CMF Test

1. Test Planning

CMFLAB Dashboard

CMFLAB Backlog

Template Tickets (these are cloned for each task)

Task	Template Ticket
TDK Intake	**CMFLAB-1125** - Getting issue details... STATUS
RDK-B Iteration/Release	**CMFLAB-552** - Getting issue details... STATUS
RDK-V Iteration/Release	**CMFLAB-365** - Getting issue details... STATUS
RDK-B CODEMGMT Iteration	**CODEMGMT-238** - Getting issue details... STATUS
RDK-B CODEMGMT Quarterly Release	**CODEMGMT-425** - Getting issue details... STATUS
RDK-V CODEMGMT Iteration	**CODEMGMT-180** - Getting issue details... STATUS
RDK-V CODEMGMT Quarterly Release	**CODEMGMT-554** - Getting issue details... STATUS
RDK-C CODEMGMT Quarterly Release	**CODEMGMT-1063** - Getting issue details... STATUS

2. Sanity Test Frameworks

CMF Video Sanity Tests

CMF Broadband Sanity Tests

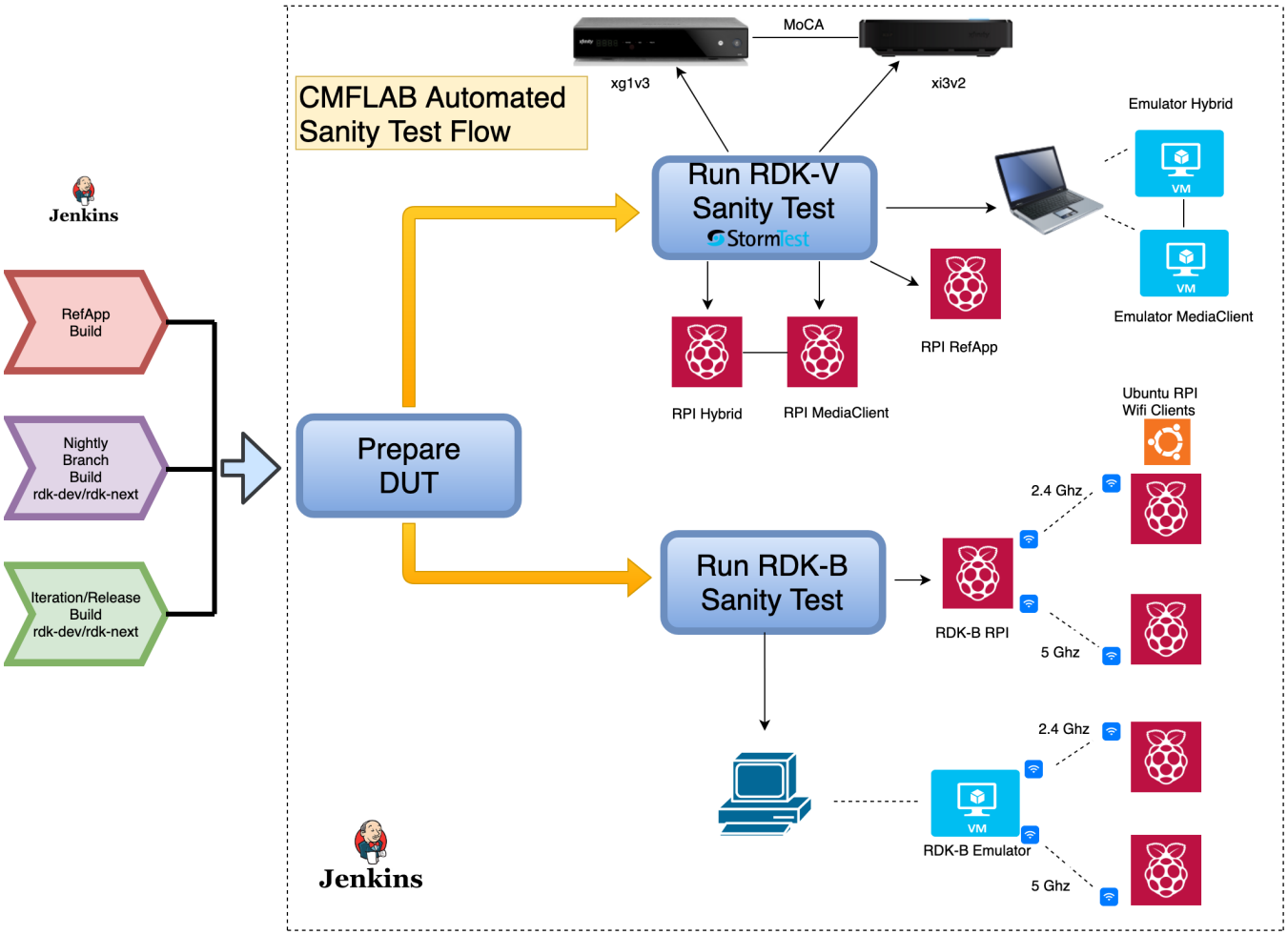
CMF Sanity Test Monitoring

Sanity Test Build Trend

Sanity Test Design

RefApp Documentation

Emulator and RPI Reference Platform Documentation



Sanity Test Flows

DUT	Jenkins Sanity Test Flow Job	Comments

EMU-B	https://jenkins.cmf.code.rdkcentral.com/view/TEST/job/test-rdkb-emulator-wifi-flow/	• there are 3 Sanity EMU-B setups on
		  ◦ Raven
		  ◦ R02Server
		  ◦ Wind

RPI-B	https://jenkins.cmf.code.rdkcentral.com/view/TEST/job/test-rdkb-raspberrypi-wifi-flow/	
EMU-V	https://jenkins.cmf.code.rdkcentral.com/view/TEST/job/test-hv04-emulators/	• EMU's are setup on laptop connected to StormTest HV04
RPI-V	https://jenkins.cmf.code.rdkcentral.com/view/TEST/job/test-rdkv-raspberrypi-flow/	• RPI-V RPI's are setup on R04 which is StormTest enabled rack

RPI RefApp	https://jenkins.cmf.code.rdkcentral.com/view/TEST/job/test-raspberrypi-refapp-flow/	• RPI-V Refapp RPI's are setup on R04 which is StormTest enabled rack • Uses StormTest OCR to check subtitles
RPI Thunder	https://jenkins.cmf.code.rdkcentral.com/view/TEST/job/test-raspberrypi-thunder-flow/	• RPI Thunder are setup on R04 stormtest rack
RPI Westeros	https://jenkins.cmf.code.rdkcentral.com/view/TEST/job/test-oss-raspberrypi-flow/	• Deprecated ?
~~xg1v3~~ ~~xi3v2~~	~~https://jenkins.cmf.code.rdkcentral.com/view/TEST/job/test-hv04-platforms/~~	• ~~Deprecated~~ • ~~StormTest HV04~~

3. TDK Test Frameworks

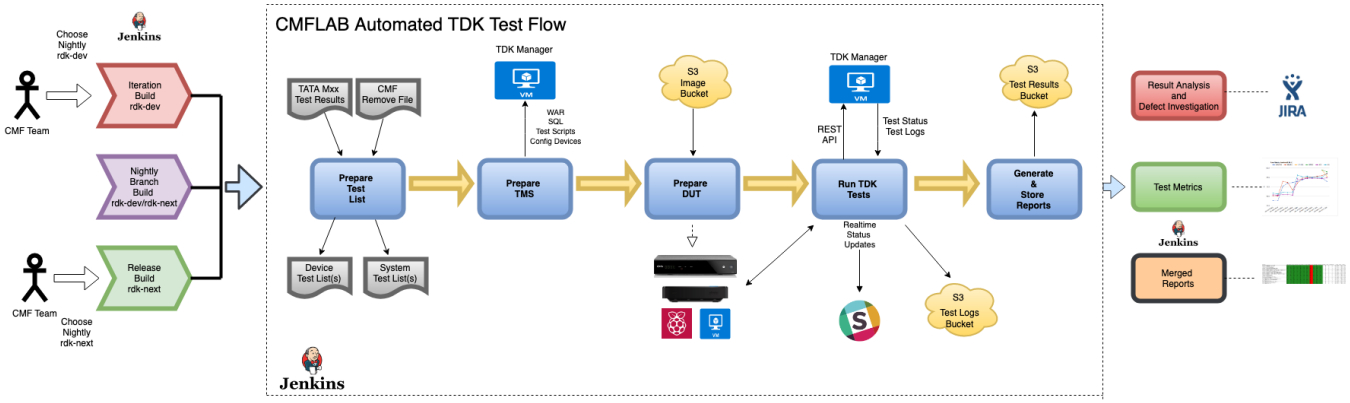
TDK Documentation and Releases

- Contains all the TDK Documentation you need from
  - Setting up the TDK manager (note we have automated this in CMF via vagrant/puppet)
  - TDK additional framework setup (e.g. aamp, spark, xconf, RDKB-E2E frameworks etc etc)
  - TDK Video and Broadband Release Documentation
  - Lots of other useful TDK information

Emulator and RPI Reference Platform Documentation

- Contains setup and build instructions for emulator and RPI reference platforms

CMF Test Framework Design



CMF TDK Testing is done using 3 frameworks (2 in active use):

- EACH PHASE OF THE FRAMEWORK CAN BE RUN INDEPENDENTLY

DUT	TDK Framework	Jenkins Flow	TDK Manager URL	LabSlave	Comments
EMU-B	ATF	https://jenkins.cmf.code.rdkcentral.com/view/TEST/job/test-atf-start-flow-emub/	http://192.168.32.63/rdk-test-tool/?targetUri=%2Fexecution%2Fcreate	griffen	• https://code.rdkcentral.com/r/plugins/gitiles/cmf/test/+/master/cmf_testbed/atf/docs/
RPI-B	ATF	https://jenkins.cmf.code.rdkcentral.com/view/TEST/job/test-atf-start-flow-rpib/	http://192.168.32.69/rdk-test-tool/?targetUri=%2Fexecution%2Fcreate	griffen	
RPI-V	ATF	https://jenkins.cmf.code.rdkcentral.com/view/TEST/job/test-atf-start-flow-rpiv/	http://192.168.32.70/rdk-test-tool/?targetUri=%2Fexecution%2Fcreate	griffen	

EMU-V	RDKV-EMU	https://jenkins.cmf.code.rdkcentral.com/view/TEST/job/test-emu-start-flow/	see section "Access EMU-V TDK manager on web browser" in Useful Information	heron	• uses ATF for test list creation and report generation (these jobs run on griffen) • https://code.rdkcentral.com/r/plugins/gitiles/cmf/test/+/master/cmf_testbed/emulator/docs/
~~xg1v3~~ ~~xi3v2~~	~~RDKV-Platforms~~	~~https://jenkins.cmf.code.rdkcentral.com/view/TEST/job/test-platforms-start-flow/~~		corvus	• ~~uses ATF for test list creation and report generation~~ • ~~**Deprecated since March 2019**~~

All of which are kicked off with a set of input parameters to their various Jenkins jobs. Currently supporting 4 'RUNTYPE' options – 'release', 'nightly', 'nightly-full', and 'master' (no longer used).

There is also a dry-run option which we often to use the debug issues or for testing changes to the frameworks.

3.1. CMF TDK Test Result Tools

CMF TDK Merged Reports Tool

- This produces a comparison report per platform on TDK test runs on nightly and release branches, this it the main tool we use to track TDK test results and identify regressions
- Report can also be run for rdk-dev or rdk-next only based nightly branches
- note to be added to the daily email this tool automatically generates add your email to:
    - https://code.rdkcentral.com/r/plugins/gitiles/cmf/jenkins-dsl/jobs/test/+/master/test_summary_report_generator.groovy

TDK Merged Reports Tool

- This produces a comparison report per platform on a configurable number of TDK Reference Releases
- Useful on TDK Intake to see any change in test status or tests removed/added etc

3.2. TDK Test Metrics

Test Metrics	Link
Nightly	CMF Nightly Test Metrics
Iteration	CMF Iteration Test Metrics
Quarterly	CMF Release Test Metrics

4. CMFLAB

CMFLAB Confluence

5. Test and Git

CMF Gerrit

Comcast Gerrit

CMF Code Contribution Process

- Very useful doc to explain how to setup git and push changes for review to CMF Gerrit

CMF Test Git Repositories and Useful Info

```
# clone the test repo
git clone "https://code.rdkcentral.com/r/cmf/test"

# browse code in browser via gitiles (must be logged into gerrit)
https://code.rdkcentral.com/r/plugins/gitiles/cmf/test

# TDK test results repo (soon to be deprecated, used to store TDK reference results)
git clone "https://code.rdkcentral.com/r/tata/test-results"

# clone the jenkins dsl test repo where all our test jenkins jobs are maintained
git clone "https://code.rdkcentral.com/r/cmf/jenkins-dsl/jobs/test"

# setup the commit hook if you want push changes for review in CMF Gerrit
cd <repo>
gitdir=$(git rev-parse --git-dir); curl -o ${gitdir}/hooks/commit-msg https://code.rdkcentral.com/r/tools/hooks
/commit-msg ; chmod +x ${gitdir}/hooks/commit-msg

# useful git commands/workflow sequence if/when working on branches rather than pushing changes directly on
master
git checkout master
git pull –rebase
git checkout -b <branch name>

… make changes
git add <files>
git commit -m "CMFLAB-XXX <commit msg>"
git push origin <branch name>
  if single commit:
    git checkout master
    git cherry-pick <commit ID>
    git push origin HEAD:refs/for/master
  if multiple commits:
    git rebase -i HEAD~<# of commits from HEAD>
    … pick top commit; squash all others
    git log (**check to see that squashed commit has Change-ID)
    if no Change-ID in commit: git commit –amend; exit editor (Change-ID should now be there)
    git push origin HEAD:refs/for/master
```

6. Test AWS S3 Buckets

S3 Bucket	Purpose
s3://rdkcmf-test-results/	• stores nightly and release sanity and TDK test results • store TDK reference results
s3://rdkcmf-test-logs/	• stores nightly and release logs from TDK test runs • *may be used in future for sanity testing*
s3://rdkcmf-artifacts/jobs/	• where all builds from internal jenkins are stored • we mainly use this to retrieve images from tdk and community build jobs
s3://rdkcmf-artifacts/jobs/test-manual-artifacts/	• where we store our development builds, refer to next section below
s3://rdkcmf-community-artifacts/jobs/	• where all builds are stored from contribution builds, useful if you want to run some testing on a contribution

7. Test Development Builds

The test team maintain their own build scripts which we use to build and test debug images for defect investigations. These scripts are run on personal build slaves and can be used to build on any branch we support in CMF.

The scripts will upload the images to an s3 bucket and can also be used to automatically smoke test the image.

The scripts are maintained in the cmf test git repository https://code.rdkcentral.com/r/plugins/gitiles/cmf/test/+/master/cmf_testbed/tools/builds/

1) launch build slave via https://jenkins.cmf.code.rdkcentral.com/job/jenkins-reserve-slave

2) ssh to build slave (see jenkins console o/p for IP address)

3) set up build scripts on your build slave (if not already setup)

```
cd ~/jenkinsroot/workspace
git clone "https://code.rdkcentral.com/r/cmf/test"
ln -s test/cmf_testbed/tools/builds/  builds
```

4) build your image, e.g.

```
cd ~/jenkinsroot/workspace/builds
./build_<platform>.sh <branch> <prepare|build>

# Note follow the prompts to select what image(s) you want to build and whether you want to automatically smoke test the image
# typical branches used are `rdk-next`, `rdk-dev-yymm`, `nightly/yyymmdd-rdk-dev`, `nightly/yyymmdd-rdk-next`
```

- Notes
  - if you want to apply patches or do some reverts or cherrypicks do the build in two stages
    - run script with prepare first (this will do repo init and repo sync)
    - make code changes
    - then run script with build
  - if you just want to build a branch then just run script with no prepare or build options

8. Useful Information

8.1. Debugging RDKB

```
ifconfig
iwconfig
systemctl | grep -i ccsp
systemctl -l status <service>
journalctl -u <service>
ps -axf | grep hostapd <- check hotspot
```

- If groups of tests failing, likely issue with one service
- Some services have dependancies so not unusual for a few of them to be in bad state
- Can take a few minutes for all services to start

8.2. Debugging RDKV

```
systemctl
systemctl -l status <service>
journalctl -u <service>
ps -axf | grep rmfstreamer
```

- If E2E/RMF tests failing check if rmfStreamer is running

8.3. TDK EMU-B VM

on griffen:

```
get_vpc_key -i 192.168.32.74 -u <username> griffen
sudo su jenkins

# general info commands
vboxmanage list vms
vboxmanage list runningvms
vboxmanage showvminfo "rdkb-emu-tdk"
vboxmanage list usbhost

# restart vm (needed during runs where VM has crashed, i.e. ping not working)
vboxmanage startvm "rdkb-emu-tdk" --type headless

# if above start doesn't work power it off first
vboxmanage controlvm rdkb-emu-tdk poweroff
vboxmanage startvm "rdkb-emu-tdk" --type headless

# destroy VM - mechanism 1
vboxmanage controlvm rdkb-emu-tdk poweroff
vboxmanage unregistervm "rdkb-emu-tdk" --delete

# destroy VM - mechanism 2
vboxmanage startvm rdkb-emu-tdk --type emergencystop
vboxmanage unregistervm --delete rdkb-emu-tdk

# rdp session for RDK-B Emulator, run following ssh command then on RDP client connect to localhost:3388
ssh -o "PasswordAuthentication=no" -L 3388:192.168.32.74:3388 -i ~/.griffen-<username>.rsa <username>@192.
168.32.74
```

8.4. Access EMU-V TDK Managers via web browser

```
# do get vps on heron first
get_vpc_key -i 192.168.32.76 -u <username> heron

# ssh and tunnel through local port for the TDK Manager your interested in
ssh -L 8085:192.168.21.31:8080 -i ~/.heron-jamescol.rsa jamescol@192.168.32.76
ssh -L 8086:192.168.21.32:8080 -i ~/.heron-jamescol.rsa jamescol@192.168.32.76
ssh -L 8087:192.168.21.33:8080 -i ~/.heron-jamescol.rsa jamescol@192.168.32.76
ssh -L 8088:192.168.21.34:8080 -i ~/.heron-jamescol.rsa jamescol@192.168.32.76
ssh -L 8089:192.168.21.35:8080 -i ~/.heron-jamescol.rsa jamescol@192.168.32.76
ssh -L 8090:192.168.21.36:8080 -i ~/.heron-jamescol.rsa jamescol@192.168.32.76
ssh -L 8091:192.168.21.38:8080 -i ~/.heron-jamescol.rsa jamescol@192.168.32.76
ssh -L 8092:192.168.21.39:8080 -i ~/.heron-jamescol.rsa jamescol@192.168.32.76

# now you can access any of the 8 TDK EMU-V TDK Managers in your browser via
http://localhost:<port>/rdk-test-tool/execution/create

# e.g.
ssh -L 8085:192.168.21.31:8080 -i ~/.heron-jamescol.rsa jamescol@192.168.32.76
http://localhost:8086/rdk-test-tool/execution/create
```

8.5. Accessing Platforms/Device Under Test

Sometimes it's useful to access the platform/DUT directly for defect investigation etc for emulators we can ssh to the device, for RPI's we can connect to the devices either by serial (via telnet on iolan) or by ssh'ing directly to the device.

For RPI Rack Setup and IP addresses, see

- RPI R03 Setup
- RPI R04 Setup

```
# rdkb-emu TDK VM
get_vpc_key -i 192.168.32.74 -u <username> griffen
sudo su jenkins
```

```
ssh -o "StrictHostKeyChecking=no" root@10.5.25.100

# RPI via telnet
# telnet <iolan ip> <port>
# note the port number will follow the slot the RPI is in, e.g. 10001 is R03S01, 10008 is R03S08
telnet 10.5.25.5 10002

# use CTRL-] to quit
# tip if you accidently hit CTRL-C while in a telnet session it will stop responding, quit and do following to
recover
# echo 03 | xxd -r -p | nc <iolan ip> <port>
echo 03 | xxd -r -p | nc 10.5.25.5 10015

# RPI via ssh
# ssh -o "StrictHostKeyChecking=no" root@<IP>

# e.g. 2 RPI-B TDK
ssh -o "StrictHostKeyChecking=no" root@10.5.25.102
ssh -o "StrictHostKeyChecking=no" root@10.5.25.108

# RDKV-EMU TDK Hybrids and Clients
# these device IP's are not static so you either get it's IP from the console output of the
# https://jenkins.cmf.code.rdkcentral.com/view/TEST/job/test-emu-prepare-dut/ job are use vagrant to connect on
heron
get_vpc_key -i 192.168.32.76 -u <username> heron
sudo su jenkins

vagrant global-status
id       name           provider   state   directory
---------------------------------------------------------------------------------------------------------------
---------------------------------
7a9a0d5  default        virtualbox running /home/jenkins/jenkins2root/workspace/test-emu-prepare-dut/cmf_testbed
/emulator/functional/em-tdk-hyb1
c8902cd  default        virtualbox running /home/jenkins/jenkins2root/workspace/test-emu-prepare-dut/cmf_testbed
/emulator/functional/em-tdk-mc1
11caf80  default        virtualbox running /home/jenkins/jenkins2root/workspace/test-emu-prepare-dut/cmf_testbed
/emulator/functional/em-tdk-hyb2
a0e30e0  default        virtualbox running /home/jenkins/jenkins2root/workspace/test-emu-prepare-dut/cmf_testbed
/emulator/functional/em-tdk-mc2
bee0188  default        virtualbox running /home/jenkins/jenkins2root/workspace/test-emu-prepare-dut/cmf_testbed
/emulator/functional/em-tdk-hyb3
e1fc8c3  default        virtualbox running /home/jenkins/jenkins2root/workspace/test-emu-prepare-dut/cmf_testbed
/emulator/functional/em-tdk-mc3
318efaa  default        virtualbox running /home/jenkins/jenkins2root/workspace/test-emu-prepare-dut/cmf_testbed
/emulator/functional/em-tdk-hyb4
c0ef362  default        virtualbox running /home/jenkins/jenkins2root/workspace/test-emu-prepare-dut/cmf_testbed
/emulator/functional/em-tdk-mc4
1e66545  default        virtualbox running /home/jenkins/jenkins2root/workspace/test-emu-prepare-dut/cmf_testbed
/emulator/functional/em-tdk-hyb5
2f90e49  default        virtualbox running /home/jenkins/jenkins2root/workspace/test-emu-prepare-dut/cmf_testbed
/emulator/functional/em-tdk-mc5
d51c759  default        virtualbox running /home/jenkins/jenkins2root/workspace/test-emu-prepare-dut/cmf_testbed
/emulator/functional/em-tdk-hyb6
3bb2290  default        virtualbox running /home/jenkins/jenkins2root/workspace/test-emu-prepare-dut/cmf_testbed
/emulator/functional/em-tdk-mc6
186b64d  default        virtualbox running /home/jenkins/jenkins2root/workspace/test-emu-prepare-dut/cmf_testbed
/emulator/functional/em-tdk-hyb7
1ad3bc7  default        virtualbox running /home/jenkins/jenkins2root/workspace/test-emu-prepare-dut/cmf_testbed
/emulator/functional/em-tdk-mc7
972eb8b  default        virtualbox running /home/jenkins/jenkins2root/workspace/test-emu-prepare-dut/cmf_testbed
/emulator/functional/em-tdk-hyb8
b96a896  default        virtualbox running /home/jenkins/jenkins2root/workspace/test-emu-prepare-dut/cmf_testbed
/emulator/functional/em-tdk-mc8

# vagrant ssh <id>
vagrant ssh 7a9a0d5

# or also cd into relevant dir and ssh directly
cd /home/jenkins/jenkins2root/workspace/test-emu-prepare-dut/cmf_testbed/emulator/functional/em-tdk-hyb1
vagrant status
vagrant ssh
```

8.6. TDK Manger Tips

To connect to any of the TDK Manager VM's used in ATF testing: **EMU-B TDK Manager VM**, **RPI-B TDK VM**, **RPI-V TDK VM** you can use one of two methods either via vagrant ssh or ssh directly using it's IP

(i) via vagrant

```
# first connect to griffen
get_vpc_key -i 192.168.32.74 -u <username> griffen
sudo su jenkins

# get list of VM's running and their status
vagrant global-status

# output will look like
vagrant global-status
id       name                           provider   state   directory
---------------------------------------------------------------------------------------------------
f6380e9  nfs-server-rdkb                virtualbox running /home/jenkins/cmflab-vagrants/nfs-server/rdkb
0885fe0  tdk-emulator-rdkb              virtualbox running /home/jenkins/cmflab-vagrants/rdkb/emulator/tdk
5e18def  tdk-emulator-rdkb-rpi          virtualbox running /home/jenkins/cmflab-vagrants/rdkb/rpi/tdk
f3fdd9a  tdk-emulator-raven-atf-rpi-rdkv virtualbox running /home/jenkins/cmflab-vagrants/rdkv/rpi/tdk
81a242c  tdk-emulator-rdkb-rpi-e2e      virtualbox running /home/jenkins/cmflab-vagrants/rdkb/rpi/tdk-e2e
1b99642  rdkb-webserver                 virtualbox running /home/jenkins/cmflab-vagrants/rdkb/webserver

# run command `vagrant ssh <id>`
vagrant ssh 0885fe0

# some other vagrant commands
vagrant status <id>
vagrant halt <id>
vagrant up <id>
```

(ii) via ssh

```
# first connect to griffen or some other lab server machine (e.g. heron/corvus/raven)
get_vpc_key -i 192.168.32.74 -u <username> griffen
sudo su jenkins

# ssh to TDK VM using it's IP
# TDK Manager Emulator
ssh -o "StrictHostKeyChecking=no" vagrant@10.5.25.40
ssh -o "StrictHostKeyChecking=no" vagrant@192.168.32.63
http://192.168.32.63/rdk-test-tool/execution/create

# TDK Manager RPI-B
ssh -o "StrictHostKeyChecking=no" vagrant@10.5.25.41
ssh -o "StrictHostKeyChecking=no" vagrant@192.168.32.69
http://192.168.32.69/rdk-test-tool/execution/create

# TDK Manager RPI-V
ssh -o "StrictHostKeyChecking=no" vagrant@10.5.25.43
ssh -o "StrictHostKeyChecking=no" vagrant@192.168.32.70
http://192.168.32.70/rdk-test-tool/execution/create
```

To connect to any of the 8 EMU-V TDK VM's

```
# first connect to heron
get_vpc_key -i 192.168.32.76 -u <uname> heron
sudo su jenkins

# get list of VM's running and their status
vagrant global-status

# output will list all emu-v VM's and the following TDK VM's
vagrant global-status
id       name           provider   state   directory
--------------------------------------------------------------------------------------------------------------------
--------------------------------
eeacce1  tdk-emulator-1 virtualbox running /home/jenkins/manager-vagrant
c4c70a1  tdk-emulator-2 virtualbox running /home/jenkins/manager-vagrant
148b8d3  tdk-emulator-3 virtualbox running /home/jenkins/manager-vagrant
09ef373  tdk-emulator-4 virtualbox running /home/jenkins/manager-vagrant
e778206  tdk-emulator-5 virtualbox running /home/jenkins/manager-vagrant
9633df6  tdk-emulator-6 virtualbox running /home/jenkins/manager-vagrant
9e17573  tdk-emulator-7 virtualbox running /home/jenkins/manager-vagrant
ae67a6f  tdk-emulator-8 virtualbox running /home/jenkins/manager-vagrant

# run command `vagrant ssh <id>`
vagrant ssh eeacce1

# some other vagrant commands
vagrant status <id>
vagrant halt <id>
vagrant up <id>

# if vagrant-global status does not work (can be buggy) then cd into dir where vagrants are run from and run
command vagrant status
cd /home/jenkins/manager-vagrant

vagrant status
Current machine states:

tdk-emulator-1            running (virtualbox)
tdk-emulator-2            running (virtualbox)
tdk-emulator-3            running (virtualbox)
tdk-emulator-4            running (virtualbox)
tdk-emulator-5            running (virtualbox)
tdk-emulator-6            running (virtualbox)
tdk-emulator-7            running (virtualbox)
tdk-emulator-8            running (virtualbox)

vagrant ssh tdk-emulator-1

# to ssh using IP , cat the VagrantFile to get the IP for the manager you want
cat /home/jenkins/manager-vagrant/Vagrantfile

tdk_managers = [
  {:hostname => 'tdk-emulator-1', :ip => '192.168.21.31', :box => 'ubuntu/trusty64'},
  {:hostname => 'tdk-emulator-2', :ip => '192.168.21.32', :box => 'ubuntu/trusty64'},
  {:hostname => 'tdk-emulator-3', :ip => '192.168.21.33', :box => 'ubuntu/trusty64'},
  {:hostname => 'tdk-emulator-4', :ip => '192.168.21.34', :box => 'ubuntu/trusty64'},
  {:hostname => 'tdk-emulator-5', :ip => '192.168.21.35', :box => 'ubuntu/trusty64'},
  {:hostname => 'tdk-emulator-6', :ip => '192.168.21.36', :box => 'ubuntu/trusty64'},
  {:hostname => 'tdk-emulator-7', :ip => '192.168.21.38', :box => 'ubuntu/trusty64'},
  {:hostname => 'tdk-emulator-8', :ip => '192.168.21.39', :box => 'ubuntu/trusty64'}
]

 ssh -o "StrictHostKeyChecking=no" vagrant@192.168.21.31 <-- this is currently looking for password, need to
check with ALAN
```

Some General TDK Manager VM Commands

```
# check if tftp server is running on the TDK Manager VM
vagrant@tdk-emulator-rdkb:~$ ps -ef | grep tftp
vagrant  12660 12630  0 12:17 pts/0    00:00:00 grep --color=auto tftp
root     13444 13389  0 Sep18 ?        00:00:00 sudo python /var/lib/tomcat6/webapps/rdk-test-tool/fileStore
/tftp_server.py 69 /var/lib/tomcat6/webapps/rdk-test-tool/logs/logs/
root     13446 13444  0 Sep18 ?        00:00:05 python /var/lib/tomcat6/webapps/rdk-test-tool/fileStore
/tftp_server.py 69 /var/lib/tomcat6/webapps/rdk-test-tool/logs/logs/

# Manually start TDK TFTP Server on a TDK VM (tftp server is used on manager to transfer agent logs from DUT)
sudo python /var/lib/tomcat6/webapps/rdk-test-tool/fileStore/tftp_server.py 69 /var/lib/tomcat6/webapps/rdk-
test-tool/logs/logs/ &

# kill tftp serrver
sudo pkill -f tftp

# to manually set the route on TDK Manager VM (this is needed if you can't connect to the webui of VM via your
browser), e.g. can happen when VM was restarted abnormally
route del default
sudo route del default; sudo route add default gw 192.168.32.1 dev eth1

# to change interface or mechanism for agent log transfer (can be set to tftp/REST) edit tm.config file and
restart tomcat
vi /var/lib/tomcat6/webapps/rdk-test-tool/fileStore/tm.config
sudo pkill -f tftp
sudo service tomcat6 stop
sudo service tomcat6 start
```

8.7. Provisioning RPI SD Cards for first time use:

The CMF procedure for flashing RPI's using UBOOT for automated testing is documented in RPI flashing procedure documentation

8.7.1. Instructions for flashing RPI SD images (MAC)

To format SD card using SD Formatter

- https://www.sdcard.org/downloads/formatter/eula_mac/index.htm
- Format as FAT32
- Note could probably also use diskutil

To flash RDK SD cards with a prebuilt sd image:

- https://www.balena.io/etcher/
- manually (be careful!):
    - https://www.raspberrypi.org/documentation/installation/installing-images/mac.md

8.7.2. Instructions for partitioning RPI SSD (from Ubuntu Host)

Best results on linux seen with GNOME Partition Editor (gparted)

Install from universal repositories, requires super user permissions to run the program.

Partitioning Info:

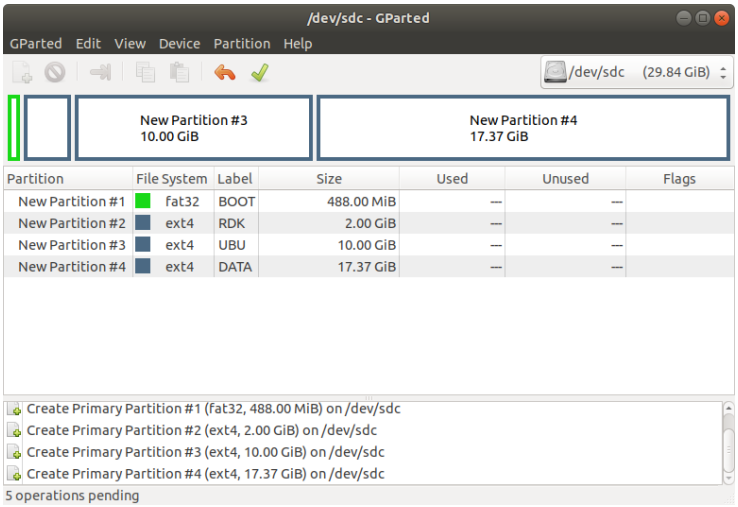
16GB SSD Layout:

Partition	Label	Designation	File System	Size (MiB)
1	BOOT	Primary	Fat32	488
2	RDK	Primary	EXT4	2048
3	UBU	Primary	EXT4	10240
4	DATA	Primary	EXT4	(Remaining Space)

32GB SSD Layout:

Partition	Label	Designation	File System	Size (MiB)

1	BOOT	Primary	Fat32	488
2	RDK	Primary	EXT4	10240
3	UBU	Primary	EXT4	10240
4	DATA	Primary	EXT4	(Remaining Space)



8.7.3. Instructions for flashing boot & root filesystems onto SSD (Nix)

Important: the boot partition must be populated at a bare minimum before the ssd is put into a DUT for the CMF Automation to succeed.

The partitions created in the previous section should be mounted to your filesystem, in recent ubuntu, these should mount to

- /media/($USER)/BOOT
- /media/($USER)/RDK
- /media/($USER)/UBU

```
# Acquire an RDK Build archive:
$ aws s3 ls s3://rdkcmf-artifacts/jobs/community-build-rdkv-raspberrypi/
# Choose latest build number, e.g. 1472, check files therein, then copy the build to your local workspace:
$ aws s3 cp s3://rdkcmf-artifacts/jobs/community-build-rdkv-raspberrypi/1472/test-archive-rdk-hybrid-raspberrypi-community-nightly-<build-id>.tar.gz .
# Extract the archive into a local directory:
$ mkdir rdk-hybrid && tar xvzf test-archive-rdk-hybrid-raspberrypi-community-nightly-<build-id>.tar.gz -C rdk-hybrid
# Configure u-boot as explicit kernel image for first time sd provisioning:
# Edit the file rdk-hybrid/boot/config.txt, find the line `#kernel=""` and change it to `kernel=u-boot.bin`, save and close the file.
# Copy the boot contents to BOOT partition
$ cp -r rdk-hybrid/boot /media/($USER)/BOOT
# Extract the root filesystem to RDK partition
$ tar xvjf rdk-hybrid/rootfs/rdk-generic-hybrid-wpe-image-raspberrypi-rdk-hybrid.tar.bz2 -C /media/($USER)/RDK

# Provisioning RDK-CMF-UBUNTU (WiFi Client):
# Acquire RDK-CMF-Ubuntu:
$ aws s3 ls s3://rdkcmf-artifacts/test-artifacts/test-ubuntu-artifacts
# Choose latest build number, e.g. 3, check files therein, then copy the build root file system (rootfs) to your local workspace (No need for boot filesystem):
$ aws s3 cp s3://rdkcmf-artifacts/test-artifacts/test-ubuntu-artifacts/3/ubuntu_rootfs_<build-sha>.tar.bz2 .
# Extract the archive into the SSD UBU Partition:
$ tar xvjf ubuntu_rootfs_<build-sha>.tar.bz2 -C /media/($USER)/UBU

# Ensure all Read/Write cases complete
$ sudo sync

Unmount/Eject the sd card and transfer to the DUT
```

8.7.4. Useful RPI setup links:

- https://projects.raspberrypi.org/en/projects/raspberry-pi-setting-up/3
- https://www.raspberrypi.org/help/
- https://www.raspberrypi.org/documentation/installation/