Plensor installation & instruction manual

To setup the edge computer for our specific use case, the following setup types are needed

Contents

Automatic setup: flashing image

Create a disk image of our 'original setup edge computer' eMMC

- Connect to the Edge Computer: Log into your edge computer that has the configured eMMC. You can do this directly or via SSH if remote access is configured.
- Identify the eMMC Device: Use the lsblk or fdisk -l command to identify the eMMC device. It's typically listed as /dev/mmcblk0 or similar.
    - *sudo fdisk -l*
- Create the Disk Image: Use the dd command to create an image of the eMMC. Replace /dev/mmcblk0 with the actual device identifier of your eMMC:
  sudo dd if=/dev/mmcblk0 of=/media/plense/9C33-6BBD/emmc_clone.img bs=4M status=progress conv=notrunc
    - conv=notrunc is a flag to ensure that if there is already an image in the goal output file, it will not be overwritten
    - if means input file (your eMMC device).
    - of means output file (the path where you want to save the image). In this case, it is the mounted SD card
    - bs=4M sets the block size to 4MB for faster copying.
    - status=progress will show the progress of the operation.
- Compress the Image (optional): To save space and speed up future transfers, you can compress the image:
  gzip /media/plense/9C33-6BBD/emmc_clone.img

Flashing the image to a new edge computer

THESE STEPS HAVE NOT BEEN TESTED YET

- Prepare the New Device: Ensure the new edge computer is powered off, and you have access to its eMMC via an external eMMC reader/writer or direct connection, depending on the device's design.
- Connect the eMMC to Your Computer: Use an eMMC to USB adapter or similar device to connect the eMMC to a computer where you will perform the flashing.
- Write the Image: First, decompress the image if you compressed it. Then use dd to write it to the new eMMC. Ensure you identify the correct device name (/dev/sdx) for the connected eMMC.
  gunzip -c /path/to/your/image/emmc_clone.img.gz > /path/to/your/image/emmc_clone.img
  sudo dd if=/path/to/your/image/emmc_clone.img of=/dev/sdx bs=4M status=progress
- Be very careful with dd as selecting the wrong output file device (of=) can overwrite any disk in the system.
- Ensure Data Integrity: After the cloning process, it's a good practice to run sync to ensure that all data is written to the disk:
  sync
- Remove the eMMC: Safely eject the eMMC from your computer and install it back into the new edge computer.

Manual steps after flashing the image

Local save of the hostname

See: Setting up the hostname

Bring needed connections up or downls

- For example, the hotspot. The hotspot is turned off on the eMMC image because the image was created via SSH over the Plense tech WiFi network and thus the hotspot was not turned on at the time the image was created.
- Do so using the NetworkManager command line interface nmcli

Manual setup

Follow the following steps to manually set up a compute module or Raspberry Pi.

General setup

To setup the Pi for serial communication and communication to the cloud several steps need to be taken. These are as follows:

- Install OS and update it
- Ensure the user is called plense (this is important for running our Greengrass Docker container correctly later on)
    - If this is not the case, follow these steps: https://thepihut.com/blogs/raspberry-pi-tutorials/how-to-change-the-default-account-username-and-password
- Ensure auto login is turned on. Go to Preferences > Settings > toggle auto login
- Keyboard settings: US
- Install VSCode: *sudo apt install code*
- Change the hostname to specified hostname in Preferences > Raspberry Pi configuration > hostname and reboot to take effect.
- Log in to GitHub and log in in the VSCode terminal using:
    - *git config --global user.email xxx@plense.tech*
    - *git config --global user.name yyy*
    - Also install python through VSCode
- *Sudo raspi-config*
    - Go to interface, Serial port, select no on shell and yes on enabling serial port hardware.
    - In interface also toggle enable for SSH
    - Configure rest of SSH (needed for Greengrass installation later on)
        - Identify IP address using one of the following methods: https://www.raspberrypi.com/documentation/computers/remote-access.html
            - Enter in cmd: *hostname -I*
        - Document IP address in the following document https://plensetechnologiesbv.sharepoint.com/:x:/s/ResearchDevelopment/EeD9Rxmg3lpHn48-mDt1LIEBLMwetRaL7h52h4b4dyZnyA?e=2zgrFG
        - On development computer, which should be connected to the same network, enter the following in cmd: *ssh username@pi-ip-adress*
            - Eg. for plensepi00003: *ssh plense@192.168.0.173*
        - On development computer cmd, enter password for plense user when asked
        - On development computer, check ssh connection by entering in cmd: *ssh python --version* which should print the python version in the Raspberry Pi terminal
    - Reboot the computer
- Go to command terminal and type: *sudo nano /boot/firmware/config.txt*
  If it cannot be found, the file might be located in /boot/config.txt
    - Add at the bottom *enable_uart = 1*
    - And then at the bottom add *dtoverlay=disable-bt*
    - Press ctrl+O, enter and then ctrl-X

    - Reboot

Enable local messaging over ethernet to the FLIR heat camera

- Enable i2c in sudo raspi-config
    - This might cause the lightdm system service to stop working. Then reinstall it using
        -
- *sudo ip addr add 169.254.182.240/16 dev eth0*
- Pinging to the connected camera should return packets!
- To implement this permanently (so also after reboot), open dhcpcd config file with sudo nano /etc/dhcpcd.conf and add the following at the bottom and reboot
    - *interface eth0*
    - *static ip_address=169.254.182.240/16*
- Ensure the dhcpcd is installed as a system service thus starts automatically at reboot

Add static ethernetport using NetworkManager, do not use dhcpcd and networking system services

- sudo nmcli con add type ethernet con-name flir-static
- sudo nmcli con modify flir-static ipv4.addresses 169.254.182.242/16
- sudo nmcli con modify flir-static ipv4.method manual
- sudo nmcli con up flir-static
- check if the flir-static ethernet connection has as property autoconnect

Verify ethernet connection with *ping 169.254.182.241*

Setup the pi_id for later use as metadata in containerized programmes

- Make a file (no extension) with only plensepiXXXXX in it in the etc folder like
    - *sudo touch /etc/hostname*
- *sudo nano /etc/hostname* and then enter the hostname id (pi_id), save with Ctrl+S, exit with Ctrl+X

Metadata folder which is needed for the sensor components

- sudo mkdir /home/plense/metadata
- Also put a hostname file in there:
- sudo touch /home/plense/metadata
- sudo nano /home/plense/metadata
- Place pi_id in that file and save

GSM SIM connection set-up for the edge computer

- Linux system service NetworkManager should be installed, enabled and running
    - Verify this by *sudo systemctl status NetworkManager*
- Create a gsm connection that is managed by NetworkManager with the correct configuration:
    - *nmcli con add type gsm ifname '*' con-name 'gsm-connection' apn '<APN_NAME>'*
    - apn is the Access Point Name and is defined per telecom provider. In the case of KPN, it is portalmmm.nl, thus for our config: *nmcli con add type gsm ifname '*' con-name 'my-gsm-connection' apn 'portalmmm.nl'*
        - Odido: smartsites.t-mobile
    - You may need to provide additional parameters such as username, password, and pin code (if required by your mobile carrier). In our case, add the gsm pin.
    - *nmcli con modify 'my-gsm-connection' gsm.username 'your_username' gsm.password 'your_password'*

- *nmcli con modify 'my-gsm-connection' gsm.pin 'your_pin'*
- Enable autoconnect: *nmcli con modify ‘gsm-connection’ connection.autoconnect yes*
- Bring the connection up: *nmcli con up ‘gsm-connection’*
- Verify connection status: *nmcli con show*

Greengrass and Docker installation & deployment

After this step we need to set up the Greengrass side. Note that not all AWS regions support AWS IoT Greengrass. For a tutorial see: https://docs.aws.amazon.com/greengrass/v2/developerguide/getting-started.html. We will now discuss the steps required with our specifications:

Prerequisites

Some prerequisites from the Greengrass installation tutorial are fulfilled already, which are:

- The AWS account is already available, which is the masterpi IAM user.
- The Raspberry Pi is set up with the Raspbian OS if the steps of the Section "The Edge - Raspberry Pi: Setup" are followed.
- Python 3.5 or later is already installed on the Raspberry Pi for all users. This could be checked by entering the following in cmd, which should return the python version: *python --version*

Execute the following steps to fulfill the prerequisites for the Greengrass installation on the Raspberry Pi:

- Ensure the Raspbian OS is booting in 64-bit modus
    - Enter: *uname -m*
      It should return aarch64
    - If it doesn't, add the line *arm_64bit=1* in /boot/config.txt using *sudo nano /boot/config.txt*
    - Another way could be to delete the v7 images in the /boot directory
        - *cd /boot*
        - Check contents; only the kernel8.img should be present
        - If kernel7l.img and kernel7.img are present, delete those using rm.
- Ensure the plense user on the Raspberry Pi has administrator permissions.
    - Enter: *groups plense*
    - In the output, either sudo or wheel should appear
    - Enter: *sudo visudo*
        - Add the line: *plense ALL=(ALL:ALL) ALL*
        - Reboot
- Ensure the Raspberry Pi is connected to the same network as the development computer.
- Download and install AWS CLI for Linux ARM system
    - Enter the following in the command line:
        - *curl "https://awscli.amazonaws.com/awscli-exe-linux-aarch64.zip" -o "awscliv2.zip"*
        - *unzip awscliv2.zip*
        - *cd aws*
        - *sudo ./install*
    - Verify that AWS CLI has been installed by: *aws --version*
        - If it is installed correctly, it shows an output in the following format: *aws-cli/2.10.0 Python/3.11.2 Linux/4.14.133-113.105.amzn2.x86_64 botocore/2.4.5*
    - AWS CLI installation manual for more information: https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html
- Configure AWS CLI on core edge device, using same AWS region as development computer and goal cloud architecture. The IAM role for the Pi's has been configured in AWS already, so those access keys can be used.

    - In the command terminal type: *aws configure*
    - Copy and enter our Access key and Secret access key which can be found here: https://plensetechnologiesbv.sharepoint.com/:x:/s/ResearchDevelopment/ERtefil3nedHvB2iUgIPcnABOr5T5GiZdFbuiD87qLA3lA?e=HDUpsQ
    - Set region to eu-central-1 (OR THE APPROPIATE ONE WE ARE USING)
    - Leave "Default output format" blank by hitting Enter
- Install Java runtime
    - Enter in cmd: *sudo apt install default-jdk*
    - Verify installation, enter in cmd: *java --version*
        - Output should be in the format:
          openjdk version "11.0.9.1" 2020-11-04
          OpenJDK Runtime Environment (build 11.0.9.1+1-post-Debian-1deb10u2)
          OpenJDK 64-Bit Server VM (build 11.0.9.1+1-post-Debian-1deb10u2, mixed mode)
- Install Docker, following these steps: https://www.simplilearn.com/tutorials/docker-tutorial/raspberry-pi-docker
    - *sudo apt-get update*
    - *sudo apt-get upgrade*
    - *curl -fsSL https://get.docker.com -o get-docker.sh*
    - *sudo sh get-docker.sh*
    - *sudo usermod -aG docker plense*
    - Logout and log back in again or reboot
    - Verify installation: *docker version* or *docker info*
    - If permission errors arise, follow these steps: https://www.digitalocean.com/community/questions/how-to-fix-docker-got-permission-denied-while-trying-to-connect-to-the-docker-daemon-socket
    - Verify docker runs successfully by entering in the terminal: *docker run hello-world*

Install and run Greengrass on the Raspberry Pi using CLI

- Log in on AWS console, using the following credentials:
    - Account ID: 1060 0588 4410
    - IAM user name: masterpi
    - Password: see masterpi credentials file
- Install and configure the AWS IoT Greengrass Core software
    - Enter in cmd to switch to the home directory: *cd ~*
    - Enter in cmd to download the AWS IoT Greengrass Core software to a file named greengrass-nucleus-latest.zip: *curl -s https://d2s8p88vqu9w66.cloudfront.net/releases/greengrass-nucleus-latest.zip > greengrass-nucleus-latest.zip*
    - Enter in cmd to unzip the software to the GreengrassInstaller folder on the Raspberry Pi: *unzip greengrass-nucleus-latest.zip -d GreengrassInstaller && rm greengrass-nucleus-latest.zip*
    - Enter in cmd, where the yellow marked Greengrass thing-name should be replaced with plensepiXXXXX, which is our Raspberry Pi hostname:
      *sudo -E java -Droot="/greengrass/v2" -Dlog.store=FILE *
      *-jar ./GreengrassInstaller/lib/Greengrass.jar *
      *--aws-region eu-central-1 *
      *--thing-name plensepi00001 *

--thing-group-name MyGreengrassCoreGroup \
--thing-policy-name GreengrassV2IoTThingPolicy \
--tes-role-name GreengrassV2TokenExchangeRole \
--tes-role-alias-name GreengrassCoreTokenExchangeRoleAlias \
--component-default-user ggc_user:ggc_group \
--provision true \
--setup-system-service true \
--deploy-dev-tools true

- Following message should appear when the installer succeeded:
  Successfully configured Nucleus with provisioned resource details!
  Configured Nucleus to deploy aws.greengrass.Cli component
  Successfully set up Nucleus as a system service

- Run the Greengrass software
  - Enter in cmd: *sudo /greengrass/v2/alts/current/distro/bin/loader*
  - Software prints the following message if it launches successfully: *Launched nucleus successfully.*
- Verify the Greengrass CLI installation
  - Enter in cmd, where the yellow marked Greengrass thing-name should be replaced with plensepiXXXXX, which is our Raspberry hostname: *aws greengrassv2 list-effective-deployments --core-device-thing-name plensepi00001*
  - The coreDeviceExecutionStatus should have the status SUCCEEDED, meaning that Greengrass is up and running. Other components belonging to the configured GreengrassThingGroup should also appear.
  - When access denied errors appear while interacting with Greengrass from the Raspberry Pi cmd, enter the following in the cmd to set the permissions for the plense user to administrator
    - *sudo chown plense:plense /greengrass/v2/logs*
    - *sudo chown plense /greengrass/v2/cli_ipc_info/user-0*
  - The deployment of the Greengrass components on our Greengrass core devices can be checked from the console as well:
    - Navigate to the Greengrass service > Greengrass devices > core devices
    - Select the core device for which the components need to be checked
    - Check the components
  - Add the user that runs the docker container using the Greengrass daemon on the Raspberry Pi to the docker user group, so that at startup, the docker container can access the docker socket (the entry point for docker containers to run): *sudo usermod -aG docker ggc_user*
    - In our case, the user is ggc_user. This is defined in the Greengrass installation.
  - Also add ggc_user to the dialout group to ensure serial communication over the USB ports can take place
    - *sudo usermod -aG dialout ggc_user*
    - reboot
  - Advanced permission denied errors can arise when running a docker container as a Greengrass component, even when the user is added to the docker group and a hello-world docker container runs successfully. This permission denied error is printed in the corresponding docker container local log file on the core device. To ensure these advanced permission denied errors do not occur, do the following:
    - In the terminal enter:

        - *sudo chmod 666 /var/run/docker.sock*
        - *sudo chgrp $USER /lib/systemd/system/docker.socket*
        - *sudo chmod g+w /lib/systemd/system/docker.socket*
    - https://stackoverflow.com/questions/48957195/how-to-fix-docker-got-permission-denied-issue?page=1&tab=scoredesc#tab-top
- Minimal IAM policy to provision resources, as recommended by AWS, not executed for the test install

When the Greengrass software is up and running on the Raspberry Pi, and the Greengrass core device is configured correctly to the GreengrassThingGroup, the components of that group will deploy automatically to the Raspberry Pi. When the Raspberry Pi happens to reboot, the Greengrass software automatically starts again, and the components will be brought back online.

Troubleshooting

- Local greengrass log files include more elaborate logs to troubleshoot any errors. These can be found in a directory which is defined by the Greengrass nucleus' component configuration. In our case, the logs can be found in: /greengrass/v2/logs

After all these installation and configuration steps, the pi is ready for the custom Plense AWS Greengrass components. Well done!

Install TeamViewer

- Download and install TeamViewer arm64 Host version from https://www.teamviewer.com/nl/download/raspberry-pi/
- Open TeamViewer by clicking on the icon and add Thijs Bieling as the account
    - Credentials: ask Thijs
    - Thijs should add the device to the list of trusted devices by clicking the link in the email that has been sent
- Test the connection from the TeamView full client application on one of the development computers.
- Enjoy the remote connection 🛜 🛜 🛜

Setup the GSM connection

Set up the GSM connection using NetworkManager

- Identify the APN of the used provider
- In the case of KPN – our provider – that is portalmmm.nl
-

Setup the SD card connection

- Identify the mount location of the SD card for the specific edge computer.
    - For our first edge computer, it is /media/plense/9C33-6BBD
- Is this the same for every edge computer? If not, this will cause extra difficulty in bind mounting the SD card location in the Greengrass component recipes. But these are worries for a later Miranda (I cannot test that right now).
- The mount location can be changed.

Setup metadata folder

For the local caching of the metadata of the connected sensors. This metadata is stored on the eMMC of the compute module since it is crucial information!

- Make a new directory /home/plense/metadata
- Ensure that it exists: ls -l /home/plense/metadata
    - Should show the contents
- Ensure that the Docker daemon has permissions to access the directory
    - sudo chmod -R 755 /home/plense/metadata

Setup the edge computer GSM connection as a WiFi hotspot

Edge computer prerequisites

- Inspect the name of the wireless interface by
  - *Here the command*
  - Normally, the wireless interface is called wlan0, however, it can also be named differently, like
- Check that the wireless interface of the edge computer is able to be used as an access point.
  - *nmcli -f WIFI-PROPERTIES.AP device show wlan0*
  - Should return: WIFI-PROPERTIES.AP: yes
- The gsm connection is managed via the wwan 0 interface and is a distinct interface from the wlan0 interface, so the edge computer is able to create a wlan hotspot using the gsm connection.

Configure the wlan0 interface

So that it is recognized, ready to make new connections, and is in the default mode.

- Check that the wlan0 interface is recognized by the system
  - *iwconfig*
  - Should output the wlan0 interface and its properties
- Bring the wlan0 up so that new connections can be made
  - *sudo ip link set wlan0 up*
  - When an error arises that bringing wlan0 up is blocked by RFkill, do the following
    - *rfkill list*
    - Check if in the output, wlan0 is present. All blocking can be taken out of place by
      - *sudo rfkill unblock all*
    - Or, a specific device can be unblocked by
      - *sudo rfkill unblock [index]*
    - Try bringing up the interface again with *sudo ip link set wlan0 up*
- Check that the wlan0 interface is up
  - *ip a show wlan0*
  - Should output in the format: wlan0 <BROADCAST, MULTICAST, UP, LOWER_UP> mtu 1500 qdisc pfifo_fast state DOWN group default qlen 1000
    - In between < >, it should be UP
    - Outside < >, it can be down. This indicates that there is currently no active connection.
- Set the state to default
  - *sudo ip link set wlan0 mode default*

Configure NetworkManager

- Ensure NetworkManager manages the wlan0 interface
  - *nmcli dev status*
  - You should see wlan0 listed with "managed" status. If it's not, you might need to force NetworkManager to manage it:

        - *nmcli dev set wlan0 managed yes*
- Ensure that no other network services interfere with NetworkManager
    - Check the NetworkManager configuration
      cat /etc/NetworkManager/NetworkManager.conf
      There should be no entries that disable NetworkManager managing the wlan0 connection.
    - If errors arise later down the line, investigate more into the network managing services that might interfere with the correct set up of the hotspot. In our case, probably nothing interferes with the NetworkManager since we did not set any other services to manage wlan0 and NetworkManager is the default.
- Ensure NetworkManager does not automatically reassign MAC addresses
  NetworkManager does this for safety, but it can interfere a stable connection.
    - *sudo nano /etc/NetworkManager/NetworkManager.conf*
    - add the lines at the bottom:
        - *[device]*
        - *wifi.scan-rand-mac-address=no*
- Restart NetworkManager to ensure all configuration changes will take place
    - *sudo systemctl restart NetworkManager*

Create the hotspot using NetworkManager

- Enter the following:
- *nmcli con add type wifi ifname wlan0 con-name <my-hotspot> autoconnect yes ssid <MyHotspot> mode ap ipv4.method shared*
- *nmcli con modify <my-hotspot> 802-11-wireless-security.key-mgmt wpa-psk*
- *nmcli con modify <my-hotspot> 802-11-wireless-security.psk <"YourStrongPassword">*
- *nmcli con up <my-hotspot>*
    - Should return “Connection successfully activated …”

Test the hotspot

- Connect to the hotspot with another device, eg. your mobile phone.
- Check that the hotspot turns on automatically after
    - A cmd reboot
    - A physical reboot

Appendix A

Screenshots of the problem and solution dealing with the advanced permission denied errors when running a Greengrass component docker container.

https://stackoverflow.com/questions/48957195/how-to-fix-docker-got-permission-denied-issue?page=1&tab=scoredesc#tab-top

In the rare case the Stackoverflow page is deleted.

After an upgrade I got the permission denied. Doing the steps of 'mkb' post install steps don't have change anything because my user was already in the 'docker' group; I retry-it twice any way without success.

After an search hour this following solution finaly worked :

```
sudo chmod 666 /var/run/docker.sock
```

Solution came from Olshansk.

Look like the upgrade have recreate the socket without enough permission for the 'docker' group.

Problems

This hard *chmod* open security hole and after each reboot, this error start again and again and you have to re-execute the above command each time. I want a solution once and for all. For that you have two problems :

- 1) **Problem with `SystemD`** : The socket will be create only with owner 'root' and group 'root'.

  You can check this first problem with this command :

  ```
  ls -l /lib/systemd/system/docker.socket
  ```

  If every this is good, you should see '`root/docker`' not '`root/root`'.

- 2) **Problem with *graphical Login*** : https://superuser.com/questions/1348196/why-my-linux-account-only-belongs-to-one-group

  You can check this second problem with this command :

  ```
  groups
  ```

  If everything is correct you should see the *docker* group in the list. If not try the command

  ```
  sudo su $USER -c groups
  ```

  if you see then the *docker* group it is because of the bug.

Solutions

If you manage to to get a workaround for the graphical login, this should do the job :

```
sudo chgrp docker /lib/systemd/system/docker.socket
sudo chmod g+w /lib/systemd/system/docker.socket
```

But If you can't manage this bug, a not so bad solution could be this :

```
sudo chgrp $USER /lib/systemd/system/docker.socket
sudo chmod g+w /lib/systemd/system/docker.socket
```

This work because you are in a graphical environnement and probably the only user on your computer. In both case you need a reboot (or `sudo chmod 666 /var/run/docker.sock`)

Share Improve this answer Follow

edited Nov 23, 2023 at 1:18

Benjamin Loison
4,145 ● 4 ● 17 ● 35

answered Jul 16, 2018 at 13:05

Galigator
9,587 ● 2 ● 26 ● 40