Créer un cluster privé avec DeskPi Super6c et Ansible

Table of Contents

DeskPi Super6C est la carte cluster Raspberry Pi, une carte mini-ITX de taille standard à mettre dans un boîtier avec jusqu'à 6 modules de calcul **RPI CM4**.

Préparation du système

Tout d'abord, télécharger la dernière image Raspberry Pi OS 64 bits à partir de : raspios_arm64-2021-11-08

Les étapes suivantes doivent être répétées sur tous les modules CM4.

Si les modules CM4 n'ont pas d'**eMMC** à bord, ce sera facile, il suffit de flasher le système d'exploitation Raspberry Pi sur la carte TF en utilisant l'une des méthodes décrites ici pour graver l'image sur la carte SD, puis de les insérer dans l'emplacement de la carte..

Si les modules CM4 ont **eMMC** à bord, il faut flasher le système d'exploitation Raspberry Pi sur le module **EEPROM** en utilisant l'une des méthodes décrites ici, le lecteur SSD et la carte TF peuvent être un stockage de masse externe.

Avant que le CM4 soit prêt à fonctionner, il faut activer les ports USB (il n'est pas besoin d'exécuter cette étape lorsqu'on utilise la dernière version).

Dans le dossier `/boot` dans la carte TF , rechercher le fichier `config.txt` et ajouter ce qui suit :

```
dtoverlay=dwc2,dr_mode=host
```

Cela activera les ports USB au démarrage, mais si on laisse accidentellement le micro USB branché, on risque de rencontrer des problèmes. Avant de commencer le premier démarrage, il faut s'assurer que tout est débranché. Démarrer puis brancher la souris ou le clavier.

Pour activer la connexion distante, créer un nouveau fichier nommé `ssh` sur la carte TF/dossier de démarrage, puis insérer la carte TF dans l'emplacement de la carte sur la carte Super6C.

Connecter l'alimentation à la prise CC ou à l'aide de l'alimentation ATX. (Max. 24V/6.15A)

Connecter le câble HDMI pleine taille au super6C, l'autre tête se connecte à un téléviseur ou moniteur.

Connecter le clavier et la souris au port USB2.0 de la carte.

Connecter le câble Ethernet à ETH1.

Appuyer sur le bouton **PWR_BTN** pour démarrer tous les CM4.

Pour se connecter à Raspberry Pi utiliser le **nom d'utilisateur:mot de passe pi:raspberry**

Installation de l'écosystème

Préparation des environnements kubernetes

Kubernetes est une solution de conteneurisation open-source, simplifiant le déploiement, la mise à l’échelle, et la gestion en général, d’applications conteneurisées.

Préparation des environnements ansibles

Pour utiliser **Ansible** pour gérer le cluster Raspberry Pi, il faut mettre à niveau la version python de 3.7 à 3.8 en procédant comme suit :

Supprimer l'ancienne version de python

```
sudo apt supprimer ansible
sudo apt purge -y python2.7-minimal
```

Mettre à jour le système.

```
sudo apt update
sudo apt upgrade –y
```

Installer les dépendances.

```
sudo apt-get install -y build-essential tk-dev libncurses5-dev \
libncursesw5-dev libreadline6-dev libdb5.3-dev libgdbm-dev libsqlite3-dev \
libssl-dev libbz2-dev libexpat1-dev liblzma-dev zlib1g-dev libffi-dev
```

Télécharger le code source et compiler.

version=3.8.5 wget https://www.python.org/ftp/python/$version/Python-$version.tgz tar zxf Python-$version.tgz cd Python-$version ./configure –enable-optimizations make -j4 sudo make altinstall

Ceci installe Python dans /usr/local/bin

Installer **ansible**, on peut utiliser apt ou un gestionnaire de packages de distribution particulier, mais en faisant cela, on risque de ne pas obtenir la dernière version disponible. Dans certains cas, c'est une bonne idée, pour que le système soit maintenu stable et que les packages aient été suffisamment testés, mais dans des technologies comme **Ansible**, qui évolue toujours si rapidement, il est recommandé d'utiliser la dernière version, qui présente des différences d'utilisation des modules.

Vérifier la version installée à l'aide d'apt dans Raspbian.

```
sudo apt list ansible
pip3 install ansible --user
```

L'installation d'**Ansible** à l'aide de l'option **–user** fera que le binaire sera disponible directement sous l'accueil utilisateur, et non dans le répertoire racine bin (/bin,/usr/bin..)

Il est alors recommandé d'ajouter le path `$user_home/.local/bin` à l'environnement **PATH**.

Pour récupérer le chemin d'installation du binaire utiliser la commande `which ansible` .

Maintenant qu'on a installé **Ansible**, il faut définir une configuration de base.

Créer un répertoire pour un premier projet où on va définir les différents fichiers de configuration et les playbooks **Ansible**

```
mkdir AnsibleComment
```

Il faut créer un premier fichier de configuration **Ansible**. Cela garantira qu'on utilise le bon fichier d'inventaire et certaines options spécifiques pour ce projet. Dans ce cas, on utilisera le fichier d'inventaire dans le répertoire local et on connectera au serveur distant en tant que root.

De plus, il faut désactiver la vérification de la clé ssh afin de pouvoir ses connecter aux serveurs sans faire la première poignée de main avec la clé ssh (est un risque pour la sécurité, mais celas aidera à détruire et à déployer plus rapidement les conteneurs et les machines virtuelles à l'avenir)

```
cat > /home/pi/Ansible/ansible.cfg<< 'EOF'
[defaults]
```

```
host_key_checking=False
deprecation_warnings=False
inventory=./inventory
remote_user=pi
private_key_file=/home/pi/.ssh/id_rsa
EOF
```

L'utilisation de l'attribut **-version** permet de s'assurer qu'on utilise le bon fichier de configuration :

```
ansible --version
 ansible [core 2.11.12]
 config file = /home/pi/Ansible/ansible.cfg
 configured module search path =
['/home/pi/.ansible/plugins/modules', '/usr/share/ansible/
plugins/modules']
 ansible python module location =
/home/pi/.local/lib/python3.7/site-packages/ansible
 ansible collection location =
/home/pi/.ansible/collections:/usr/share/ansible/collections
 executable location = /home/pi/.local/bin/ansible*
 python version = 3.7.3 (default, Jan 22 2021, 20:04:44) [GCC
8.3.0]
 jinja version = 2.10
 libyaml = True
```

Créer le fichier d'inventaire avec la liste des hosts:

```
cat > /home/pi/Ansible/inventory<< 'EOF'
[all]
node[1:4]

[webservers]
node1
node2
node3
node4
EOF
```

On peut alors répertorier tous les hôtes de l'inventaire

```
ansible all --list-hosts

hosts (4):
 node1
 node2
```

```
node3
node4
```

Mais si on essaye la commande `ansible all -m ping`, il y aura des erreurs, car on ne peut pas réellement se connecter aux serveurs en utilisant les clés publiques SSH, et avant cela, il faut créer un fichier appelé `ping.yml`:

```
cat > /home/pi/Ansible/ping.yml<< 'EOF'
- name: Test Connection to each CM4 device
  hosts: all
  tasks:
         - action: ping
EOF
```