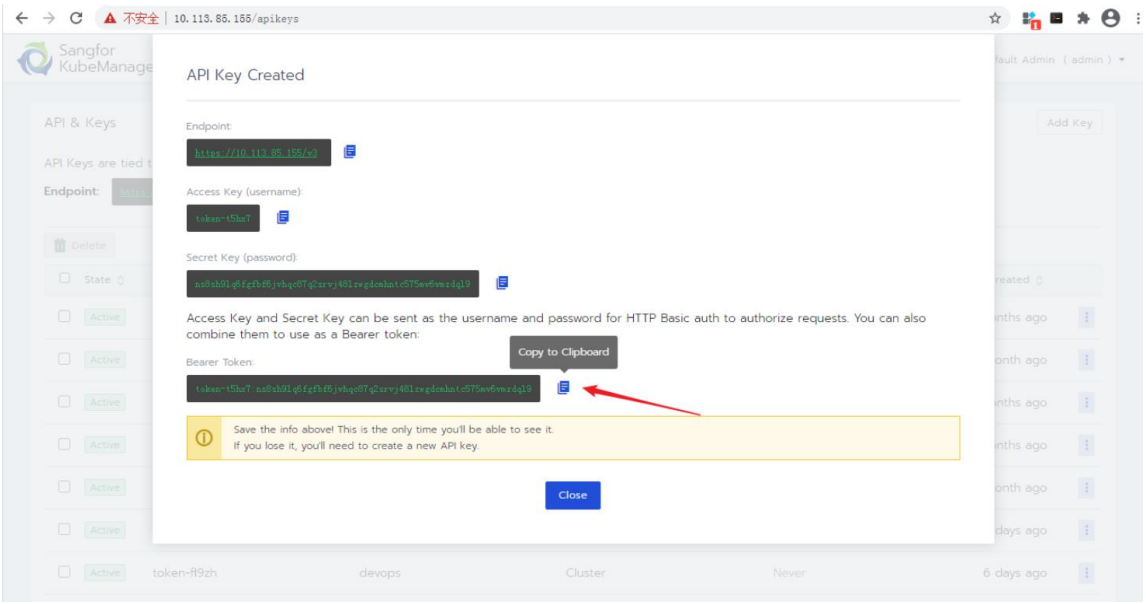
Define constants:

```
def kubemanager_ip="10.113.85.155"
def kubemanager_registry_ip="10.113.85.156"
def kubemanager_token="token-8wmgj:v6vbx2b6t22bqtqth28wnrzdxnt4hnqb4n5
↪ b5qcmfzfc2r64spx49n"
def kubemanager_project="c-zk8h7:p-s4qjg"
```

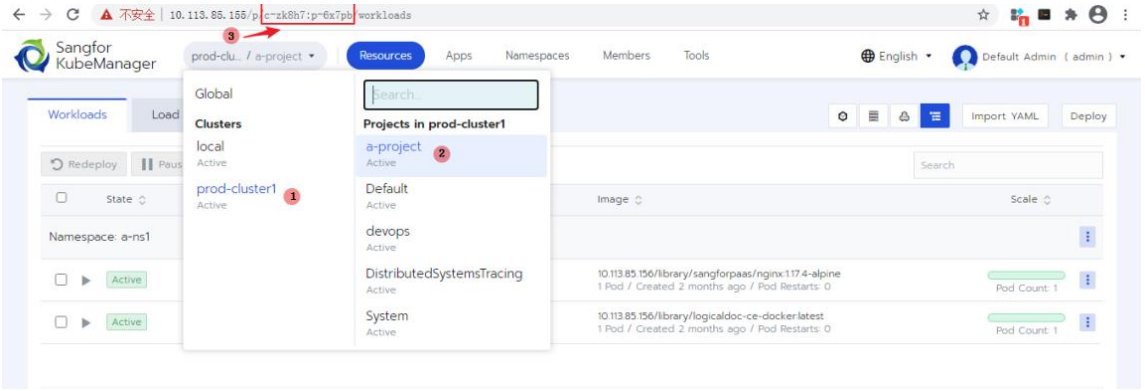
Constant	Description
kubemanager_ip	ip address of kubemanager
kubemanager_registry_ip	ip address of kubemanager built-in image repository
kubemanager_token	api token created in kubemanager interface
kubemanager_project	project name of kubemanager pipeline

Create api token:

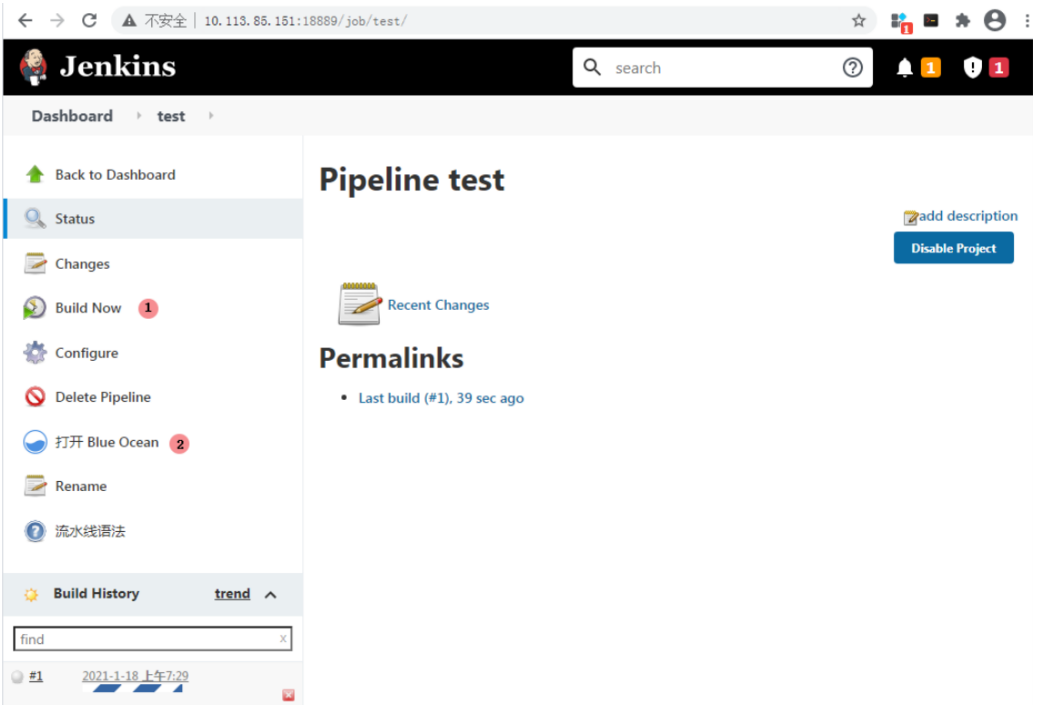




Achieve the project name of pipeline:

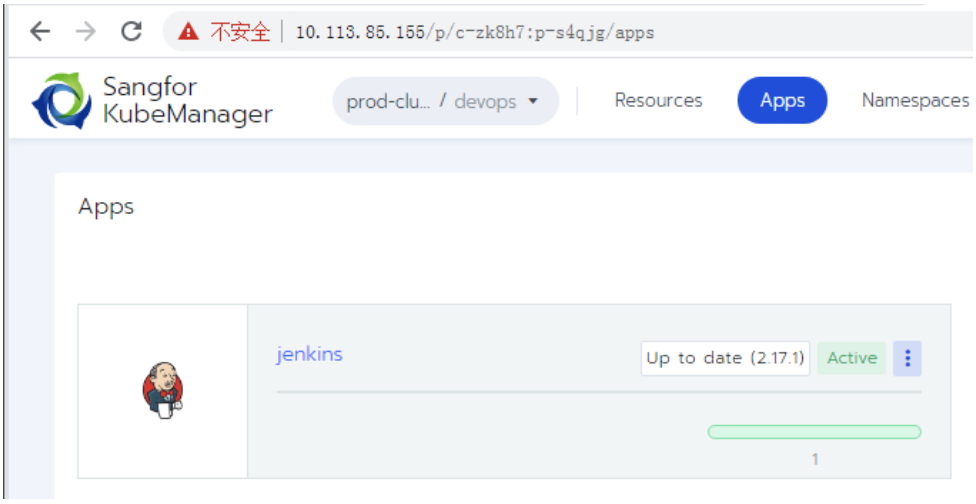


3. Build and Open Blue Ocean.

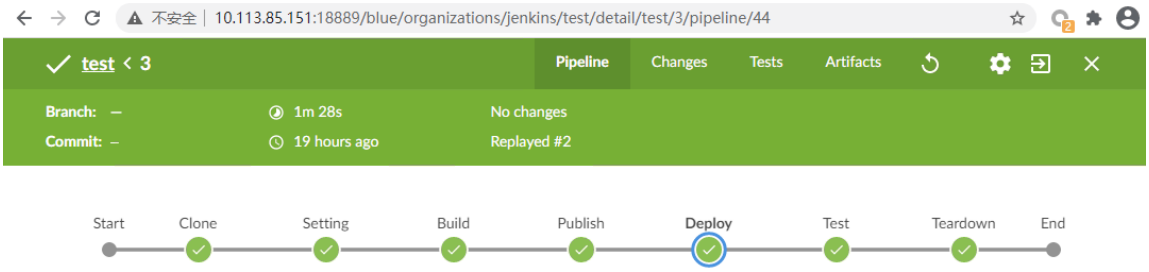


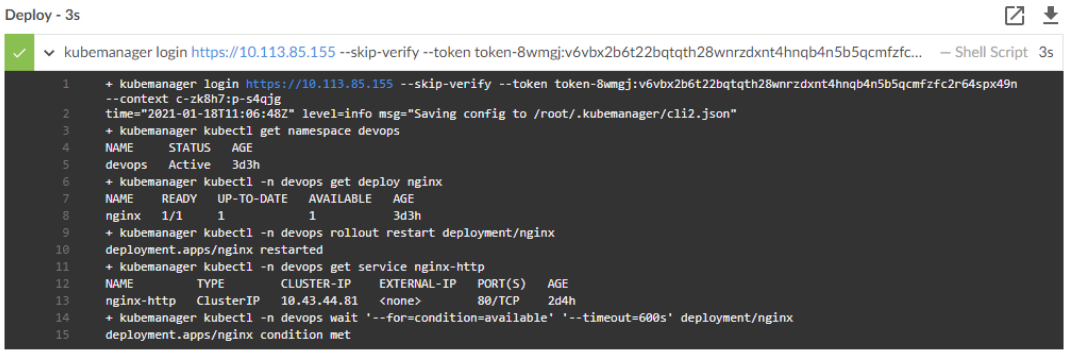
9.3 Verification

1. Click "App Store" and check whether jenkins has been deployed.



2. View the pipeline.





9.4 Questions for Discussion

1. How do Jenkins configure storage for data persistence?
2. Does Jenkins pipeline support customized images building?
3. Does the "Workload" provided on the interface support the deployment of all K8s applications?
4. What programming language does Jenkins use to implement the declarative pipeline syntax?
5. Does PodTemplate follow the K8S configuration specification in the pipeline?What can

be configured?

6. How is persistence implemented in Jenkins? Use Minio or Jenkins Built-in Artifact?