This course introduces setup and access procedures for AWX Version 21.3.0.

AWX is an environment that allows for more precise control and execution of Ansible via a web application.

It enables automated task execution, provides a REST API, and separates task execution and editing permissions.

It is positioned as an open-source version of the commercial Red Hat Automation Controller project.

First, as prerequisites for deploying the AWX application, install Docker, Python, Ansible, and other software on a virtual machine in the practice environment.

Then, build the AWX application using the installer (or create the environment using a shortcut).

The AWX application is configured as a Docker container within the killercoda VM.

Next, create three CentOS nodes to be managed by AWX.

These nodes are also configured as containers using docker-compose.

Log in to the AWX web application and run a sample job.

Step 1: Install AWX prereq package #1 (docker)

This section explains how to install Docker, one of the prerequisite software. The buildkit environment for Killercoda's Ubuntu environment is not yet fully developed, so please reinstall it.

Setting up a Japanese environment

sudo timedatectl set-timezone Asia/Tokyo

sudo localedef -i ja\_JP -f UTF-8 ja\_JP.UTF-8

export LANG=ja\_JP.UTF-8

date

① Uninstall the old Docker environment

sudo apt update

sudo apt -y remove --purge docker docker-engine docker.io containerd runc

sudo apt -y autoremove

sudo apt -y autoclean

② Update the software repository

sudo apt update &&\

sudo apt-get -y install ca-certificates curl gnupg lsb-release &&\

sudo mkdir -m 0755 -p /etc/apt/keyrings &&\

curl -fsSL https://download.docker.com/linux/ubuntu/gpg |\

sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg &&\

echo \

"deb [arch=$(dpkg --print-architecture) \

signed-by=/etc/apt/keyrings/docker.gpg] \

https://download.docker.com/linux/ubuntu \

$(lsb\_release -cs) stable" | \

sudo tee /etc/apt/sources.list.d/docker.list > /dev/null &&\

sudo apt -y update

sudo apt -y autoremove

sudo apt -y autoclean

③ Installing Docker

export DEBIAN\_FRONTEND=noninteractive

sudo apt-get -y install docker-ce docker-ce-cli \

containerd.io docker-buildx-plugin \

docker-compose-plugin

If there are any conflicting configuration files, as shown below, proceed with the default (=N). conflict\_package\_setting.jpg

④ Verify Docker is running

sudo docker run hello-world

The official procedure for installing Docker on an Ubuntu environment is here.

① Install prerequisite packages

sudo apt-get update

sudo apt-get -y install python3 python3-pip virtualenv docker-compose

② Configure a Python virtual environment

virtualenv ~/venv\_ansible

. ~/venv\_ansible/bin/activate

③ Install ansible and docker-compose

Install the required packages for installation: ansible and docker-compose.

Install setuptools\_scm, as it is a prerequisite.

pip install pip --upgrade

pip install setuptools\_scm setuptools

pip install ansible

ansible --version

Install AWX using ansible.

① Installing prerequisite packages (git)

sudo apt -y install git

② Copying the AWX project

Explicitly specify version 21.3.0 and copy the project.

git clone -b 21.3.0 https://github.com/ansible/awx.git ~/awx

③ Installing AWX (Running the playbook)

The standard procedure runs the make command to create an image, create a container, and build the frontend (React).

Since this can be time-consuming, we have provided a shortcut.

cd ~/awx

make SHELL=/bin/bash PYTHON=python3 docker-compose-build

make SHELL=/bin/bash PYTHON=python3 COMPOSE\_UP\_OPTS=-d docker-compose

docker exec tools\_awx\_1 make clean-ui ui-devel

The above task takes a long time, so using a pre-built image can shorten the time somewhat (still about 15 minutes).

cd ~/awx

docker pull mirrorcity/awx213:1.2

docker tag mirrorcity/awx213:1.2 ghcr.io/ansible/awx\_devel:HEAD

docker rmi mirrorcity/awx213:1.2

make SHELL=/bin/bash PYTHON=python3 COMPOSE\_UP\_OPTS=-d docker-compose

docker exec tools\_awx\_1 tar xf /awxui\_21.3.tar.gz

cat << EOF > startnpm.sh

#!/bin/sh

export TERM=dumb

yum -y --quiet install lsof

env NODE\_OPTIONS=--max-old-space-size=3076 npm --prefix awx/ui --loglevel warn start | cat &

echo wait to open listen port 8013...

until lsof -i -n -P | grep :8013 ; do printf . ; sleep 1 ; done

echo "open! [8013]"

sleep 1

EOF

chmod 755 ./startnpm.sh

docker exec --user root tools\_awx\_1 ./startnpm.sh

After installation is complete, initialize the administrator user password. (Set a password of your choice.)

docker exec -ti tools\_awx\_1 awx-manage changepassword admin

④ Check the Docker environment after installation

You can see that the following six containers are configured:

tools\_awx\_1 (Application server: AWX main unit)

tools\_postgres\_1 (DB server)

tools\_redis\_1 (Cache server)

tools\_receptor\_1 (Replication server)

tools\_receptor\_2 (Replication server)

tools\_receptor\_hop (Replication server)

The awx installer PLAYBOOK generates a Docker-compose file at ~/awx/tools/docker-compose/\_sources/docker-compose.yml . You can use this file to check operation and start and stop the system.

docker-compose -f ~/awx/tools/docker-compose/\_sources/docker-compose.yml ps

Let's take a break from building AWS and set up the managed endpoint as a Docker container.

① Extract and place the files you brought with you

cd ~/

gzip -cd my\_assets3.tar.gz | tar tvf -

gzip -cd my\_assets3.tar.gz | tar xf -

cp -iv ~/awx-asset.git/ansible.cfg ~/.ansible.cfg

perl -i -ple 's@^inventory.\*$@inventory = ~/awx-asset.git/my-inventory@' ~/.ansible.cfg

② Change to the working directory and create an endpoint container

Create an SSH key pair to access the endpoint you will create.

cd ~/endpoint-docker/

ssh-keygen -t ed25519 -N "" -f my-ssh-keypair -C "ssh-key to access node01-node03"

cp -iv my-ssh-keypair ~/.ssh/ssh-keypair

cp -iv my-ssh-keypair.pub ssh-keypair.pub

Let's check the container information (Dockerfile) for the container we're about to create.

We've created a user and copied the SSH key to the container so that it can connect using epuser.

grep -e ^ -e epuser -e ssh-keypair.pub Dockerfile

To include the endpoint we're creating in the same network as tools\_awx\_1, we'll update the docker-compose file with the network information.

AWXNET=$(docker inspect --format '{{range $k,$v := .NetworkSettings.Networks}}{{println $k}}{{end}}' tools\_awx\_1 2