Video Security Monitoring R5 API Document

Introduction

These APIs provide services for manipulating ote-stack platform on AI Edge Blueprint.

API Definitions

otestack v1

User Management

1.Login

POST /v1/authorization

Description

An access token, which contains user information and expiry time, will be returned by presenting username and password. The client should includes the token in the authorization header of HTTP request to access subsequent request until the token is expired. During the validity period (1 hour), the new token with new expiry time can be obtained through the Update token API with current token.

Parameters

Name	Type	Required	Default	Description
user	string	yes		user name
password	string	yes		user password

Response codes

Code	Description
200	Successful
500	Unexpected internal errors
422	Parameter validation error

Example

Example

```
POST /v1/authorization
Body:
{
        "user": "u1",
        "password": "xxx"
}
Response:
{
        "token": "eyxxxxxxxxxxxxx",
        "code": 200,
        "message": "success"
}
```

2.Update Token

PUT /v1/authorization

Description

Returns a new authorization token with new expiry time.

Parameters

Name	Type	Required	Default	Description
Authorization	string	yes		token string in Authorization Header

Response codes

Code	Description
200	Successful
500	Unexpected internal errors
422	Parameter validation error

Example

Example

```
PUT /v1/authorization
Header: Authorization: eyxxxxxxxxxxxxx
Response: {
        "token": "eyyxxxxxxxxxxx2",
        "code": 200,
        "message": "success"
}
```

3.Create User

POST /v1/user

Description

Create a user without access token authorization. This user will be available after system administrator approval.

Parameters

Name	Type	Required	Default	Description
user	string	yes		User name
password	string	yes		Password should be 8-20 characters long with at least 1 uppercase, 1 lowercase and 1 number.
phone	string	yes		Phone number

isAdmin	bool	yes		true: general administrator, false: general user
realName	string	yes		Real name

Response code

Code	Description
200	Success
500	Unexpected internal errors
422	Parameter validation error

Example

Example

```
POST /v1/user
body:
{
        "user": "u1",
        "password": "xxX@1234",
        "phone": "12345678910",
        "realName": "hello"
        "isAdmin": true
}
Response:
{
        "code": 200,
        "message": "success"
}
```

4.Update Password

PUT /v1/user

Description

Change the password of the logged-in user.

Parameters

Name	Type	Required	Default	Description
oldPassword	string	yes		old password
password	string	yes		new password

Response code

Code	Description
200	Success
500	Unexpected internal errors
422	Parameter validation error

Example

Example

```
PUT /v1/user
body:
{
        "oldPassword": "xxx0",
        "password": "xxx"
}
Response:
{
        "code": 200,
        "message": "success"
}
```

5.Delete User

DELETE /v1/admin/user/[userid]

Description

This can only be done by general administrator and system administrator. General administrator can delete user in own business namespace while system administrator can delete all users.

Parameters

Name	type	Required	Default	Description
userid	uint64	yes		User id that needs to be deleted.

Response code

Code	Description
200	Success
500	Unexpected internal errors
422	Parameter validation error

Example

Example

```
DELETE /v1/admin/user/1
Response:
{
        "code": 200,
        "message": "success"
}
```

6.Reset Password

PUT /v1/admin/user/[userId]/password

Description

This can only be done by general administrator and system administrator. General administrator can update user in own business namespace while system administrator can update all users.

Parameters

Name	Type	Required	Description
userId	uint64	yes	UserId that needs to update. Located in url path.
password	string	yes	New password that will be updated.Located in body.

Response code

Code	Description
200	Success
500	Unexpected internal errors
422	Parameter validation error

Example

Example

```
PUT /v1/admin/user/1/password
Body:
{
        "password": "xxx"
}
Response:
{
        "code": 200,
        "message": "success"
}
```

7.Get User List

GET /v1/admin/user

Description

This can only be done by general administrator and system administrator. General administrator can get users in own business namespace while system administrator can get all users.

Parameters

Name	Type	Required	Default	Description
orderBy	string	no	`id`	orderBy allows sorting by id and user_name.
order	string	no	asc	sort order: asc, desc
page	int	no	1	page number
pageSize	int	no	10	page size

Response code

Code	Description
200	Success
500	Unexpected internal errors
422	Parameter validation error

Response Parameters

Name	Type	Description
`id`	int	User id
`user`	string	User name

status	int	Status 0: Pending 1: Approved 2: Not Approved 3: Forbidden
realName	string	Real name
phone	string	Phone number
total	int	Total number of users
createTime	int	Create time of user
updateTime	int	Update time of user

Example

Example

```
GET /v1/admin/user
Response
{
        "data":
        [
        {"id":1, "user":"user1", "status":3, "realName":"user1", "phone":"18923441163", "createTime":
1551758321, "updateTime":1551758321 },
        {"id":2, "user":"user2", "status":3, "realName":"user2", "phone":"18923441123", "createTime":
1551758321, "updateTime":1551758321 }
        ],
        "total": 2,
        "code": 200,
        "message": "Success"
}
```

8.Audit User

PUT /v1/admin/user/[userId]

Description

This can only be done by system administrator for auditing new user. The new user will be available to create business namespace after system administrator approval.

Parameters

Name	Type	Required	Description
userId	uint64	yes	UserID that needs to be audited.
status	int	yes	Status 0: Pending 1: Approved 2: Not Approved 3: Forbidden

Response code

Code	Description
200	Success
500	Unexpected internal errors

422	Parameter validation error

Example

Example

```
PUT /v1/admin/user/1
body:
{
        "status": 0
}
Response:
{
        "code": 200,
        "message": "success"
}
```

9.Logout

DELETE /v1/authorization

Description

Logout by deleting the access token.

Response code

Code	Description
200	Success
500	Unexpected internal errors
422	Parameter validation error

Example

Example

```
DELETE /v1/authorizations
Header: Authorization: eyxxxxxxxxxxxxx
Response:
{
        "code": 200,
        "message": "Success"
}
```

10.Get Authorization Information

GET /v1/authorization

Description

Return the authorization information, such as user name, user role.

Response parameters

Name	Type	Description
displayName	string	User name
hasRepoAccount	bool	Whether or not the user has created a harbor account

role	int	User role: 0: register user that waiting for approval 1approved user that do not has business namespace 2general user 3business administrator 4system administrator
privateProject	bool	Flag to indicate if the project is private.
enableIngress	bool	Flag to indicate if the ingress is enabled.
enableAlert	bool	Flag to indicate if the alert information is enabled.
enableAdminAutoDeploy	bool	Allow deployment that admin submits run automatically without audit.

Example

Example

```
GET /v1/authorization
Response:
{
        "data":
        {
                "displayName": "ote_test1",
                "hasRepoAccount": true,
                "role": 4,
                "privateProject": true,
                "enableIngress": false,
                "enableAlert": true,
                "enableAdminAutoDeploy": true
        },
        "code": 200,
        "message": "Success"
}
```

11.Create Sub-User

POST /v1/admin/user

Description

Create a user with access token authorization. This user will has the same business namespace with current logged-in administrator. This can only be done by general administrator and system administrator.

Parameters

Name	Type	Required	Default	Description
user	string	yes		user name
password	string	yes		password
phone	string	yes		phone number
isAdmin	bool	yes		true: admin false: general user
realName	string	yes		real name

Response code

Code	Description
200	Success
500	Unexpected internal errors

422	Parameter validation error

Business Management

1.Create Business

POST /v1/business

Description

Create a new business namespace. This can only be done by admin who do not create business yet.

Parameters

Name	Type	Required	Default	Description
name	string	yes		Business name
introduce	string	yes		Introduction of business
objective	string	yes		purpose for creating business
scale	string	yes		the scale of resources excepted to use

Example

Example

```
POST /v1/business
{
"name": "name",
"introduce": "intrduce",
"objective": "objective",
"scale": "scale"
}
```

2.Get Business List

GET /v1/business

Description

Return the business list.

parameters

Name	Type	Required	Default	Description
page	int	no	1	page
pageSize	int	no	10	page size
orderBy	string	no	id	orderBy allows sorting by id.
order	string	no	asc	sort order: asc, desc
name	string	no		business name filter

Response parameters

Name	Type	Required	Description
name	string	yes	business name
userId	int	yes	user id
comment	string	yes	audit comment

status	int8	yes	status 0: pending 1: approved 2: disapproved 3: deleted

Example

Example

```
GET /v1/business
{
        "data": [
        {
                    "name": "name",
                    "userId": 123,
                    "comment": "comment",
                    "id": 1,
                    "status": 0,
                    "createTime": 1590062196,
                    "updateTime": 1590062196
        }
          ],
        "total": 1,
        "code":200,
        "message": "Success"
}
```

3.Get Business

GET /v1/business/id/[id]

Description

Get the specify business.

Parameters

Name	Type	Required	Description
name	string	yes	business name
userId	int	yes	use id
introduce	string	yes	Introduction of business
objective	string	yes	purpose for creating business
scale	string	yes	the scale of resources excepted to use

Example

```
GET /v1/business/id/1

{
"data": {
    "name": "name",
    "userId": 123,
    "introduce": "introduce",
    "objective": "",
    "scale": "1400000000",
    "comment": "comment",
    "id": 1,
    "status": 0,
    "createTime": 1590062196,
    "updateTime": 1590062196
  }
"code":200,
"message": "Success"
}
```

4.Audit Business

PUT /v1/business/id/[id]

Description

This can only be done by system administrator. A namespace, named "ns+business_id", will be created in all edge clusters. For example, the business id is 123, then the name of namespace is ns123.

Parameters

Name	Type	Required	Description
id	string	yes	business id
comment	string	yes	audit comment
status	int8	yes	comment result: 1: approved 2: disapproved

Example

```
PUT /v1/business/id/1

{
    "comment": "pass",
    "status": 1
}
```

Cluster Management

1.Get Cluster List

GET /v1/cluster

Description

Get all the cluster list.

Parameters

Name	Type	Required	Default	Description
`clusterLabel`	string	no	all	cluster label filter
page	int	no	1	page

pageSize	int	no	no limited	page size

Response Parameters

Name	Type	Description
clusterName	string	cluster name
nodeCount	int	number of nodes in the cluster
isReady	bool	flag to indicate if the cluster is alive
total	int	total number of clusters under the filter

Example

Example

```
GET /v1/cluster?clusterLabel=all&page=1&pageSize=10
Response:
{
    "code" : 200,
    "message" : "success",
    "data": [
         { "clusterName": "GuangZhou", "nodeCount": 1, "isReady": false},
         { "clusterName": "BeiJing",   "nodeCount": 2, "isReady": true}
    },
    "total": 2
}
```

2.Add Label to Cluster

POST /v1/cluster/label

Description

Label the clusters.

Parameters

Name	Type	Required	Description
clusterName	[]string	yes	the list of clusters that need to be labeled.
clusterLabel	string	yes	label name that only use uppercase and lowercase letters, numbers and special character - .

Example

```
POST /v1/cluster/label
{
    "clusterLabel": "label1",
    "clusterName": ["GuangZhou", "BeiJing"]
}
Response
{
    "code" : 200,
    "message" : "success"
}
```

3.Delete Label

DELETE /v1/cluster/label

Description

Delete label of specified clusters.

Parameters

Name	Type	Required	Description
clusterName	[ ]string	yes	the list of clusters that need to be labeled.
clusterLabel	string	yes	label name that needs to be removed.

Example

```
DELETE /v1/cluster/label
{
    "clusterLabel": "label1",
    "clusterName": ["GuangZhou", "BeiJing"]
}
Response
{
    "code" : 200,
    "message" : "success"

}
```

4.Get Label List

GET /v1/cluster/label

Description

Get the list of labels that have been created.

Parameters

Response Parameters

Name	Type	Description
data	[]string	the list of label

Example

```
GET /v1/cluster/label
Response:
{
    "code" : 200,
    "message" : "success",
    "data": [
         "label1", "label2"
    ]
}
```

5.Get Label

GET /v1/cluster/label/[clusterNamel]

Description

Get the labels under the specified cluster.

Parameters

Name	Type	Required	Default	Description

clusterName	string	yes		cluster name
page	int	no	1	page
pageSize	int	no	10	page size. 0: no limited

Response Parameters

Name	type	Description
clusterLabel	string	cluster label
createTime	int64	create time of label
total	int32	total number

Example

```
GET /v1/cluster/label/cluster1?page=1&pageSize=10
Response:
{
    "code" : 200,
    "message" : "success",
    "data": [
         { "clusterLabel":"GuangZhou", "createTime": 1234567890 },
         { "clusterLabel":"BeiJing", "createTime": 1234567891 }
    ],
    "total": 2
}
```

Node Management

1.Get Node List

GET /v1/node

Description

Parameters

Name	Type	Required	default	Description
field	string	no		fields that need be presented
nodeLabel	string	no		node label to filter nodes
clusterName	string	no		cluster name for filtering result
nodeName	string	no		node name for filtering result.
page	int	no	1	the page number
pageSize	int	no	10	the size of per page

Response Parameters

Name	Tyep	Description
nodeName	string	node name
clusterName	string	cluster that node belongs to
operatingSystem	string	os of node
kernelVersion	string	kernel version of node
ip	string	ip address
memory	int64	memory(byte)
cpu	int	cpu core

isReady	bool	flag to indicate if the node is alive.
total	int	total number of nodes
gpu	int	GPU memory(byte)
disk	int64	disk size(byte)
readyNodeCount	int	the number of alive node

Example

```
GET /v1/node
Response:
{
    "code" : 200,
    "message" : "success",
    "data": [
         {
           "nodeName": "GZ1",
           "operatingSystem": "Linux",
           "clusterName": "GuangZhou",
           "kernelVersion" : "2.6",
           "isReady": true,
           "ip": "1.2.3.4"
         },
         {
           "nodeName": "BJ1",
           "operatingSystem": "Linux",
           "clusterName": "BeiJing",
           "kernelVersion" : "2.6",
           "isReady": true,
           "ip": "1.2.3.4"
         }
    },
    "meta": {
        "readyNodeCount": 2
    },
    "total": 2
}
```

2.Add Label to Node

POST /v1/node/label

Description

Label the nodes.

Parameters

Name	Type	Required	Description
nodeName	[]string	yes	the list of nodes that need to be labeled.
nodeLabel	string	yes	label name that only use uppercase and lowercase letters, numbers and special character - .
clusterName	string	yes	cluster that nodes belong to.

Example

```
POST /v1/node/label
{
    "nodeLabel": "label1",
    "clusterName": "c1",
    "nodeName": ["GuangZhou", "BeiJing"]
}
Response:
{
    "code" : 200,
    "message" : "success"
}
```

3.Delete Label

DELETE /v1/node/label

Description

Delete label of specified nodes.

Parameters

Name	Type	Required	Default	Description
nodeName	[ ]string	yes		the list of nodes that need to remove label.
nodeLabel	string	yes		the label that needs to be removed.
clusterName	string	yes		cluster that nodes belong to.
deleteDeploy	bool	no	no	allow to delete deployment on the node that have been labeled.

Example

```
DELETE /v1/node/label
{
    "nodeLabel": "label1",
    "clusterName": "c1",
    "nodeName": ["GuangZhou", "BeiJing"],
    "deleteDeploy": true
}
Response:
{
    "code" : 200,
    "message" : "success"
}
```

4.Get Label List

GET /v1/node/label

Description

Return the list of label of nodes under the specified cluster.

Parameters

Name	Type	Required	Description
clusterName	string	no	Return node labels in all clusters by default.

Response Parameters

Name	Type	Description

data	[ ]string	the list of label

Example

```
GET /v1/node/label?clusterName=c1
Response:
{
    "code" : 200,
    "message" : "success",
    "data": ["label1"]
}
```

Repository Management

1.Get Images

GET /v1/repository/image

Description

List images of the specified project.

Parameters

Name	Type	Required	Default	Description
orderBy	string	no	id	OrderBy allows sorting by id.
order	string	no	desc	Sort order: asc, desc.
page	int	no	1	The page number.
pageSize	int	no	10	The size of per page, 0 for no limited.
projectName	string	yes		The name of project.
imageName	string	no		Image name for filtering result.

Response code

Code	Description
200	Searched for images of Harbor successfully.
400	Invalid parameters.
401	User need to log in first.
403	User does not have permission of admin role.
500	Unexpected internal errors

Response Parameters

Name	Type	Description
projectName	string	The name of project
public	bool	The flag to indicate the publicity of the image.
imageName	string	The name of image
imageAddress	string	The registry url of image which do not including tag.
createTime	int	The create time of image
updateTime	int	The update time of image
total	int	total number of images

Example

```
GET /v1/repository/image
Response:
{
    "data": [
        {
            "projectName": "test",
            "public": false,
            "imageName": "test/test-demo1",
            "imageAddress": "registry.dcdn.baidu.com/test/test-demo1",
            "createTime": 1551758321,
            "updateTime": 1551758321
        },
        {
            "projectName": "test2",
            "public": false,
            "imageName": "test2/test-demo2",
            "imageAddress": "registry.dcdn.baidu.com/test2/test-demo2",
            "createTime": 1551758321,
            "updateTime": 1551758321

        }
    ],
    "total": 2,
    "code": 200,
    "message": "success"
}
```

2.Get Tag of Image

GET /v1/repository/image/tag?imageName=[imageName]

Description

List tags under the specific image name.

Parameters

Name	Type	Required	Default	Description
page	int	no	1	The page number.
pageSize	int	no	10	The size of per page, 0 for no limited.
imageName	string	yes		The image name for filtering result.

Response code

Code	Description
200	Get tags successfully.
400	Bad request.
401	User need to log in first.
403	User does not have permission of admin role.
500	Unexpected internal errors

Response Parameters

Name	Type	Description
tag	string	tag of docker image
imageAddress	string	repository of docker image
createTime	int	create time
updateTime	int	update time

total	int	total number of images

Example

```
GET /v1/repository/image/tag?imageName=test/demo1

{
    "data": [
        {
            "tag": "latest",
            "imageAddress": "registry.dcdn.baidu.com/test/test-demo1",
            "createTime": 1551758321,
            "updateTime": 1551758321
        },
        {
            "tag": "v1",
            "imageAddress": "registry.dcdn.baidu.com/test/test-demo1",
            "createTime": 1551758321,
            "updateTime": 1551758321

        }
    ],
    "total": 2,
    "code": 200,
    "message": "success"
}
```

3.Delete Image

DELETE /v1/repository/image

Description

Delete multiple images specified by name.

Parameters

Name	Type	Required	Default	Descripton
imageName	[ ]string	yes		name of images

Example

```
DELETE /v1/repository/image
Body:
{
   "imageName": ["img1", "img2"]
}


Response:
{
    "code": 200,
    "message": "success"
}
```

4.Delete Tag of Image

DELETE /v1/repository/image/tag

Description

Delete the image specified by name and tags.

Parameters

Name	Type	Required	Default	Description
tag	[ ]string	yes		tags of image that needs to be removed
imageName	string	yes		name of image that needs to be removed

Example

```
DELETE /v1/repository/image/tag
Body:
{
   "imageName": "test/demo1"
   "tag": ["v1", "v2"]
}


Response:
{
    "code": 200,
    "message": "success"
}
```

5.Create User

POST /v1/repository/user

Description

Create a new user of harbor repository.

Parameters

Name	Type	Required	Default	Description
user	string	yes		username
password	string	yes		password

Response code

Code	Description
200	User created successfully.
400	Unsatisfied with constraints of the user creation.
409	username conflict
403	User registration can only be used by admin role user when self-registration is off.
415	The Media Type of the request is not supported, it has to be "application/json"
500	Unexpected internal errors.

Example

```
POST /v1/repository/user
{
"user": "user1",
"password": "xxxx"
}

Response:
{
    "code": 200,
    "message": "success"
}
```

6.Update Password

PUT /v1/repository/user

Description

Modify password of current logged-in user.

Parameters

Name	Type	Required	Default	Description
password	string	yes		new password

Response code

Code	Description
200	Updated password successfully.
400	Invalid user ID; Old password is blank; New password is blank.
401	Don't have authority to change password. Please check login status.
403	The caller does not have permission to update the password of the user with given ID, or the old password in request body is not correct.
500	Unexpected internal errors.

Example

```
PUT /v1/repository/user
{
"password": "xxxx"
}

Response:
{
    "code": 200,
    "message": "success"
}
```

7.Get Projects of Repository

GET /v1/repository/project

Description

List all projects of current logged-in user.

Parameters

Name	Type	Required	Default	Description
projectName	string	no		Project name for filtering results.
public	bool	no		Public sign for filtering projects.
page	int	no	1	The page number.
pageSize	int	no	10	The size of per page.

Response code

Code	Description
401	User need to log in first.
200	Return all matched projects.
500	Internal errors.

Response parameters

Name	Type	Description
projectId	int	Project ID.
projectName	string	The name of the project
createTime	string	The creation time of the project.
updateTime	string	The update time of the project.
public	bool	The public status of the project.
imageCount	int	The count of the images under this project.
total	int	Total number of the projects.

Example

```
GET /v1/repository/project

Response:
{
"data": [ {
  "projectId": 13,
  "projectName": "calico",
  "createTime": 1551758321,
  "updateTime": 1551758321,
  "imageCount": 3,
  "public": true
  } ],
"total": 2,
"code": 200,
"message": "success"
}
```

8.Create Project

POST /v1/repository/project

Description

Create a new project of the current user.

Parameters

Name	Type	Required	Default	Description
projectName	string	yes		The name of project.
public	bool	yes		The public status of the project.

Response code

Code	Description
201	Project created successfully.
400	Unsatisfied with constraints of the project creation.
401	User need to log in first.
409	Project name already exists.
415	The Media Type of the request is not supported, it has to be "application/json"
500	Unexpected internal errors.

Example

```
POST /v1/repository/project
{
  "projectName": "calico",
  "public": true
}

Response:
{
"code": 200,
"message": "success"
}
```

9.Delete Project

DELETE /v1/repository/project

Description

Delete the projects specified by ID.

Parameters

Name	Type	Required	Default	Description
projectId	[ ]int64	yes		The IDs of projects

Response code

Code	Description
201	Project created successfully.
400	Unsatisfied with constraints of the project creation.
401	User need to log in first.
409	Project name already exists.
415	The Media Type of the request is not supported, it has to be "application/json"
500	Unexpected internal errors.

Example

```
DELETE /v1/repository/project
{
  "projectId": [ 1, 2 ]
}

Response:
{
"code": 200,
"message": "success"
}
```

10.Create External Repository

POST /v1/repository/third

Description

Create a secret registry record for pulling docker images from external repository.

Parameters

Name	Type	Required	Description

repositoryId	string	yes	A unique name for specifying repository.
address	string	yes	The address of repository.
user	string	yes	username for pulling images.
password	string	yes	password for pulling images.

Example

```
POST /v1/repository
Body:
{
    "repositoryId": "repo1",
    "address": "127.0.0.1",
    "user": "user1",
    "password": "password1"
}


Response:
{
"code": 200,
"message": "success"
}
```

11.Delete External Repository

DELETE /v1/repository/third

Description

Delete the external repository record by ID. It won't affect existing deployment.

Parameters

Name	Type	Required	Description
repositoryId	[ ]string	yes	The ID of repository

Example

```
DELETE /v1/repository
Body:
{
    "repositoryId": ["repo1", "repo2"]
}
Response:
{
        "code": 200,
        "message": "success"
}
```

12.Get External Repository List

GET /v1/repository/third

Description

List the external repository.

Parameters

Name	Type	Required	Default	Description
pageSize	int	no	10	The size of per page, 0 for no limited.
page	int	no	1	The page number.

orderBy	string	no	id	orderBy allows sorting by id.
order	string	no	asc	sort order: asc, desc.

Response Parameters

Name	Type	Description
repositoryId	string	The unique name for specifying repository.
address	string	The address of repository.
user	string	username for pulling images.

Example

```
GET /v1/repository/third?pageSize=0
response:
{
    "data": [ {
                "id": 1,
                "repositoryId":"yq01",
                "address":"xxzxsdsdsds",
                "user":"ddddddd"
    }, {
                "id": 2,
                "repositoryId":"yq02",
                "address":"xxzxsdsdsds",
                "user":"ddddddd"
    } ],
    "code": 200,
    "total": 2,
    "message": "Success"
}
```

13.Get External Repository

GET /v1/repository/third/[repoId]

Description

Return the external repository by ID.

Parameters

Name	Type	Required	Default	Description
repoId	string	yes		The unique name of repository

Response code

200	Get repository successfully.
404	Not found
500	Unexpected internal errors

Response parameters

Name	Type	Description
repositoryId	string	The unique name for specifying repository.
address	string	The address of repository.
user	string	username for pulling images.

Example

```
GET /v1/repository/third/repo1
Response:
{
    "data": {
                "repositoryId":"yq01",
                "address":"xxzxsdsdsds",
                "user":"ddddddd"
    },
    "code": 200,
    "message": "Success"
}
```

Application Management

1.Create Application

POST /v1/app

Parameters

Name	Type	Required	Description
appName	string	yes	The name of applicaton
deployType	string	yes	Deployment mode: Only deployment or daemonset can be set
image	string	yes	Image repository with specified tag.
version	string	yes	Main version with format x.x, for example 1.1.
repositoryId	string	no	The ID of repository whose secret can be used to pull docker image.
port	array	yes	The exposed port.
command	string	no	Start command line.
volume	array	no	Volume information.
env	array	no	Executing environment.
replicas	uint32	no	Default replicas. Effective only at deployment.
minReplicas	uint32	no	Minimal replicas. Effective only at deployment.
maxReplicas	uint32	no	Maximun replicas. Effective only at deployment.
minCPU	uint32	yes	Minimal cpu (%
maxCPU	uint32	yes	Maximun cpu(%
GPU	uint32	no	GPU count
isHPA	bool	yes	Allow to use horizontal pod autoscaler
targetCPUUtilization	uint32	yes	CPU utilization which will trigger autoscale
targetMemUtilization	uint32	yes	CPU utilization which will trigger autoscale
minMemory	uint32	yes	Minimal memory (MB
maxMemory	uint32	yes	Maximun memory (MB
maxUnavailable	uint32	yes	Max unavailable pod
maxSurge	uint32	yes	max surge pod
minReadySeconds	uint32	yes	min ready seconds for updating

Example

Example

```
POST  /v1/app
Body:
{
    "appName": "nginx1",
    "deployType": "deployment",
    "image": "nginx:latest",
    "version": "1.2",
    "repositoryId": "repo1",
    "replicas": 10,
    "port": [
       {"port": 8080, "hostPort": 9090}],
    "command": "",
    "volume": [
       {"path": "/host/usr/bin", "hostPath": "/usr/bin", "readOnly": true},
       {"path": "/host/usr/sbin", "hostPath": "/usr/sbin", "readOnly": true} ],
    "env": [
       {"name": "NGINX_PATH", "value": "/usr/share/nginx/" } ],
    "minReplicas": 2,
    "maxReplicas": 20,
    "minCPU": 10,
    "maxCPU": 200,
    "isHPA": true,
    "targetCPUUtilization": 100,
    "targetMemUtilization": 150,
    "minMemory": 100,
    "maxMemory": 200,
    "maxUnavailable": 1,
    "maxSurge": 1,
    "minReadySeconds": 10,
}
Response:
{
        "code": 200,
        "message": "success"
}
```

2.Delete Application

DELETE /v1/app

Parameters

Name	Type	Required	Description
appId	[ ]int	yes	The IDs of application

Example

```
DELETE  /v1/app
{
   "appId": [1,2,3,4]
}
```

3.List Applications

GET /v1/app

Description

List applications.

Parameters

Name	Type	Required	Default	Description
pageSize	int	no	10	The size of per page, 0 for limited.
page	int	no	1	The page number
appName	string	no		application name for filtering result
appNameMode	string	no	partial	the filter mode: strict, partial, prefix
version	string	no		version

Example

```
Get /v1/app
{
  "apps": [ {
    "id": 1,
    "appName": "nginx1",
    "status": 1,
    "deployType": "deployment",
    "image": "nginx:latest",
    "projectName": "proj1",
    "repositoryId": "repo1",
    "version": "1.2.3.4",
    "replicas": 10,
    "port": [
       {"port": 8080, "hostPort": 9090}],
    "command": "",
    "volume": [
       {"path": "/host/usr/bin", "hostPath": "/usr/bin", "readOnly": true},
       {"path": "/host/usr/sbin", "hostPath": "/usr/sbin", "readOnly": true} ],
    "dependence": [
       {"name": "nginx", "service": "xxxx", "port": 9090 } ],
    "env": [
       {"name": "NGINX_PATH", "value": "/usr/share/nginx/" } ],
    "minReplicas": 2,
    "maxReplicas": 20,
    "minCPU": 0.1,
    "maxCPU": 2,
    "GPU": 2,
    "isHPA": true,
    "targetCPUUtilization": 1,
    "targetMemUtilization": 150,
    "minMemory": 100,
    "maxMemory": 200,
    "maxUnavailable": 1,
    "maxSurge": 1,
    "minReadySeconds": 10,
    "createTime": 1551758321,
    "updateTime": 1551758321},
  ]
  "total": 1,
  "code": 200,
  "message": "Success"
}
```

4.Get Application

GET /v1/app/id/{appId}

Description

Return specified application information by ID.

Parameters

Name	Type	Required	Description
appId	int	no	The ID of application

Example

```
GET  /v1/app/id/1
{
  "data": {
    "id": 1,
    "appName": "nginx1",
    "status": 1,
    "deployType": "deployment",
    "image": "nginx",
    "projectName": "proj1",
    "repositoryId": "repo1",
    "version": "1.2.3",
    "replicas": 10,
    "port": [
       {"port": 8080, "hostPort": 9090}],
    "command": "",
    "volume": [
       {"path": "/host/usr/bin", "hostPath": "/usr/bin", "readOnly": true},
       {"path": "/host/usr/sbin", "hostPath": "/usr/sbin", "readOnly": true} ],
    "dependence": [
       {"name": "nginx", "service": "xxxx", "port": 9090 } ],
    "env": [
       {"name": "NGINX_PATH", "value": "/usr/share/nginx/" } ],
    "minReplicas": 2,
    "maxReplicas": 20,
    "minCPU": 0.1,
    "maxCPU": 2,
    "GPU": 1,
    "isHPA": true,
    "targetCPUUtilization": 1,
    "targetMemUtilization": 150,
    "minMemory": 100,
    "maxMemory": 200,
    "maxUnavailable": 1,
    "maxSurge": 1,
    "minReadySeconds": 10,
    "createTime": 1551758321,
    "updateTime": 1551758321
  },
  "code": 200,
  "message": "Success"
}
```

5.Upload Helm Chart to Create Application

POST /v1/app/chart

Description

Upload a helm chart tar file to create the application. The request type must be multipart/form-data.

Parameters

Name	Type	Required	Description
appName	string	yes	The name of application
version	string	yes	version
chart	file	yes	the chart file needs to be upload

Example

```
POST /v1/app/chart
appName=test
version=1.1
chart=@file

curl 0.0.0.0/v1/app/chart -F "version=1.2" -F "appName=test1" -F "chart=@file"
```

Deployment Management

1.Create Deployment

POST /v1/deploy

Description

Deploy the application to specified cluster.

Parameters

Name	Type	Required	Description
deployName	string	yes	The unique name of deployment.
appName	string	yes	The application that will be deployed.
version	string	yes	The version of application.
cluster	string	yes	The label of cluster that will install the application, default all cluster.
nodeLabel	string	yes	The label of node that will install the application, default all node.
comment	string	yes	The comment about deployment.

Example

```
POST  /v1/deploy/install
Body:
{
    "deployName": "nginx1",
    "appName": "nginx1",
    "version": "1.1.1.1",
    "cluster": "cluster1",
    "nodeLabel": "node1",
    "comment": ""
}
Response:
{
    "code": 200,
    "message": "success"
}
```

2.Delete Deployment

DELETE /v1/deploy/{deployName}

Description

Delete deployment by name.

Parameters

Name	Type	Required	Description
deployName	string	yes	The name of deployment.

Example

```
DELETE  /v1/deploy/deploy1
Response:
{
    "code": 200,
    "message": "success"
}
```

3.Upgrade Deployment

POST /v1/deploy/id/{deployId}

Description

Upgrade the deployment to specified version.

Parameters

Name	Type	Required	Description
deployId	int	yes	The ID of deployment.
version	string	yes	The version of application that needs to be upgraded to.
comment	string	no	The comment about upgrade.

Example

```
POST  /v1/deploy/id/1
Body:
{
    "version": "1.1.1.2"
}
Response:
{
    "code": 200,
    "message": "success"
}
```

4.List Deployment

GET /v1/deploy

Description

List all deployments.

Parameters

Name	Type	Required	Default	Description
pageSize	int	no	10	The size of per page, 0 for no limited.
page	int	no	1	The page number
field	string	no		Fields that need be presented, default all informations.
appName	string	no		The name of application for filtering result.
appNameMode	string	no	partial	The mode of filter: strict, partial, prefix.
editable	string	no	all	The flag, which indicates if the deployment can be update now, is used for filtering result. The value can be: true, false, or all.
running	string	no	all	The running status of deployment for filtering result. The value can be: true, or all.

Response Parameters

Name	Type	Description
id	int32	ID

deployName	string	The name of deployment.
appName	string	The name of application.
version	string	The version of application.
cluster	string	The cluster that have deployed.
nodeLabel	string	The node that have deployed.
comment	string	The comment about deployment.
deployType	int32	The deployment type: 1: new deployment, 2: upgrading, 3: rollbacking, 4: deleted
status	int32	The status of deployment0: pending 1: Approved 2: Disapproved 3:Processing 4:Deploying 5: Success 6: Partial success 7: Failed 8: Internal error 9: Deleted
createTime	int32	The creation time of deployment.
updateTime	int32	The update time of deployment.
editable	bool	The flag that indicates if the deployment can be update now.
errorMessage	string	The detailed deployment status which contains the count of running pod and error pod.
auditComment	string	The comment about audit.

Example

```
GET  /v1/deploy?pageSize=10&page=1
Response:
{
 "data": [ {
    "id": 1,
    "deployName": "nginx1",
    "appName": "nginx1",
    "version": "1.1.1.1",
    "cluster": "cluster1",
    "nodeLabel": "node1",
    "deployType": 1
    "status": 6,
        "errorMessage": "{\"running\": 10, \"error\": 3, \"total\": 13}",
    "comment": ""
    "editable": true,
    "createTime": 1551758321,
    "updateTime": 1551758321}
  ],
  "total": 1
}
```

5.Rollback Deployment

PUT /v1/deploy/id/{deployId}

Description

Rollback the deployment to previous version.

Parameters

Name	Type	Required	Description
deployId	int	yes	The ID of deployment.
version	string	yes	The version that needs to rollback to.
comment	string	yes	The comment about rollback.

Example

```
PUT  /v1/deploy/id/1
Body:
{
    "version": "1.1.1.2",
    "comment": ""
}
Response:
{
    "code": 200,
    "message": "success"
}
```

6.Get Deployment

`GET /v1/deploy/id/{deployId}`

Description

Get specified deployment information by ID.

Parameters

Name	Type	Required	Description
field	string	no	The fields that need to be presented, default all information without history version information. But when field=historyVersion, the API will return all versions of deployed application.

`Response parameters`

Same as the API of List Deployment

```
GET  /v1/deploy/id/1
Response:
{
  "data" : {
    "id": 1,
    "deployName": "nginx1",
    "appName": "nginx1",
    "version": "1.1.1.2",
    "cluster": "cluster1",
    "nodeLabel": "node1",
    "status": 6,
    "comment": "",
    "errorMessage": "{\"running\": 10, \"error\": 3, \"total\": 13}",
    "deployType": 1,
    "createTime": 1551758321,
    "updateTime": 1551758321
    }
  }
}
```

7.Delete Deployment

DELETE /v1/deploy/id/{deployId}

Parameters

Name	Type	Required	Description
deployId	int	yes	The ID of deployment that needs to be removed

Example

```
DELETE  /v1/deploy/id/1
```

8.Audit Deployment

PUT /v1/admin/deploy

Description

Audit the deployment created by general user. This can only be done by general administrator and system administrator.

Parameters

Name	Type	Required	Description
id	int	yes	ID
status	int	yes	The status of deployment
comment	string	yes	The comment of

Example

```
POST /v1/admin/deploy
Body:
{
  "id": 123456,
  "status": 0,
  "comment": "xxxx"
}
```

9.List Audit List of Deployment

GET /v1/admin/deploy

Description

List deployment that needs to be audited. This can only be done by general administrator and system administrator.

Parameters

Name	Type	Required	Default	Description
orderBy	string	no	`id`	orderBy allows sorting by id.
order	string	no	desc	order sort: asc,desc
businessName	string	no		The name of business for filtering result
page	int	no	1	The page number
pageSize	int	no	10	The page size of per page

Response code

Code	Description
200	Successfully.
422	Unprocessable Entity
500	Unexpected internal errors

Response parameters

Name	Type	Description
id	int	ID
businessName	string	The name of business.
appName	string	The name of application.
version	string	The version of application
cluster	string	The cluster that will be deployed to.

deployId	int	The ID of deployment.
deployType	int	The type of deployment, 1: new deployment, 2: upgrading, 3: rollbacking, 4: deleted
status	int	The status of audit.
comment	string	The comment of audit.
createTime	int	The create time.
updateTime	int	The time of the audit.

Example

```
GET /v1/admin/deploy

Response:
{
    "data": [
        {
            "id": 6878,
            "businessName": "a",
            "appName": "dsdar2",
            "version": "1.2.1",
            "cluster":"bjct",
            "deployType":1,
            "status":0,
            "comment":"no pass",
            "createTime": 1551758321,
            "updateTime": 1551758321
        }
    ],
    "total": 1,
    "code": 200,
    "message": "success"
}
```