Installation

Instructions for MANUAL installation Airship+Tungsten Fabric using the Regional Controller and the TF Blueprint

Requirements:

- For Regional Controller host: AWS instance ***t2.medium*** or any virtual/baremetal node with 2CPU and 4GB Memory (OS: Ubuntu Xenial 16.04)
- For Airship+Tungsten Fabric host: AWS instance ***m5.4xlarge*** or any virtual/baremetal node with 16CPU and 64GB Memory (OS: Ubuntu Xenial 16.04)

Both nodes must be available by ssh.

This document describes detailed manual installation procedure. As an option you can use Automatic deployment with ansible to get the same environment as used for CICD validation.

Overview

Akraino Regional Controller is necessary part of release 2 deployment procedure. It's Akraino approved blueprint which is common for all of release 2 blueprints and which is using for Edge Site, Blueprint and POD deployment.

After creating the POD Regional Controller creates WORKFLOW that initiates remote installation Airship with TungstenFabric.

More information about Regional Controller:

- The Regional Controller's Object Model (it helps you to figure out what is Blueprint, Edge Site, POD, Workflow)
- How to write Blueprints and Workflows
- How to load objects into the Regional Controller
- Frequently Asked Questions

If you already have Regional Controller you can use it for deployment and if you don't have it you can install it following the instruction bellow

Installing the Regional Controller

You can use any machine or VM dedicated for this purpose. See instructions on how to start the regional controller.

https://wiki.akraino.org/display/AK/Starting+the+Regional+Controller

After you have a working Regional Controller you have to login on it and follow the steps bellow

Steps to manual installation of TungstenFabric Blueprint

All this steps must be done on Regional Controller

Clone the nc/tf repository using

```
git clone https://gerrit.akraino.org/r/nc/tf
```

Setup ssh keys and put them on web-server

Regional Controller goes to the remote node by ssh, so it needs ssh private key. It can be provided as http URL. (It's not secure for production, it's only OK for the demo).

Put ssh private key and script deploy.sh on some web server. Ssh public key must be written to the .ssh/authorized_keys on remote node.

Hint: python provisional web server can be used on the localhost. Use ***python3 -m http.server***

Edit the file *setup-env.sh*

Update all the environment variables

Mandatory variables:

- RC_HOST - Regional Controller IP
- NODE - IP address of the node where airhip-in-a-bottle with TungtenFabric would be deployed
- BASE_URL (URL to download ssh key and deploy.sh script)

(the login/password shown here are the built-in values and do not need to be changed, if you have not changed them on the Regional Controller):

example of setup-env.sh

```
#Regional Controler (ip address or FQDN)
export RC_HOST=35.181.44.122
#Regional Controler credentials
export RC_USER=admin
export RC_PW=admin123


#Node for airship remote deployment (ip address or FQDN)
export NODE=52.47.109.251

#ssh user for aisrship remote deployment
export SSH_USER=ubuntu
#File with private ssh key (public key must be added into the node for auth)
export SSH_KEY=ssh_key.pem

#web server for downloading scripts and ssh key
#simpliest way is running "python3 -m http.server" in current directory
export BASE_URL=http://172.31.37.160:8000

#repo URL and branch for treasuremap with tungstenfabric
export REPO_URL=https://github.com/progmaticlab/treasuremap.git
export REPO_BRANCH=master
```

Generate yaml files from templates

```
source setup-env.sh
cat objects.yaml.env | envsubst > objects.yaml
cat TF_blueprint.yaml.env | envsubst > TF_blueprint.yaml
```

As the result you would get correct yaml files objects.yaml and TF_blueprint.yaml

example of yaml files

```
ubuntu@ip-172-31-37-160:/opt/akraino-tf$ cat objects.yaml
hardware:
  AWS_instance:
    uuid: 5367a004-71d4-11e9-8bda-0017f00dbff7
    description: AWS Ubuntu Xenial for the TF Blueprint
    yaml:
      todo: AWS instance with >=8 VCPU and >=32GB RAM

edgesites:
  TF_Edgesite:
    description: The demo singlenode TF cluster
    nodes: [ node1 ]
    regions: [ 00000000-0000-0000-0000-000000000000 ]

nodes:
  node1:
    hardware: AWS_instance
    yaml:
      oob_ip: 52.47.109.251

ubuntu@ip-172-31-37-160:/opt/akraino-tf$ cat TF_blueprint.yaml
blueprint: 1.0.0
name: TF Edge Cloud
version: 1.0.0
description: This Blueprint defines an instance of the TF Edge Cloud
yaml:
  # Required hardware profiles (can match on either UUID or name)
  # Note: UUIDs would likely require a global registry of HW profiles.
  hardware_profile:
    or:
      - { uuid: 5367a004-71d4-11e9-8bda-0017f00dbff7 }
  workflow:
    # Workflow that is invoked when the POD is created
    create:
      url: 'http://172.31.37.160:8000/deploy.sh'
      components:
        # SSH key for remote installation
        - 'http://172.31.37.160:8000/ssh_key.pem'
      input_schema:
        rc_host: { type: string }
        ssh_user: {type: string }
        node: {type: string }
        repo_url: {type: string }
        repo_branch: {type: string }
```

Clone the *api-server r*epository (optional)

(If you are working on Regional Controller this repo is should be already presented in /opt/api-server/scripts)

This provides the CLI tools used to interact with the Regional Controller. Add the scripts from this repository to your PATH:

```
git clone https://gerrit.akraino.org/r/regional_controller/api-server
export PATH=$PATH:$PWD/api-server/scripts
```

Load the objects

Load objects defined in ***objects.yaml*** into the Regional Controller using:

```
rc_loaddata -H $RC_HOST -u $RC_USER -p $RC_PW -A objects.yaml
```

Load the blueprint

Load the blueprint into the Regional Controller using:

```
rc_cli -H $RC_HOST -u $RC_USER -p $RC_PW blueprint create TF_blueprint.yaml
```

Get and export UUIDs

Get the UUIDs of the edgesite and the blueprint from the Regional Controller using:

```
rc_cli -H $RC_HOST -u $RC_USER -p $RC_PW blueprint list
rc_cli -H $RC_HOST -u $RC_USER -p $RC_PW edgesite list
```

These are needed to create the POD. You will also see the UUID of the Blueprint displayed when you create the Blueprint in step 8 (it is at the tail end of the URL that is printed).
Set and export them as the environment variables ESID and BPID.

```
export ESID=<UUID of edgesite in the RC>
export BPID=<UUID of blueprint in the RC>
```

Generate *POD.yaml*

```
cat POD.yaml.env | envsubst > POD.yaml
```

As the result you get correct POD.yaml

example of POD.yaml

```
name: My_TF_Edge_Cloud_POD
description: Put a description of the POD here.
blueprint: 76c27993-1cc3-471d-8d32-45f1c7c7a753
edgesite: 52783249-45e2-4e34-831d-c46ff5170ae5
yaml:
  rc_host: 35.181.44.122
  node: 52.47.109.251
  ssh_user: ubuntu
  repo_url: https://github.com/progmaticlab/treasuremap.git
  repo_branch: master
```

Please check that file POD.yaml contains correct data.

Create the POD

Create the POD using

```
rc_cli -H $RC_HOST -u $RC_USER -p $RC_PW pod create POD.yaml
```

This will cause the POD to be created, and the ***deploy.sh*** workflow script to be
run on the Regional Controller's workflow engine. This in turn will login to remote node by ssh
and install airship+ tungstenfabric demo on it

Checking POD status

If you want to monitor ongoing progess of the installation, you can issue periodic calls to monitor the POD with:

```
rc_cli -H $RC_HOST -u $RC_USER -p $RC_PW pod show $PODID
```

where $PODID is the UUID of the POD. This will show all the messages logged by the workflow, as well as the current status of the workflow. The status will be WORKFLOW while the workflow is running, and will change to ACTIVE if the workflow completes successfully, or FAILED, if the workflow fails.

Uninstall

Uninstall Regional Controller

As we using one-time AWS instances they can be just removed with AWS console or other tools which were used for creating (ansible, terraform, etc).

In other cases following comands can be used for manual cleanup procedure.

Deleting POD from Regional Controller

Deleting POD from Regional Controller

```
rc_cli -H $RC_HOST -u $RC_USER -p $RC_PW pod delete $PODID
```

Deleting Blueprint from Regional Controller

Deleting Blueprint from Regional Controller

```
rc_cli -H $RC_HOST -u $RC_USER -p $RC_PW blueprint delete $BPID
```

Uninstall Regional Controller itself

Uninstall Regional Controller

```
sudo docker stop $(docker ps -aq)
sudo docker rm $(docker ps -aq)
sudo docker rmi $(docker images -q)
sudo rm -rf /opt/api-server/
sudo rm -rf /opt/akraino-tf/
```

Uninstall Airship

Airship-in-a-bottle doesn't have any tools for installation. Moreover according the documentation it's not recommended to use one virtual instance twice after fail. It's better to remove the failed instance and create a new one for reinstalling.

So the best way to uninstall airship-in-a-bottle it's removing Airship+Tungsten Fabric host via AWC console.