Cradlepoint Unified Communications

This tutorial describes running an Asterisk UC core system in a Debian based container for ARM processors on Cradlepoint platforms. The container supports local SIP based clients, including ATA's, while peering with a SIP service provider for access to the PSTN and other IMS systems. Deploying UC on Cradlepoint platforms is just another example of the platform's extensibility leveraging its edge compute functionality.

Considering the FCC’s order releasing carriers from their obligation to support POTS lines on Aug 2nd, 2022, the use of UC integrated on the Cradlepoint platforms can provide immediate value. Paired with a SIP ATA (Analog Telephone Adapter/Analog Terminal Adapter), which is a device that translates analog-to-SIP (Ethernet/IP based) and vice versa, the Cradlepoint can host a local SIP-based UC server for the SIP ATA to register. There are many systems that are still leveraging analog voice, and with copper POTS lines being sunset provides a method to connect those systems until transitioned to digital/IP communication methods.

More generically, immediate value is achieved by the reduction in managed device count at a branch location. Cost for a separate piece of UC hardware, OS, licensing, support and resources (energy, cooling, floor/rack space) is reduced if not entirely removed. In the event of a WAN outage, the Cradlepoint is already capable of providing business continuity through alternate WAN connectivity. In the event no WAN options are available during an outage, the UC core running on the Cradlepoint still provides intra-site communication capabilities (i.e. handset-to-handset, video-to-video).

**Item's needed:**

* E300, E3000, IBR1700 or R1900 (i.e. any Cradlepoint platform supporting containers)
* Asterisk for ARM Container Image
* Alpine for ARM Container Image – Used for storing Asterisk system customization, Vmail, etc.
* Zoiper PC & smart phone client
* Grandstream ATA – Optional to showcase POTS replacement
* Analog phone – Optional to showcase POTS replacement
* SIP Service Provider – Showcase outbound PSTN calling (ie. Vonage, Anveo (example included) or other SIP provider)

**High Level Steps**

Create a new local area Network on the Cradlepoint specific for voice use. Assign the appropriate firewall policies with a minimum of 'allow all' from the voice LAN to the WAN zone. Modify the Alpine based image to reflect site specific Asterisk requirements (dial plan, SIP peering, voicemail, IVR, etc.). The Alpine and Asterisk containers will share the same data volume where the Asterisk UC core will read modified configuration files residing on Alpine. This removes the need to modify anything on the Asterisk image itself. Finally, create the container project definition.

In a production environment, you'd ideally separate the voice core into its own separate network from other applications and clients including voice clients. Once isolated, only the TCP/UDP ports that need to be exposed would be done using a firewall rule. For simplicity and demo purposes, this has not been done. Instead, the example uses a single voice network that includes both the voice core and clients. You'll still be able to showcase the isolation from other network segments via firewall policy. If wanting to further isolate the voice core, port information is referenced in Appendix A.

**Detailed Steps**

Docker Image Preparation

Prepare the Docker Images to host from your own repository. You will retrieve 2 x images from Docker Hub:

* clintberger/armasterisk:latest # Asterisk UC Core running on Debian 9
* clintberger/armasterisk-storage:latest # Alpine image used as modifiable storage for Asterisk

The images are based on an ARMv7 architecture. Ideally, you would use a Raspberry Pi or other ARM based product with Docker support to manage/modify the images. Buildx on other platforms is also an option, but out of scope for this document. Regardless, the only image that requires modification will be the Alpine image.

The Alpine image contains files in the '/etc/asterisk' directory used to configure the Asterisk system (SIP clients, SIP trunks, dial plans, IVR, VMail, etc.). As of this writing, it's possible to retrieve the image on a Windows 10 x86 based system, make file modifications (nano editor loaded), commit, and push back to your repository in Docker Hub without issue. The Alpine image also contains an FTP Server daemon (VSFTP) for file retrieval. VSFTP will need to be configured for your environment, including adding users.

At a minimum, the two files you'll want to configure in the '/etc/asterisk' directory are:

* sip.conf # Contains the SIP client and trunk definitions
* extensions.conf # Contains call routing instructions and the dial plan

**sip.conf**

The following are the contents of the sip.conf file. The first section (**starcom**) is a publicly available SIP trunk used for testing. It allows connectivity without an account and the ability to dial 1-800 numbers to test the trunk. The next section (**general**) will be used to define your SIP registration for inbound calling. The example below shows Anveo as the SIP provider since I already use the platform (Vonage will obviously work as well, just haven't documented yet). Simply replace username:password with your actual account information. Do the same for the **Anveo** section below for outbound calling. Lastly, manage the SIP client extensions. 3 x examples have been created with their secret the same as their user name (user1, user2 and ata1).

[starcom]

type=peer

host=173.193.144.207

context=starcom-in

insecure=port

disallow=all

allow=ulaw

qualify=yes

dtmfmode=rfc2833

sendrpid=yes

[general]

register => username:password@sip.anveo.com:5010

nat=force\_rport,comedia

directmedia=no

context=public

allowoverlap=no

udpbindaddr=0.0.0.0

tcpenable=yes

tcpbindaddr=0.0.0.0

transport=udp

srvlookup=yes

disallow=all

allow=ulaw

[anveo]

type=peer

host=sip.anveo.com

port=5010

username=username

secret=password

insecure=port,invite

disallow=all

allow=ulaw

allow=alaw

allow=g729

dtmfmode=rfc2833

context=anveo-in

[user1]

type=friend

username=user1

secret=user1

host=dynamic

context=default

[user2]

type=friend

username=user2

secret=user2

host=dynamic

context=default

[ata1]

type=friend

username=ata1

secret=ata1

host=dynamic

context=default

**extensions.conf**

The extensions.conf will establish what to do with inbound and outbound calling, as well as defining the extensions and other dial plan related information. For the most part, the config can be used as-is for demo purposes. You'll simply want to modify the extension information highlighted below to match your phone number supplied by your SIP trunk provider.

Extensions are as follows: user1 is ext 101, user2 is ext 102 and ata1 is ext 103. The thought being user1 can be assigned to a soft client on a PC/Mac, user2 can be assigned to a soft client on a smart phone and 'ata1' can be assigned to a SIP ATA for demonstrating POTS line replacement. When dialing into the Asterisk system, the current call flow defined will auto-answer after one ring with a message asking, "who would you like to call?". Pressing 1 will dial user1 at ext 101, 2 will dial user2 and 3 will dial the ata1. Much of the rest of the configuration is ignored in its current state.

; extensions.conf - the Asterisk dial plan

;

; Static extension configuration file, used by

; the pbx\_config module. This is where you configure all your

; inbound and outbound calls in Asterisk.

;

; This configuration file is reloaded

; - With the "dialplan reload" command in the CLI

; - With the "reload" command (that reloads everything) in the CLI

;

; The "General" category is for certain variables.

;

[general]

static=yes

writeprotect=no

clearglobalvars=no

autofallthrough=yes

; The default trunk that is setup is Starcom. This is required

; for sip.conf to connect this free Toll-free trunk.

[starcom-in]

; Standard extensions begin here

[default]

exten => \_18XXXXXXXXX,1,Dial(SIP/${EXTEN}@starcom)

exten => \_X.,1,Set(CALLERID(number)=1112223333)

exten => \_X.,2,Dial(SIP/anveo/${EXTEN})

exten => user1,1,Dial(SIP/user1)

exten => user1,2,HangUp()

exten => 101,1,Goto(user1,1)

exten => user2,1,Dial(SIP/user2)

exten => user2,2,HangUp()

exten => 102,1,Goto(user2,1)

exten => ata1,1,Dial(SIP/ata1)

exten => ata1,2,HangUp()

exten => 103,1,Goto(ata1,1)

[anveo-in]

exten => \_X.,1,NoOp(From Anveo ${EXTEN})

exten => \_X.,2,Goto(mainmenu,s,1)

exten => s,1,Verbose(Incoming SMS from ${CALLERID(num)})

exten => s,n,Answer

exten => s,n,Set(SMSINRAW=${MESSAGE(body)})

exten => s,n,Set(SMSIN=${URIENCODE(${SMSINRAW})})

exten => s,n,NoOp(Saving SMS message to a file)

exten => s,n,Set(FILE(/var/spool/asterisk/sms/${STRFTIME(${EPOCH},,%F-%T)}-${CA$

exten => s,n,Hangup

[smsdial]

exten => 1,1,NoOp(Sending a Message)

exten => 1,n,Set(MESSAGE(body)=Hello)

exten => 1,n,MessageSend(sip:<7209758427@sip.anveo.com:5010>,<0431463546>)

[mainmenu]

exten => s,1,Answer

exten => s,n,Background(silence/1&who-would-you-like-to-call) ;

exten => s,n,WaitExten

exten => 1,1,Goto(default,user1,1)

exten => 2,1,Goto(default,user2,2)

exten => 3,1,Goto(default,ata1,3)

exten => i,1,Hangup

[submenu]

exten => s,1,Ringing ; Make them comfortable with 2 seconds of ringback

exten => s,n,Wait(2)

exten => s,n,Background(submenuopts) ; "Thanks for calling the sales ;department. Press 1 for steve, 2 for..."

exten => s,n,WaitExten

[globals]

[local]

;

; Master context for local, toll-free, and iaxtel calls only

;

include => default

[macro-trunkdial]

;

; Standard trunk dial macro (hangs up on a dialstatus that should

; terminate call)

; ${ARG1} - What to dial

;

exten => s,1,Dial(${ARG1})

exten => s,n,Goto(s-${DIALSTATUS},1)

exten => s-NOANSWER,1,Hangup()

exten => s-BUSY,1,Hangup()

exten => \_s-.,1,NoOp

[public]

; Simple application to repeat the time.

[time]

exten => \_X.,30000(time),NoOp(Time: ${EXTEN} ${timezone})

exten => \_X.,n,Wait(0.25)

exten => \_X.,n,Answer()

; the amount of delay is set for English; you may need to adjust this time

; for other languages if there's no pause before the synchronizing beep.

exten => \_X.,n,Set(FUTURETIME=$[${EPOCH} + 12])

exten => \_X.,n,SayUnixTime(${FUTURETIME},Zulu,HNS)

exten => \_X.,n,SayPhonetic(z)

; use the timezone associated with the extension (sip only), or system-wide

; default if one hasn't been set.

exten => \_X.,n,SayUnixTime(${FUTURETIME},${timezone},HNS)

exten => \_X.,n,Playback(spy-local)

exten => \_X.,n,WaitUntil(${FUTURETIME})

exten => \_X.,n,Playback(beep)

exten => \_X.,n,Return()

***Note*** *– Any time you make a change to one of the config files, you'll need to restart the Asterisk core to accept those changes. This must be done within the Asterisk container. Once inside the container, access the Asterisk Console by typing 'asterisk -r'. At the Console type 'core restart now' to reload Asterisk. You'll immediately be brought back to the Debian Linux command line where you can exit the container. The following shortcut command line will accomplish the same: asterisk -rx “core restart now”*

Once you've modified the config files with your site-specific personal information, commit and push the images to your private Docker Hub repository (e.g., <<yourdockerhub>>/private:armasterisk) or whatever secure location makes sense.

Cradlepoint Config

Now that the images are ready, next you’ll prepare the IP network segment that will be used by both the Asterisk UC core as well as clients (e.g., the SIP ATA). Create a new LAN profile called **voice**. For the example included, I've used the following:

* Cradlepoint LAN IP Address: 192.168.205.1/24
* IP UC Core Range: 192.168.205.2-10
  + armasterisk: 192.168.205.10 # Configured in container project
  + armasterisk-storage: 192.168.205.5 # Configured in container project
* IP DHCP Pool: 192.168.205.11-100
* VLAN 205

For simplicity, choose to add the new voice network to the existing Primary LAN firewall rules. You can come back later and segment the voice network into its own firewall zone. At this point however, we want to rule out variables until verifying the system is running successfully. Lastly, move the desired LAN ports into the new voice network. At a minimum this should include the Ethernet port the ATA is attached.

Once the network configuration is in place, you can build the Asterisk UC container project. Simply create a new project and copy/paste the following as the compose configuration. Just replace 'yourdockerhubsite' with your Docker Hub location. The following assumes your Asterisk image is located at 'yourdockerhubsite/private:armasterisk' and the Alpine image is located at 'yourdockerhubsite/private:armasterisk-storage'. Also, ensure the network maps up to the voice network you just created as the UUID will be different than what’s included below:

version: '2.4'

services:

Alpine:

image: 'yourdockerhubsite/private:armasterisk-storage'

entrypoint: sh -c "/usr/sbin/vsftpd && sleep infinity"

working\_dir: /etc/asterisk

logging:

driver: json-file

restart: unless-stopped

ports:

- '8020:20'

- '8021:21'

- '8022:22'

volumes:

- 'shared-data:/etc/asterisk'

networks:

lannet:

ipv4\_address: 192.168.205.5

Asterisk:

image: 'yourdockerhubsite/private:armasterisk'

depends\_on:

- Alpine

working\_dir: /etc/asterisk

logging:

driver: json-file

restart: unless-stopped

ports:

- '8088:8088'

- '5060:5060/udp'

- '4569:4569/udp'

volumes:

- 'shared-data:/etc/asterisk'

networks:

lannet:

ipv4\_address: 192.168.205.10

volumes:

shared-data:

driver: local

networks:

lannet:

driver: bridge

driver\_opts:

com.cradlepoint.network.bridge.uuid: 00000002-1bcb-3605-9dce-9081814b1c9a

ipam:

driver: default

config:

- subnet: 192.168.205.0/24

gateway: 192.168.205.1

**Appendix A:**

Firewall port information:

By default, Asterisk uses ports UDP/TCP port 5060 for SIP signaling and UDP 10,000 through 20,000 for RTP, although that can be tuned with the rtp.conf file. Where using IAX2, UDP 5036 is used. Additional port information can be found in the Asterisk documentation.

The Asterisk UC core will initiate connectivity with the SIP provider, at which point bi-directional communication can occur. WAN facing TCP/UDP ports do not need to be exposed, nor is a public IP address on the WAN interface required.

Asterisk documentation:

<https://wiki.asterisk.org/wiki/display/AST/Home>

Asterisk Admin Guide:

<https://wiki.asterisk.org/wiki/download/attachments/19005471/Asterisk-Admin-Guide.pdf?version=2&modificationDate=1582306776654&api=v2>

Zoiper Soft Client (Windows, Mac, Linux, Android & iOS):

<https://www.zoiper.com/en/voip-softphone/download/current>

Grandstream ATA's:

<http://www.grandstream.com/products/gateways-and-atas/analog-telephone-adaptors>

Note – Tested with a HandyTone-286 which is rather old. The current version is the HT801 ($32.00 on Amazon)

Inexpensive analog phone:

<https://www.amazon.com/dp/B00R1VL7PI/ref=twister_B08F669F86?_encoding=UTF8&psc=1>

Optional: PoE Splitter for Grandstream ATA (leverage PoE port of E3000 – 48v PoE to 5v (micro USB)):

<https://www.amazon.com/UCTRONICS-802-3af-Splitter-Ethernet-Raspberry/dp/B01MDLUSE7>

Inexpensive SIP phone (eBay - Vtech VSP861 Eris Terminal SIP Color TouchScreen ):

<https://www.ebay.com>

**Appendix B:**

Docker commands supporting the customization discussed:

*Download and run the images as containers locally in detached mode:*

docker run -d –-name asterisk clintberger/armasterisk

docker run -d –-name asterisk-storage clintberger/armasterisk-storage

*Access the interactive shell of the container:*

docker exec -ti asterisk-storage sh

* cd /etc/asterisk
* nano sip.conf
* nano extensions.conf

*Save the containers with their corresponding image tag name:*

docker commit asterisk yourdockerhubsite/private:armasterisk

docker commit asterisk-storage yourdockerhubsite/private:armasterisk-storage

*Push the new images to your private Docker Hub repository:*

docker push yourdockerhubsite/private:armasterisk

docker push yourdockerhubsite/private:armasterisk-storage

Asterisk commands:

*Access the Asterisk Command Console:*

asterisk -r

*Show help from Command Console:*

?

*Show SIP Peers from Command Console (clients and trunks):*

sip show peers

*Restart the Asterisk Core from Command Console (restarts with any changes made to conf files):*

core restart now

*Restart the Asterisk Core from Linux prompt (execute any Command Console commands via Linux):*

asterisk -x "core restart now"

**Appendix C:**

Visual Representation of Elements

