Video Security Monitoring R5 Installation Document

- Introduction
- License
- How to use this document
- Deployment Architecture
- Pre-Installation Requirements
    - Hardware Requirements
        - Minimum Hardware Requirements
        - Recommended Hardware Requirements
    - Software Prerequisites
    - Database Prerequisites (option)
        - Schema scripts
    - Other Installation Requirements
        - Jump Host Requirements
        - Network Requirements
        - Bare Metal Node Requirements
        - Execution Requirements (Bare Metal Only)
- Installation High-Level Overview
    - Bare Metal Deployment Guide
        - Install Bare Metal Jump Host
        - Creating a Node Inventory File
        - Creating the Settings Files
        - Running
            - Prerequisite
            - Deploy ote-stack on one IEC cluster
            - Enable edge autonomy feature in your cluster
            - Add a new cluster to ote-stack for management (option)
    - Virtual Deployment Guide
        - Standard Deployment Overview
        - Snapshot Deployment Overview
        - Special Requirements for Virtual Deployments
            - Install Jump Host
            - Verifying the Setup - VMs
    - Upstream Deployment Guide
        - Upstream Deployment Key Features
        - Special Requirements for Upstream Deployments
        - Scenarios and Deploy Settings for Upstream Deployments
        - Including Upstream Patches with Deployment
        - Running
        - Interacting with Containerized Overcloud
- Verifying the Setup
- Developer Guide and Troubleshooting
    - Utilization of Images
    - Post-deployment Configuration
    - Debugging Failures
    - Reporting a Bug
- Uninstall Guide
- Troubleshooting
    - Error Message Guide
- Maintenance
    - Blue Print Package Maintenance
        - Software maintenance
        - Hardware maintenance
    - Blue Print Deployment Maintenance
- Frequently Asked Questions
- License
- References
- Definitions, acronyms and abbreviations

Introduction

The document provides guidance to install the ote-stack platform on AI Edge Blueprint, which covers the information of hardware requirement, prerequisite software and the detailed step. The ote-stack platform can be installed on Kubernetes or on a system with Docker support. The IEC cluster, which is based on Kubernetes, is chosen to be the edge infrastructure for AI Edge. The installation guide of IEC R2 can be got in here.

The guide shows step by step to deploy ote-stack platform with one edge cluster as well as add a new edge cluster to management.

License

Apache 2.0 license

How to use this document

The guide covers the details of pre-installation requirements, installation steps and uninstallation guide. The installation steps should be executed before the prerequisites and pre-installation requirements are ready. In addition, one IEC edge cluster with 3 nodes needs to be prepared in advance.

Deployment Architecture

The Deployment Architecture uses one IEC edge cluster with 3 nodes as the basic infrastructure and the ote-stack platform will installed on the cluster.

Pre-Installation Requirements

Hardware Requirements

This section describes the requirements for building IEC edge infrastructure and ote-stack platform. At least three edge nodes are needed to build the edge infrastructure.

Minimum Hardware Requirements

HW Aspect	Requirement
1 Jumpserver	A physical or virtualized machine that has direct network connectivity to the edge nodes. For **virtual** deployments, CPU/RAM/disk requirements of cluster nodes should be satisfiable as virtual machine resources when using the **jumpserver** as a hypervisor.
CPU	Minimum 1 socket for IEC and 2 socket for ote-stack components and usecase test (each cluster node)
RAM	Minimum 4GB/server (Depending on usecase work load)
Disk	Minimum 20GB (each cluster node)
Networks	Mininum 1

Recommended Hardware Requirements

3 Huawei TaiShan 2280 Arm Server are required:

Processor	2*Kunpeng 920 processor
RAM	12*32G-DDR4-2666MT/s
Storage	3*3.5 inch 4000G SATA and 1*2.5 inch 3200G NVMe SSD
Network	1 onboard network card, each card supports 4*GE port or 4*10GE port or 4*25GE port

Refer TaiShan 2280 Server User Guide here

Software Prerequisites

- Available harbor (>=1.7, <2.0) with admin user and password
- At least one IEC edge cluster with healthy container network.

Database Prerequisites (option)

A mysql database is required to store the related metadata of OTE models, like users, businesses, projects, apps, deployments, clusters, nodes, etc. When the OTE runs, the OpenAPI Server will connect to the database "sys_ote_manage_platform" for providing services. Some brief description of tables which related to this database are shown below:

- **tb_system_config** stores global configuration of OTE web portal.
- **tb_business_info** maintains the relation between the users of OTE and the namespace which divide cluster resources between different users.
- **tb_ote_web_users** stores the informations of general users and administrator in OTE stack.
- **tb_ote_web_repository_users** stores the information of users of harbor server for pulling docker images when applications are being deployed.
- **tb_ote_cluster_label** stores the labels which are attached to edge clusters, while the information of clusters are stored as CRD resource in the etcd.
- **tb_ote_node_label** stores the labels which are attached to edge nodes, while the information of edge nodes are stored as CRD resource in the etcd.
- **tb_app_info** stores the information about application that will be deployed, such as name, image, required resource, execution command.
- **tb_deploy_info** maintains the information and status of deployments related to the application that stored in tb_app_info.
- **tb_alert_info** displays alert messages sent by alertmanager server.

Following the installation steps below, a new mysql database container, which contains the configuration and db tables of OTE, will be deployed to the edge cluster. If you need to store these OTE metadata in your own database, you can execute the following schema script to create the database and the related tables.

Schema scripts

The following schema scripts will help you to generate the database for OTE-stack.

```
CREATE DATABASE sys_ote_manage_platform;
USE sys_ote_manage_platform;
SET names utf8;

CREATE TABLE IF NOT EXISTS `tb_system_config` (
  `key` varchar(64) NOT NULL,
  `value` varchar(64) NOT NULL,
  PRIMARY KEY (`key`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='system configuration';

CREATE TABLE IF NOT EXISTS `tb_business_info` (
  `id` bigint(20) unsigned NOT NULL AUTO_INCREMENT,
  `name` varchar(64) NOT NULL COMMENT 'business name',
  `namespace` varchar(64) NOT NULL DEFAULT '' COMMENT 'namespace',
  `user_id` bigint(20) unsigned NOT NULL COMMENT 'user id',
  `introduce` varchar(512) NOT NULL COMMENT 'business introduction',
  `objective` varchar(512) NOT NULL COMMENT 'resource requirements',
  `scale` varchar(512) NOT NULL COMMENT 'business scale',
  `comment` varchar(512) NOT NULL DEFAULT '' COMMENT 'review comments',
  `status` tinyint NOT NULL DEFAULT 0 COMMENT 'status',
  `update_time` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  `create_time` TIMESTAMP NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY (`name`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE IF NOT EXISTS `tb_ote_web_users` (
  `id` bigint(20) unsigned NOT NULL AUTO_INCREMENT,
  `uid` varchar(50) COLLATE utf8_unicode_ci NOT NULL COMMENT 'user id',
  `namespace` varchar(64) COLLATE utf8_unicode_ci DEFAULT '' COMMENT 'namespace',
  `user_name` varchar(50) COLLATE utf8_unicode_ci DEFAULT '' COMMENT 'username',
  `display_name` varchar(50) COLLATE utf8_unicode_ci DEFAULT '' COMMENT 'nickname',
  `real_name` varchar(50) COLLATE utf8_unicode_ci NOT NULL DEFAULT '' COMMENT 'real name',
  `password` varchar(60) COLLATE utf8_unicode_ci NOT NULL DEFAULT '' COMMENT 'password',
  `email` varchar(255) COLLATE utf8_unicode_ci NOT NULL DEFAULT '' COMMENT 'email',
  `phone` varchar(11) COLLATE utf8_unicode_ci NOT NULL DEFAULT '' COMMENT 'phone',
  `status` tinyint(4) unsigned NOT NULL DEFAULT '3' COMMENT 'status: 0:pending,1:reviewed,2:failed,3:forbidden',
  `is_admin` tinyint(3) unsigned NOT NULL DEFAULT '0' COMMENT 'is admin',
  `role` tinyint unsigned NOT NULL DEFAULT '0' COMMENT 'role:0:unauthorized,1:general manager,2:admin,3:super
```

```
admin',
  `updated_at` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  `created_at` timestamp NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `tb_ote_web_users_uid_unique` (`uid`) USING BTREE,
  UNIQUE KEY `tb_ote_web_users_phone_unique` (`phone`) USING BTREE,
  UNIQUE KEY `tb_ote_web_users_user_name_unique` (`user_name`) USING BTREE
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

CREATE TABLE IF NOT EXISTS `tb_ote_web_repository_users` (
  `id` bigint(20) unsigned NOT NULL AUTO_INCREMENT,
  `namespace` varchar(64) NOT NULL DEFAULT '' COMMENT 'namespace',
  `repository_uid` bigint(20) unsigned DEFAULT '0' COMMENT 'uid',
  `repository_username` varchar(50) COLLATE utf8_unicode_ci NOT NULL COMMENT 'username',
  `repository_email` varchar(255) COLLATE utf8_unicode_ci NOT NULL DEFAULT '' COMMENT 'email',
  `repository_password` varchar(255) COLLATE utf8_unicode_ci NOT NULL DEFAULT '' COMMENT 'password',
  `updated_at` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  `created_at` timestamp NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY (`namespace`) USING BTREE
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci COMMENT='harbor user';

CREATE TABLE IF NOT EXISTS `tb_ote_web_third_repository` (
  `id` bigint(20) unsigned NOT NULL AUTO_INCREMENT,
  `namespace` varchar(64) COLLATE utf8_unicode_ci NOT NULL DEFAULT '' COMMENT 'user namespace',
  `repository_id` varchar(50) COLLATE utf8_unicode_ci NOT NULL DEFAULT '' COMMENT 'repository id',
  `repository_url` varchar(500) COLLATE utf8_unicode_ci NOT NULL DEFAULT '' COMMENT 'repository addr',
  `repository_username` varchar(50) COLLATE utf8_unicode_ci NOT NULL DEFAULT '' COMMENT 'username,
  `repository_password` varchar(255) COLLATE utf8_unicode_ci NOT NULL COMMENT 'passwoed',
  `updated_at` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  `created_at` timestamp NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY (`namespace`, `repository_id`) USING BTREE
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci COMMENT='third repo';

CREATE TABLE IF NOT EXISTS `tb_ote_cluster_label` (
  `id` bigint(20) unsigned NOT NULL AUTO_INCREMENT,
  `cluster_name` varchar(64) COLLATE utf8_unicode_ci NOT NULL DEFAULT '' COMMENT 'cluster name',
  `label` varchar(64) COLLATE utf8_unicode_ci NOT NULL DEFAULT '' COMMENT 'label',
  `updated_at` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  `created_at` timestamp NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY (`cluster_name`, `label`) USING BTREE
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

CREATE TABLE IF NOT EXISTS `tb_ote_node_label` (
  `id` bigint(20) unsigned NOT NULL AUTO_INCREMENT,
  `cluster_name` varchar(64) COLLATE utf8_unicode_ci NOT NULL DEFAULT '' COMMENT 'cluste',
  `node_name` varchar(64) COLLATE utf8_unicode_ci NOT NULL DEFAULT '' COMMENT 'nodename',
  `label` varchar(64) COLLATE utf8_unicode_ci NOT NULL DEFAULT '' COMMENT 'node label',
  `updated_at` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  `created_at` timestamp NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY (`cluster_name`, `node_name`, `label`) USING BTREE
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

CREATE TABLE IF NOT EXISTS `tb_app_info` (
  `id` bigint(20) unsigned NOT NULL AUTO_INCREMENT,
  `namespace` varchar(64) NOT NULL COMMENT 'namespace',
  `app_name` varchar(64) NOT NULL COMMENT 'app_name',
  `main_version` varchar(32) NOT NULL COMMENT 'major version',
  `version_count` int(11) NOT NULL COMMENT 'minor version',
  `image` varchar(256) NOT NULL COMMENT 'image',
  `repository_id` varchar(64) NOT NULL DEFAULT '' COMMENT 'repository_id',
  `port` varchar(1024) NOT NULL DEFAULT '0' COMMENT 'port',
  `env` varchar(1024) NOT NULL DEFAULT '' COMMENT 'env',
  `volume` varchar(1024) NOT NULL DEFAULT '' COMMENT 'volume',
  `dependence` varchar(1024) NOT NULL DEFAULT '' COMMENT '',
  `replicas` int(10) NOT NULL DEFAULT '1' COMMENT 'replicas',
  `command` varchar(512) NOT NULL DEFAULT '' COMMENT 'command',
  `deploy_type` tinyint(4) NOT NULL DEFAULT '0' COMMENT 'deploy_type: 0:deployment, 1:daemonset',
```

```
  `gpu` int(10) NOT NULL DEFAULT '0',
  `request_cpu` int(10) NOT NULL DEFAULT '80',
  `request_mem` int(10) NOT NULL DEFAULT '4096',
  `limit_cpu` int(10) NOT NULL DEFAULT '80',
  `limit_mem` int(10) NOT NULL DEFAULT '4096',
  `min_replicas` int(10) NOT NULL DEFAULT '1',
  `max_replicas` int(10) NOT NULL DEFAULT '5',
  `is_hpa` tinyint(4) NOT NULL DEFAULT '0' COMMENT 'is_hpa0no1yes',
  `hpa_target_cpu` int(10) NOT NULL DEFAULT '80',
  `hpa_target_mem` int(10) NOT NULL DEFAULT '4096',
  `min_ready_seconds` int(10) NOT NULL DEFAULT '10',
  `max_surge` int(10) NOT NULL DEFAULT '1',
  `max_unavailable` int(10) NOT NULL DEFAULT '0',
  `status` tinyint(4) NOT NULL DEFAULT '0' COMMENT 'statur:0:init,1:success,2:fail,3:logical del,4:del',
  `update_time` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP COMMENT ,
  `create_time` timestamp NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `namespace` (`namespace`,`app_name`,`main_version`,`version_count`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE IF NOT EXISTS `tb_deploy_info` (
  `id` bigint(20) unsigned NOT NULL AUTO_INCREMENT,
  `name` varchar(128) NOT NULL COMMENT 'name',
  `namespace` varchar(64) NOT NULL COMMENT 'namespace',
  `app_name` varchar(64) NOT NULL COMMENT 'app_name',
  `version` varchar(48) NOT NULL COMMENT 'version',
  `cluster` varchar(64) NOT NULL COMMENT 'cluster',
  `node_label` varchar(64) NOT NULL DEFAULT 'all',
  `status` tinyint(4) NOT NULL DEFAULT '0' COMMENT 'status',
  `editable` tinyint(4) NOT NULL DEFAULT '1' COMMENT 'editable: 0:no1:yes',
  `running` tinyint(4) NOT NULL DEFAULT '0' COMMENT 'running: 0:no1:yes',
  `deploy_type` tinyint(4) NOT NULL DEFAULT '0' COMMENT 'deploy_type: 0:new,1:upgrade,2:rollback,3:del',
  `comment` varchar(256) NOT NULL DEFAULT '' COMMENT 'comment',
  `audit_comment` varchar(256) NOT NULL DEFAULT '' COMMENT 'audit_comment',
  `error_message` varchar(256) NOT NULL DEFAULT '' COMMENT 'error_message',
  `update_time` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  `create_time` timestamp NOT NULL,
  `execute_time` timestamp NOT NULL,
  PRIMARY KEY (`id`),
  KEY `name` (`namespace`,`name`),
  KEY `namespace` (`namespace`,`app_name`,`cluster`,`version`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE IF NOT EXISTS tb_domain_info (
  `id`       BIGINT UNSIGNED NOT NULL AUTO_INCREMENT,
  `namespace`  varchar(64) NOT NULL COMMENT 'namespace',
  `domain`   varchar(128) NOT NULL COMMENT 'domain',
  `used_count`   int NOT NULL DEFAULT 0 COMMENT 'used_count',
  `update_time`  TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  `create_time`  TIMESTAMP NOT NULL,
  PRIMARY KEY(`id`),
  KEY(`namespace`),
  UNIQUE KEY(`domain`)
) ENGINE = InnoDB DEFAULT CHARACTER SET = utf8;

CREATE TABLE IF NOT EXISTS tb_ingress_info (
  `id`         BIGINT UNSIGNED NOT NULL AUTO_INCREMENT,
  `namespace`  varchar(64) NOT NULL COMMENT 'namespace',
  `domain`     varchar(128) NOT NULL COMMENT 'domain',
  `uri`        varchar(128) NOT NULL COMMENT 'uri',
  `is_rewrite` int8 NOT NULL DEFAULT 0 COMMENT '',
  `deploy_name`  varchar(64) NOT NULL COMMENT 'deploy name',
  `status`       tinyint(4) NOT NULL DEFAULT '0' COMMENT 'status',
  `unique_key`   varchar(64) NOT NULL DEFAULT 'unique key',
  `update_time`  TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  `create_time`  TIMESTAMP NOT NULL,
  PRIMARY KEY(`id`),
  UNIQUE KEY(`namespace`,`unique_key`)
) ENGINE = InnoDB DEFAULT CHARACTER SET = utf8;

CREATE TABLE IF NOT EXISTS `tb_alert_info` (
```

```
  `id` bigint(20) unsigned NOT NULL AUTO_INCREMENT,
  `instance` varchar(100) COLLATE utf8_unicode_ci NOT NULL DEFAULT '' COMMENT 'alert host',
  `ipaddr` varchar(20) COLLATE utf8_unicode_ci NOT NULL DEFAULT '' COMMENT 'alert ip',
  `alert_name` varchar(100) COLLATE utf8_unicode_ci NOT NULL DEFAULT '' COMMENT 'alert name',
  `alert_user` varchar(50) COLLATE utf8_unicode_ci NOT NULL DEFAULT '' COMMENT 'alert user',
  `limit_value` varchar(10) COLLATE utf8_unicode_ci NOT NULL DEFAULT '' COMMENT 'alert limit value',
  `current_value` varchar(20) COLLATE utf8_unicode_ci NOT NULL DEFAULT '' COMMENT 'current alert value',
  `description` varchar(1000) COLLATE utf8_unicode_ci NOT NULL DEFAULT '' COMMENT 'description',
  `generator_url` varchar(500) COLLATE utf8_unicode_ci NOT NULL DEFAULT '' COMMENT '',
  `status` varchar(10) COLLATE utf8_unicode_ci NOT NULL DEFAULT '' COMMENT 'status',
  `update_time` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  `create_time` TIMESTAMP NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY(`instance`, `alert_name`,`create_time`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
```

Other Installation Requirements

Jump Host Requirements

Software requirements:

- git
- docker (version >= 1.13)
- golang (version >= 1.12)
- kubectl CLI

Network Requirements

- internet connectivity

Bare Metal Node Requirements

Execution Requirements (Bare Metal Only)

Installation High-Level Overview

Bare Metal Deployment Guide

Install Bare Metal Jump Host

The prerequisites software should be ready.

Creating a Node Inventory File

N/A

Creating the Settings Files

Copy the kubeconfig file of IEC cluster to directory /root/.kube/config.

Clone the AIEdge git repo and edit the configuration file (interface_conf) by setting:

- hostname of the node in edge cluster that the ote-stack platform will be deployed to
- docker repository for pulling images of ote-stack components
- informations of harbor repository such as admin password and domain

```
git clone https://gerrit.akraino.org/r/aiedge.git
cd aiedge
git submodule init
git submodule update
cd ote-stack/deployments
vim interface_conf
```

Running

Prerequisite

Compile cluster controller, controller manager and shim components following below scripts. Then docker push the output image to a docker repository which is set in the interface_conf file.

```
bash build/build.sh build_image
```

Other components are not source code released currently but released as docker images, and they can be pull from docker hub. Make sure that the docker repository set in the configuration file have all docker images required by ote-stack platform.

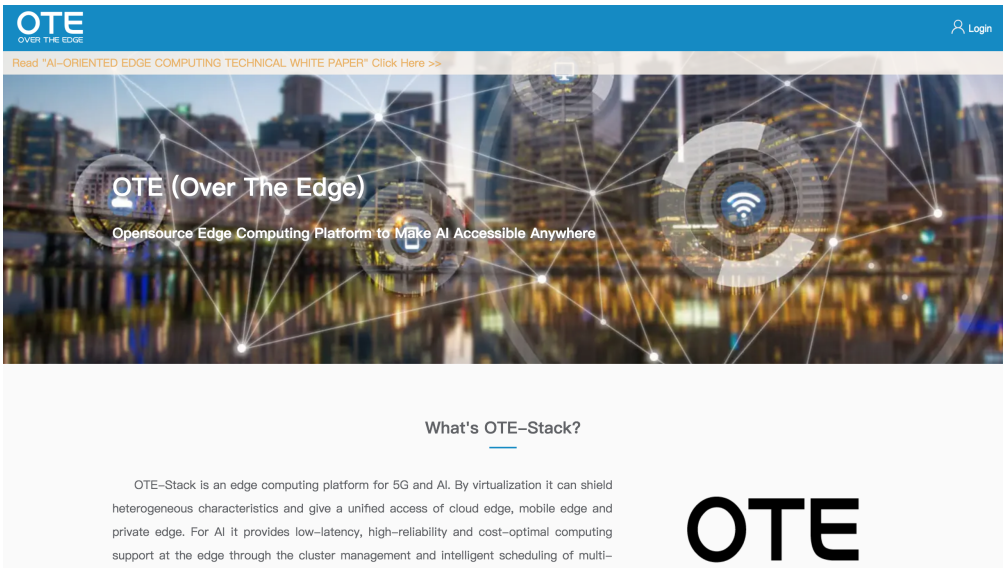
Deploy ote-stack on one IEC cluster

Simply start the installation script with default parameters in the same directory:

```
# create yaml file based on interface_conf
sh create_yml.sh start
# apply yaml file to cluster
sh apply_yml.sh start
```

All components of ote-stack will be installed on the IEC edge. But some components can be replaced with external ones outside the cluster through changing the parameter of yaml file. For example, the open-api server can connect to an existing mysql server by modifying the parameters `mysqlHost` and `mysqlPort` in file deployments/open-api/open-api.yml and commenting out the line to deploy mysql.

After executing the installation script, the ote-stack web platform will be deployed on the node1_name set in the configuration file and expose port 8995. Now you can open the website through URL http://<node1_name_ip>:8995/ to use ote-stack. Below is the the index page of OTE.



When ote-stack runs, a super administrator will be created. The username and password can be modified by changing parameters `superAdminName` and `superAdminPassword` in file deployments/open-api/open-api.yml. Note that the initial username and password of admin, only works for the first time when ote-stack starts.

Enable edge autonomy feature in your cluster

1. Deploy EdgeHub

For k8s cluster

- Modify the kubelet-bootstrap.conf file of kubelet, the server field points to the listening address of EdgeHub's edge server, the default is: https://127.0.0.1:8778.And kubelet should add startup parameters: --kube-api-content-type=application/json
- Modify the kube-proxy.conf file of kube-proxy, the server field points to the listening address of EdgeHub's edge server, the default is: https://127.0.0.1:8778. And kube-proxy adds startup parameters: --kube-api-content-type=application/json
- Copy the certificate of k8s apiserver, including three files ca.pem, kube-apiserver.pem, kube-apiserver-key.pem. Rename kube-apiserver.pem to edge-server.pem and rename kube-apiserver-key.pem to edge-server-key.pem
- Copy the kubeconfig file to the working directory, modify the server field of the kubeconfig file to point to the listening address of the EdgeHub load balancer, the default is: https://127.0.0.1:6888.
- Start EdgeHub./ote_edgehub k8s -v 2 --kube-config kubeconfig --init-server ${master_ip}:${master_port}

For k3s cluster

- Set k3s agent startup parameters

```
--server=https://127.0.0.1:8778
--kubelet-arg=kube-api-content-type=application/json
--kube-proxy-arg=kube-api-content-type=application/json
```

- Create the ssl directory under the working directory, and copy the k3s certificate files server-ca.crt, server-ca.key and token to the ssl directory. Rename server-ca.crt to edge-server.pem, and rename server-ca.key to edge-server-key.pem.
- Copy the kubeconfig file to the working directory, modify the server field of the kubeconfig file to point to the listening address of the EdgeHub load balancer, the default is: https://127.0.0.1:6888.
- Start EdgeHub

```
./ote_edgehub k3s -v 2 --kube-config kubeconfig --init-server ${master_ip}:${master_port}
```

2.Deploy EdgeController

- First need to create EdgeNode CRD

```
kubectl apply -f edgenode-crd.yml
```

- Run edge-controller on a master node of the k8s or k3s cluster

```
./ote_edgecontroller -v 2 -kubeconfig /root/.kube/config
```

Add a new cluster to ote-stack for management (option)

If there is another cluster that needs to be added to the ote-stack, some components should be deployed on the edge to connect with root cluster:

- tiller/ tiller-proxy
- ote-shim/ ote-cc
- node-agent/ nodes-server/ prometheus
- exporters like node-exporte (optional)
- fluent-bit (optional)

Virtual Deployment Guide

Standard Deployment Overview

Please refer section Bare Metal Deployment Guide for deployment, the installation can run successfully in virtual hardware too.

Snapshot Deployment Overview

N/A

Special Requirements for Virtual Deployments

Install Jump Host

N/A

Verifying the Setup - VMs

N/A

Upstream Deployment Guide

Upstream Deployment Key Features

- Provide the execution and network infrastructure for installing ote-stack and running the AI application.

Special Requirements for Upstream Deployments

Please refer the section "Pre-Installation Requirements" in wiki IEC Type1&2 Installation Guide for R2 and make sure the environment requirements met.

Scenarios and Deploy Settings for Upstream Deployments

Please prepare three arm servers and build a edge cluster according to wiki IEC Type1&2 Installation Guide for R2.

Including Upstream Patches with Deployment

N/A

Running

N/A

Interacting with Containerized Overcloud

N/A

Verifying the Setup

Verify manually:

- Login to the otestack web platform and create a simply application on the website
- Check the deployment status of application using kubectl cli on the IEC cluster

Or Run installation and verify scripts through CI/CD.

Developer Guide and Troubleshooting

Utilization of Images

Post-deployment Configuration

N/A

Debugging Failures

N/A

Reporting a Bug

N/A

Uninstall Guide

```
# remove all components deployed in edge cluster
sh apply_yml.sh stop
```

MySQL data will be persisted on disk /home/work/ote/mysql-data/. You can use command `rm` to clean up the residual data.

Troubleshooting

Error Message Guide

N/A

Maintenance

Blue Print Package Maintenance

Software maintenance

N/A

Hardware maintenance

N/A

Blue Print Deployment Maintenance

N/A

Frequently Asked Questions

N/A

License

/*

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
*/

References

- Apache License 2.0: https://www.apache.org/licenses/LICENSE-2.0
- AI Edge Wiki: Video Security Monitoring Release 5 Documentation

Definitions, acronyms and abbreviations