MITSUBISHI ELECTRIC
Changes for the Better

Continuous Integration and Autotest Environment using Fuego

Kengo IBE and Kenji TADANO
Mitsubishi Electric

13th, October 2016

Embedded Linux Conference Europe

MITSUBISHI ELECTRIC CORPORATION

Who am I

- Kengo IBE
  - Embedded Linux Developer at the Mitsubishi Electric Information Technology R&D Center
  - Also I’ve been on loan to Linux Foundation

- Kenji TADANO
  - Embedded Linux Developer at the Mitsubishi Electric Information Technology R&D Center

- We have been collaborating with OSS community!!
  - LTSI : Long Term Support Initiative
  - AGL : Automotive Grade Linux

Outline

- Overview
- Back Ground
- Test Framework / Fuego
- Further Improvement
  - Running a test automatically
  - Utilizing OSS test suite
- Conclusion

Overview

- At ELCE 2015, we showed how to customize and run Fuego (LTSI Test Framework) with your test target

- On this session, we share how to utilize Fuego as test framework for embedded systems, based on our experience

Back Ground

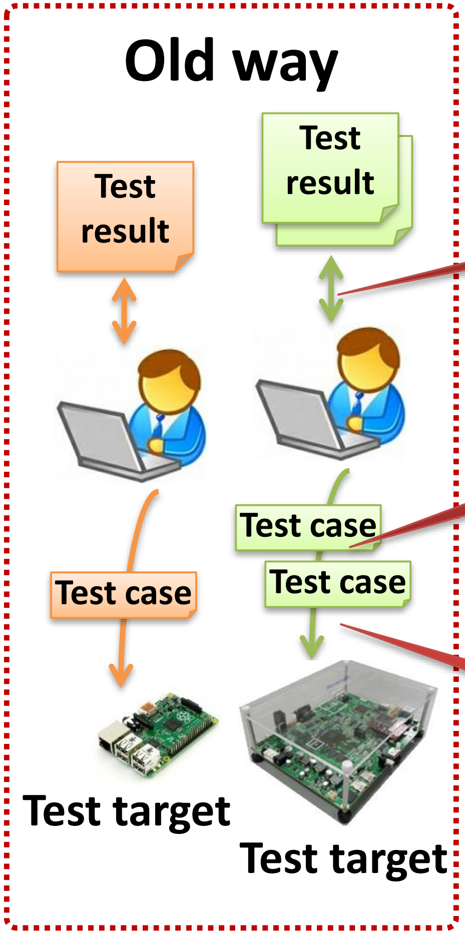
- For embedded systems, Linux kernel is used widely

- Because Linux Kernel is very huge, the discussion on how to ensure the quality is often occurred

- Introducing test framework such as Fuego into development, should ensure the quality effectively

Introduction of Test Framework

- When running test on target, there are some issues

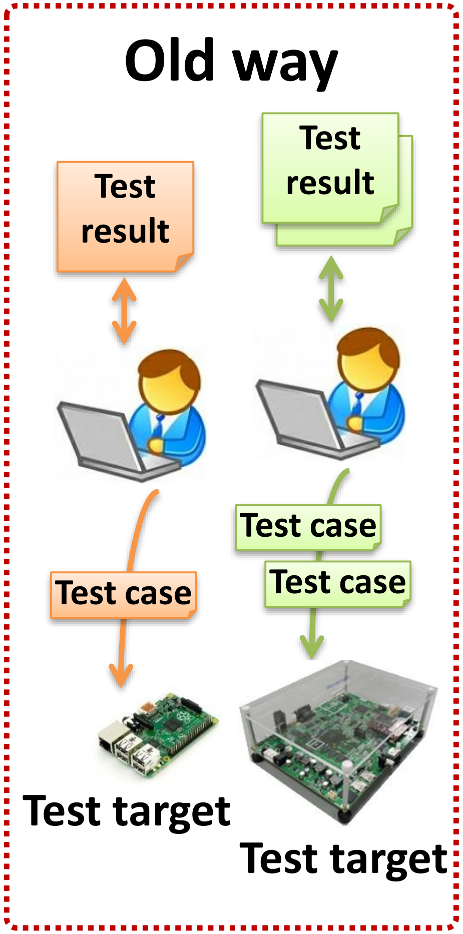


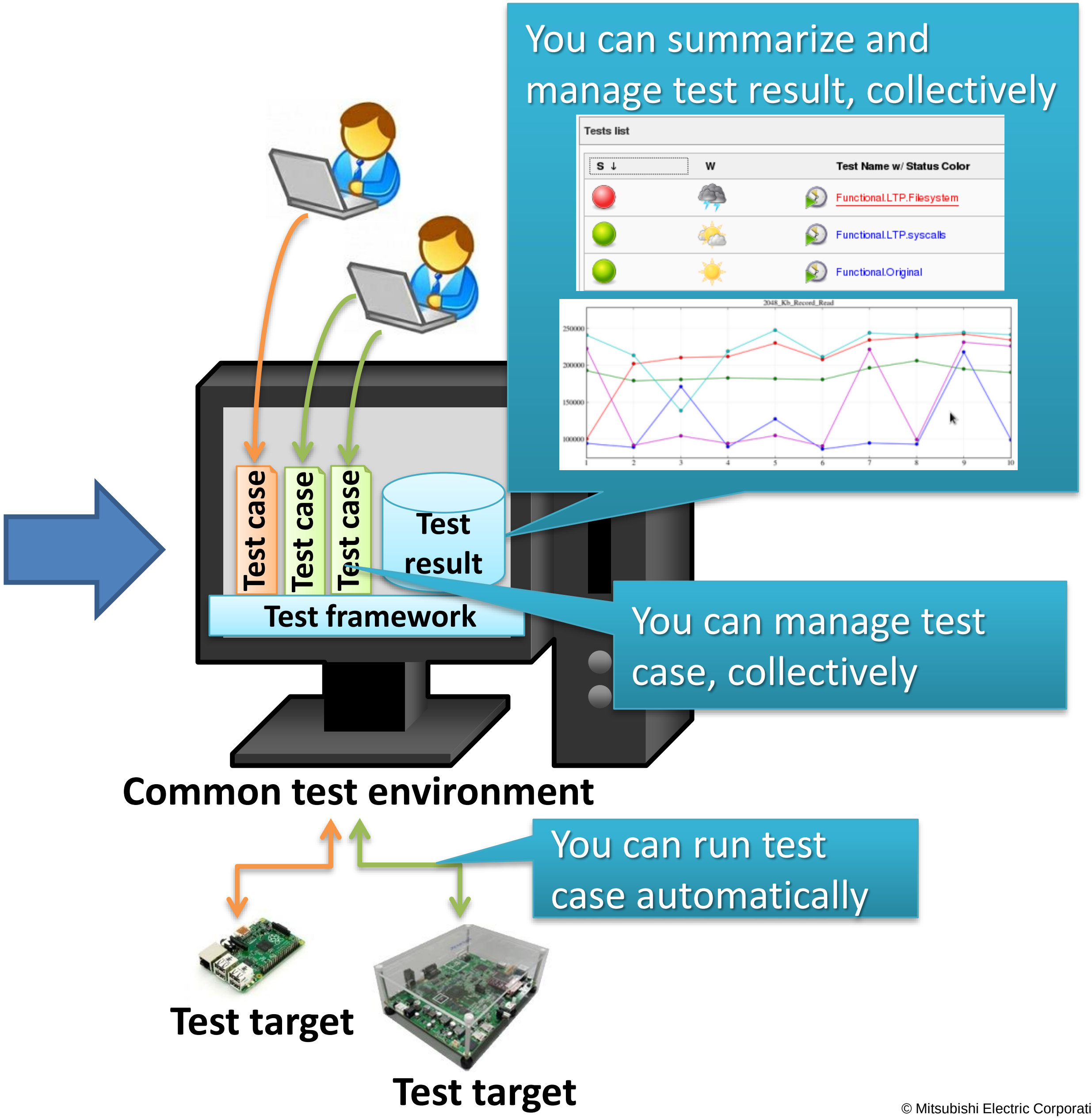
You need to check test result one by one

You need to create and manage test case for each target

You need to run test case manually

Introduction of Test Framework





You can summarize and manage test result, collectively

Tests list

S ↓	W	Test Name w/ Status Color
●	☁	Functional.LTP.Filesystem
●	☀	Functional.LTP.syscalls
●	☀	Functional.Original

You can manage test case, collectively

You can run test case automatically

Why Fuego

- Fuego is one of the test framework that is created by LTSI project, based on Jenkins
- Fuego is OSS that anyone can use and contribute
- Some manufacturers are using Fuego as test framework
- Recently, AGL chose Fuego as standard test environment(AGL-JTA)

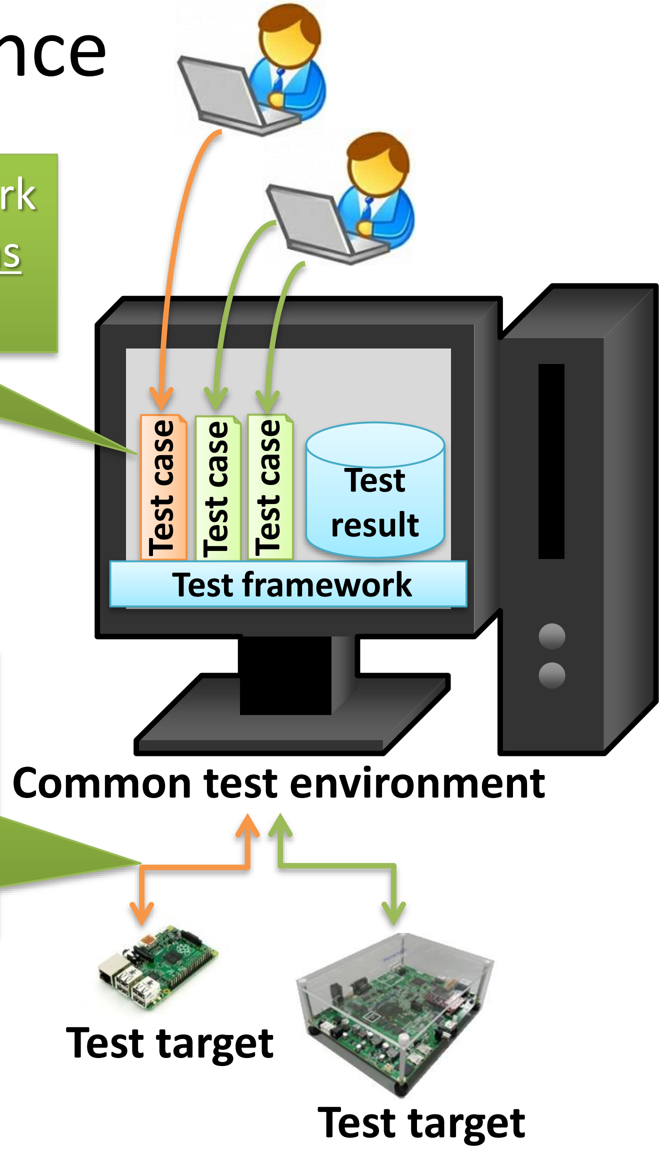
You can choose Fuego and introduce it into your development. Fuego includes many useful functions but…

Further Improvement

- To become more convenient, share some ideas using our experience

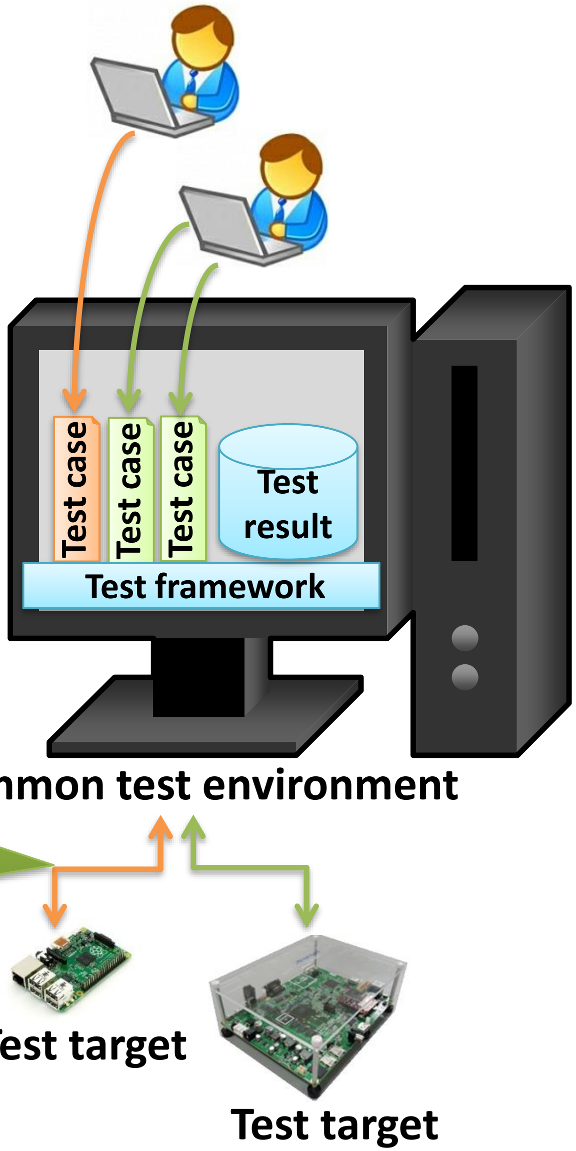
Creating all test cases is tough work
<u>→ Utilize OSS test suite as much as possible</u>

Waste much time for executing test, repeatedly
<u>→ Introduce the automated test that is triggered by software update</u>



Running a test automatically

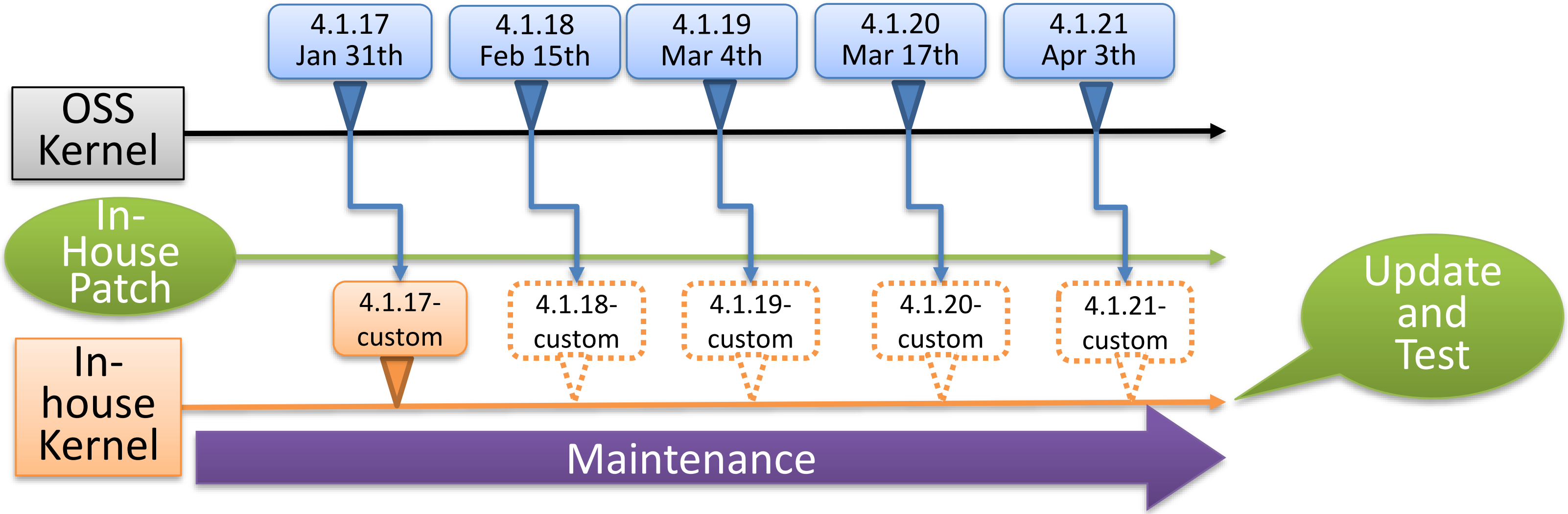
- How to introduce the automated test that is triggered by software update?



Waste much time for executing test, repeatedly
→ Introduce the automated test that is triggered by software update

Current Situation

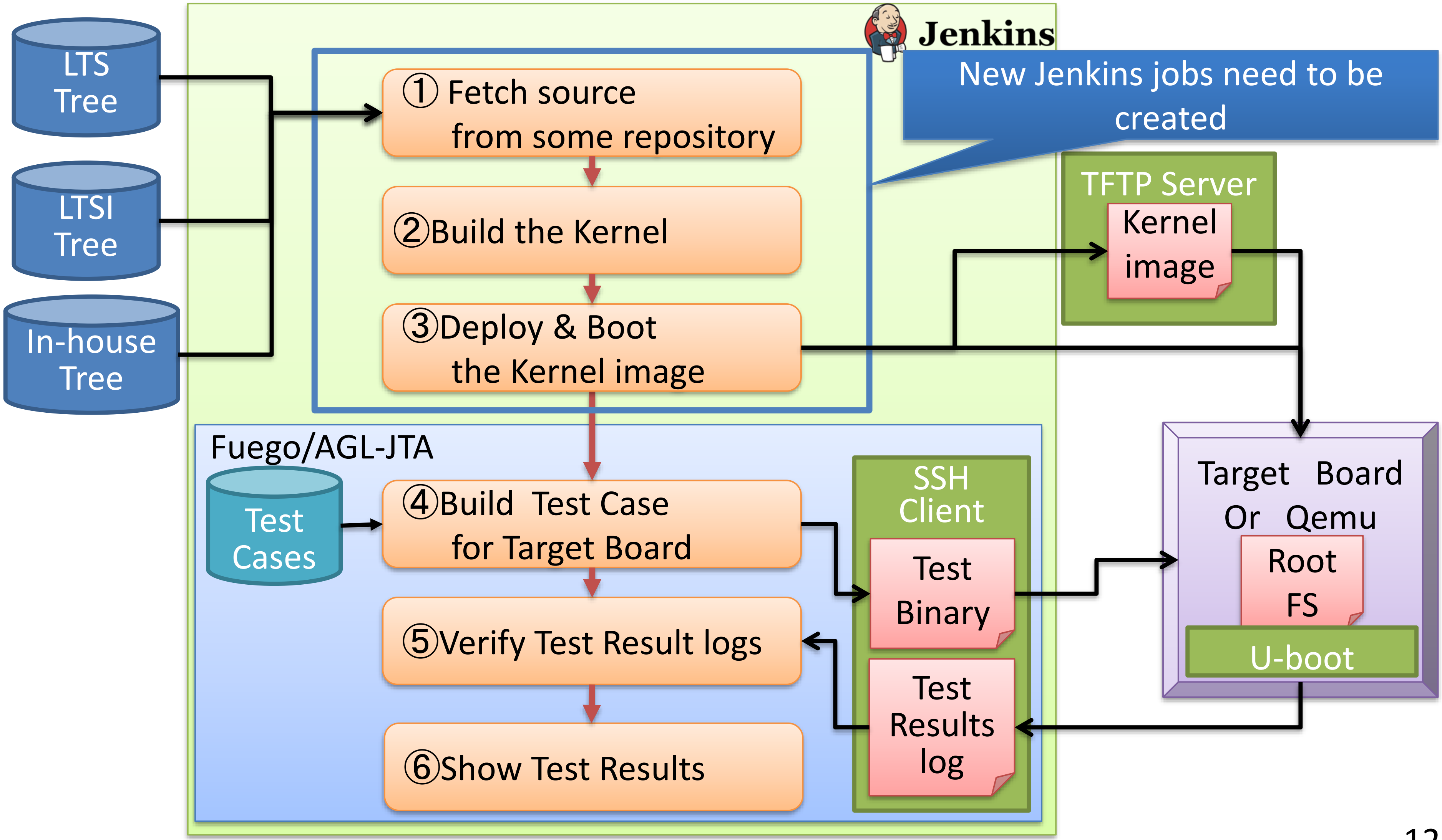
- Release new version kernel cyclically
  - For maintenance, run test for new kernel version each time
  - When detecting bug, it needs to be fixed manually



Share how to run test automatically when OSS updates

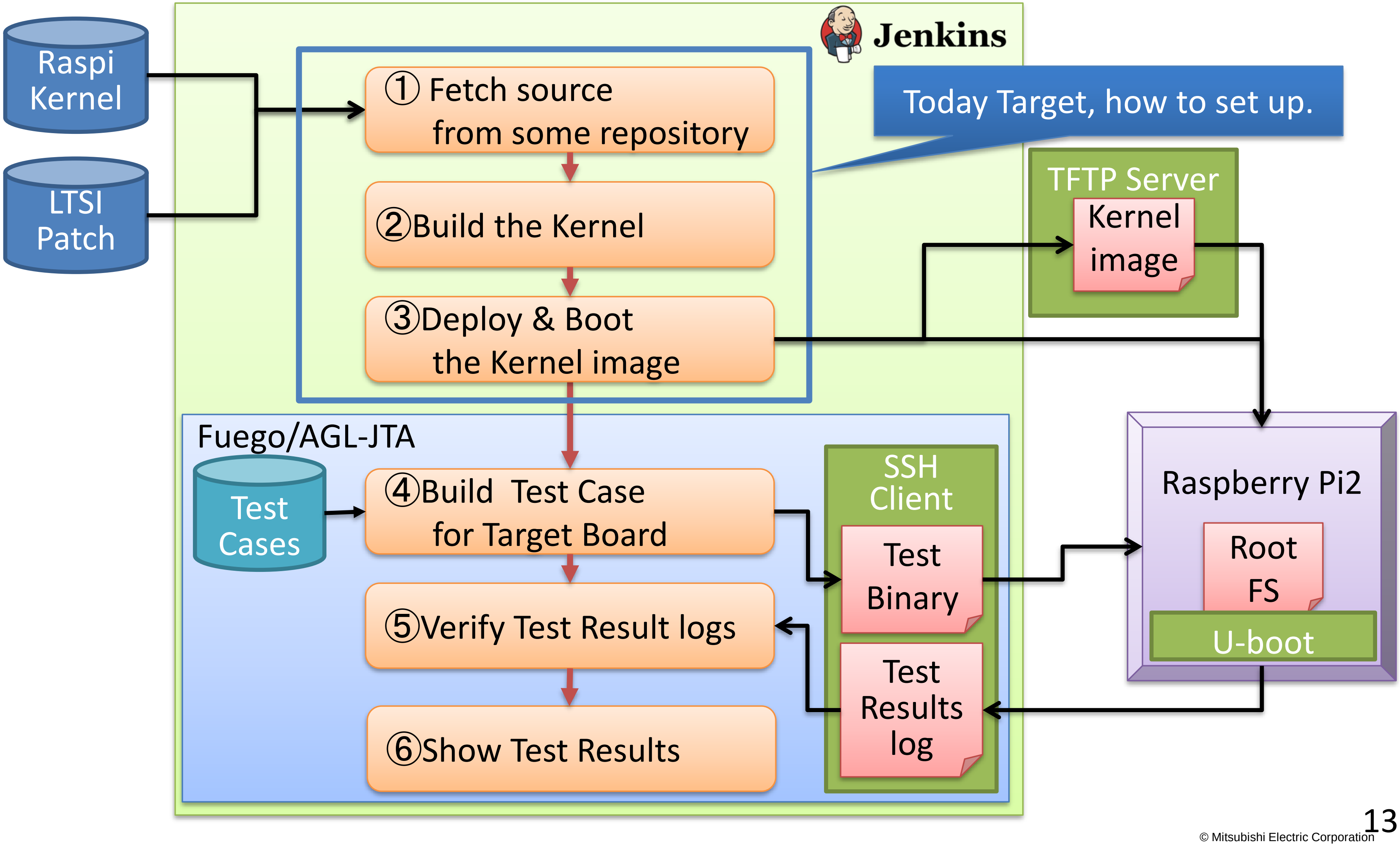
Automated test environment using Fuego

- Overview of Automated test environment



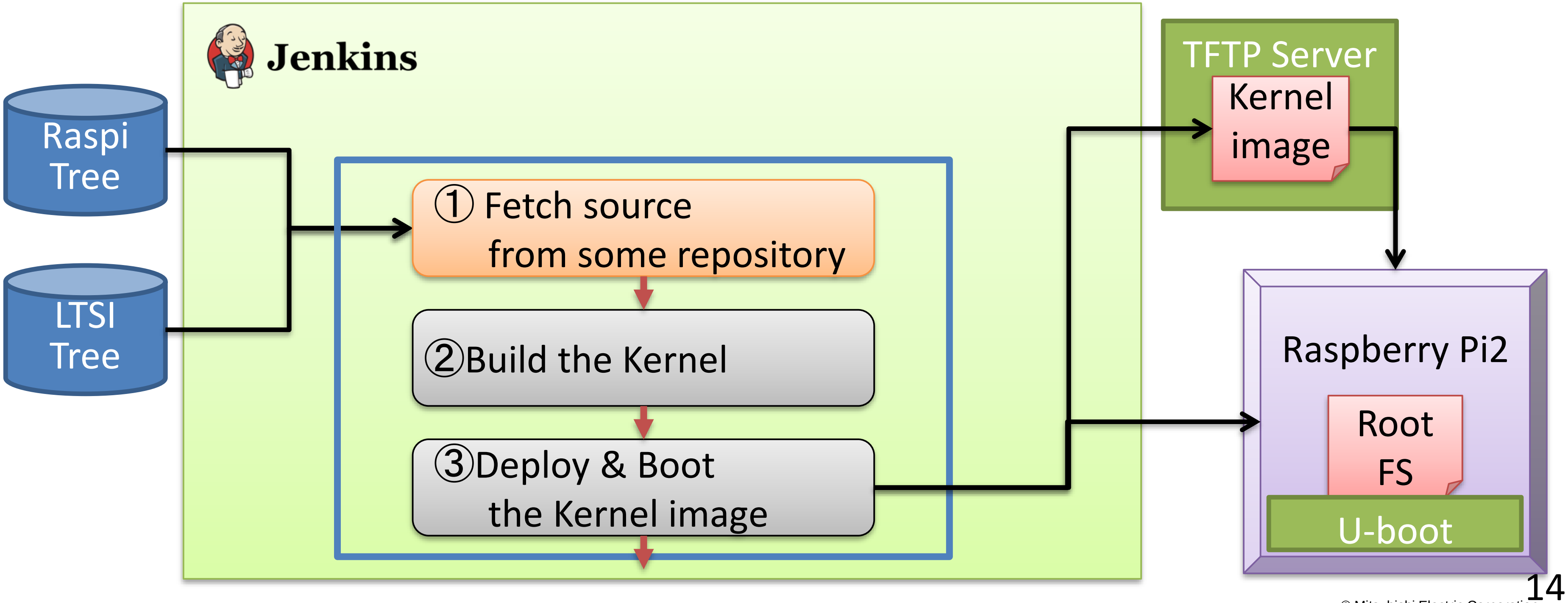
Case study : Raspberry Pi kernel Tree + LTSI patch

- When Raspberry Pi kernel is updated, Fuego starts to test



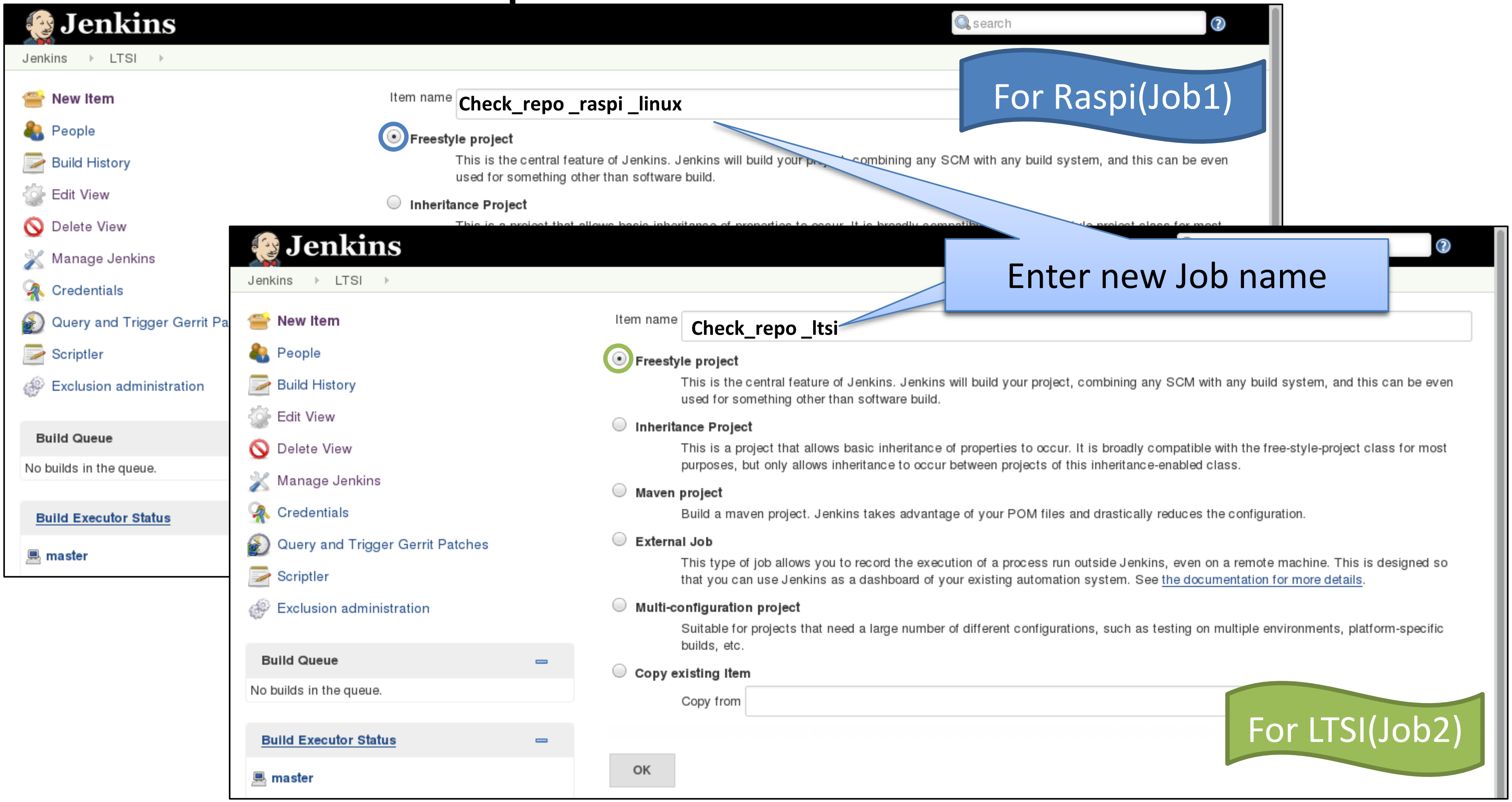
①Fetching source from repositories

- 2 jobs to fetch sources
  - Job1: Fetch Raspberry Pi kernel Tree
  - Job2: Fetch LTSI patch Tree



① cont. (Create Job1 & Job2)

- Job1: Fetch Raspberry Pi kernel Tree
- Job2: Fetch LTSI patch Tree



Jenkins

search

Jenkins ▸ LTSI ▸

- New Item
- People
- Build History
- Edit View
- Delete View
- Manage Jenkins
- Credentials
- Query and Trigger Gerrit Pa
- Scriptler
- Exclusion administration

Build Queue

No builds in the queue.

Build Executor Status

master

Item name: Check_repo _raspi _linux

● Freestyle project

This is the central feature of Jenkins. Jenkins will build your pr... combining any SCM with any build system, and this can be even used for something other than software build.

○ Inheritance Project

For Raspi(Job1)

Enter new Job name

Jenkins

Jenkins ▸ LTSI ▸

- New Item
- People
- Build History
- Edit View
- Delete View
- Manage Jenkins
- Credentials
- Query and Trigger Gerrit Patches
- Scriptler
- Exclusion administration

Build Queue

No builds in the queue.

Build Executor Status

master

Item name: Check_repo _ltsi

● Freestyle project

This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

○ Inheritance Project

This is a project that allows basic inheritance of properties to occur. It is broadly compatible with the free-style-project class for most purposes, but only allows inheritance to occur between projects of this inheritance-enabled class.

○ Maven project

Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.

○ External Job

This type of job allows you to record the execution of a process run outside Jenkins, even on a remote machine. This is designed so that you can use Jenkins as a dashboard of your existing automation system. See the documentation for more details.

○ Multi-configuration project

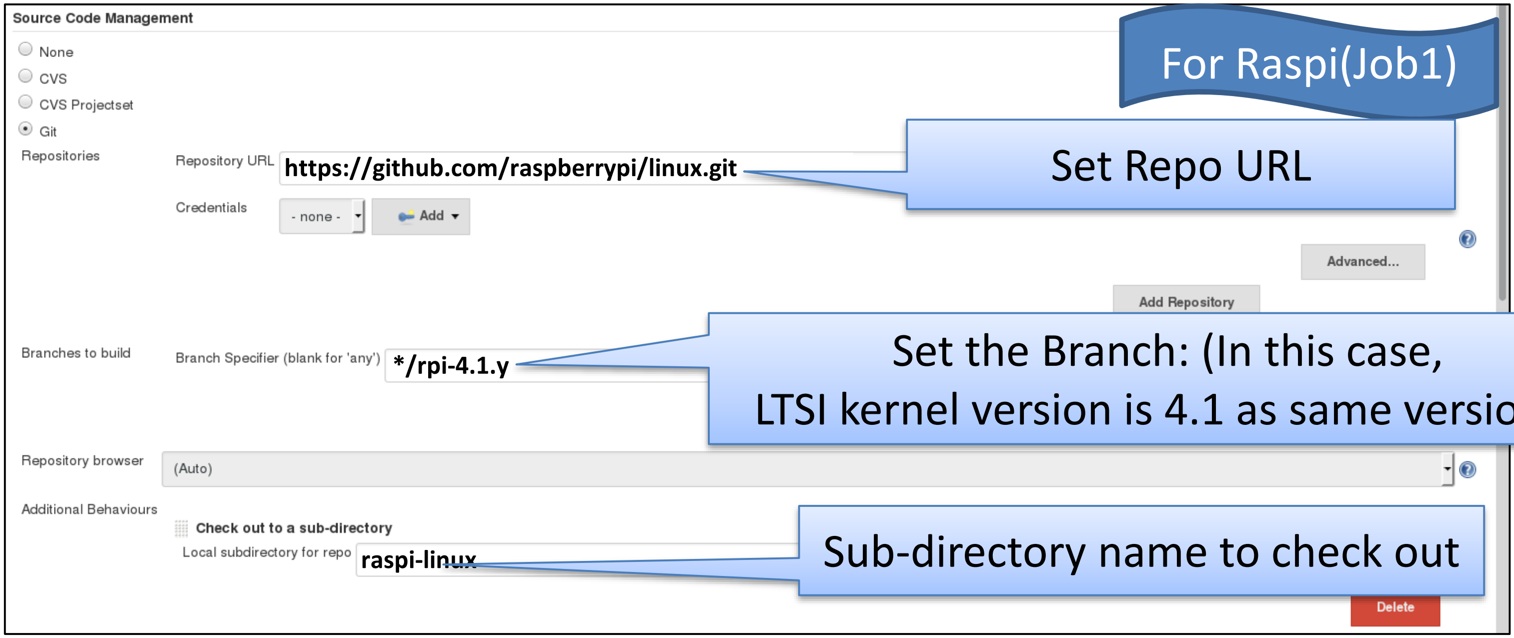
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

○ Copy existing Item

Copy from

For LTSI(Job2)

① cont. (Set Repo URL)



Source Code Management

○ None
○ CVS
○ CVS Projectset
● Git

Repositories

Repository URL: https://github.com/raspberrypi/linux.git

Credentials: - none - | Add

Advanced... | Add Repository

Branches to build

Branch Specifier (blank for 'any'): */rpi-4.1.y

Repository browser: (Auto)

Additional Behaviours

Check out to a sub-directory

Local subdirectory for repo: raspi-linux

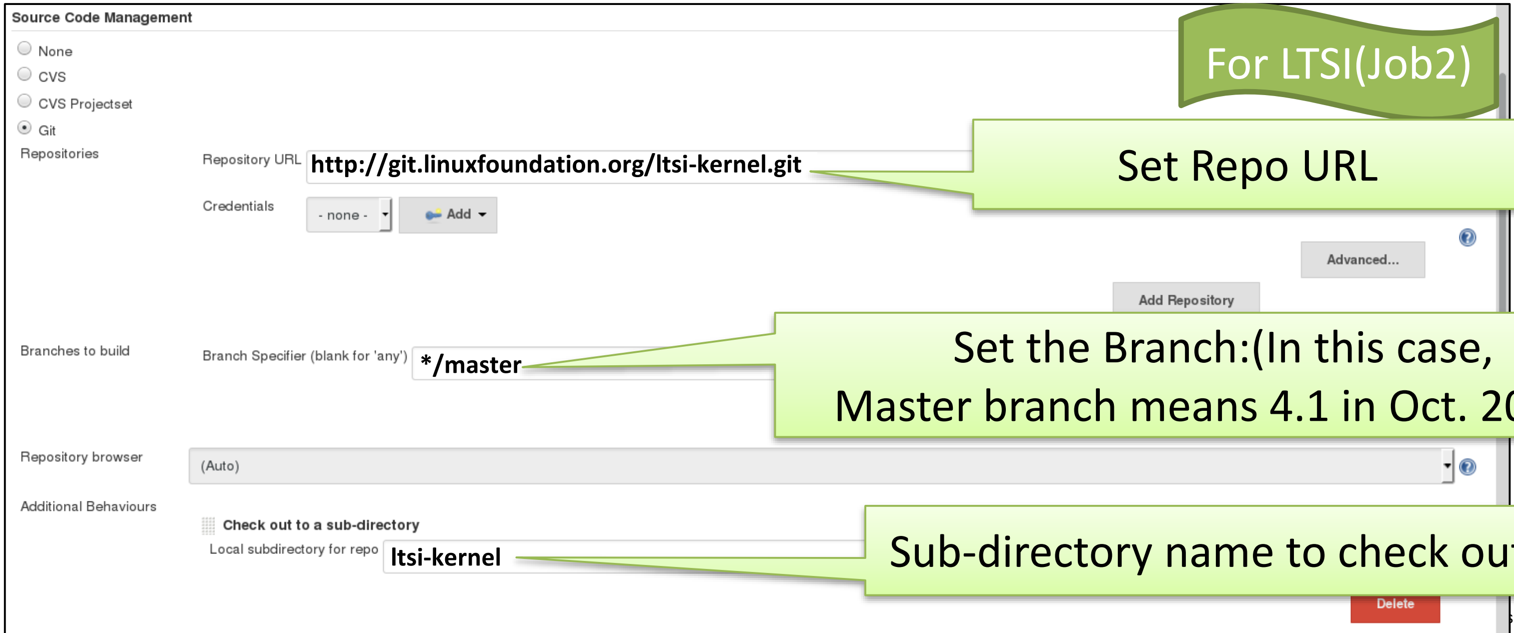
Delete

For Raspi(Job1)

Set Repo URL

Set the Branch: (In this case,
LTSI kernel version is 4.1 as same version.)

Sub-directory name to check out



Source Code Management

○ None
○ CVS
○ CVS Projectset
● Git

Repositories

Repository URL: http://git.linuxfoundation.org/ltsi-kernel.git

Credentials: - none - | Add

Advanced... | Add Repository

Branches to build

Branch Specifier (blank for 'any'): */master

Repository browser: (Auto)

Additional Behaviours

Check out to a sub-directory

Local subdirectory for repo: ltsi-kernel

Delete

For LTSI(Job2)

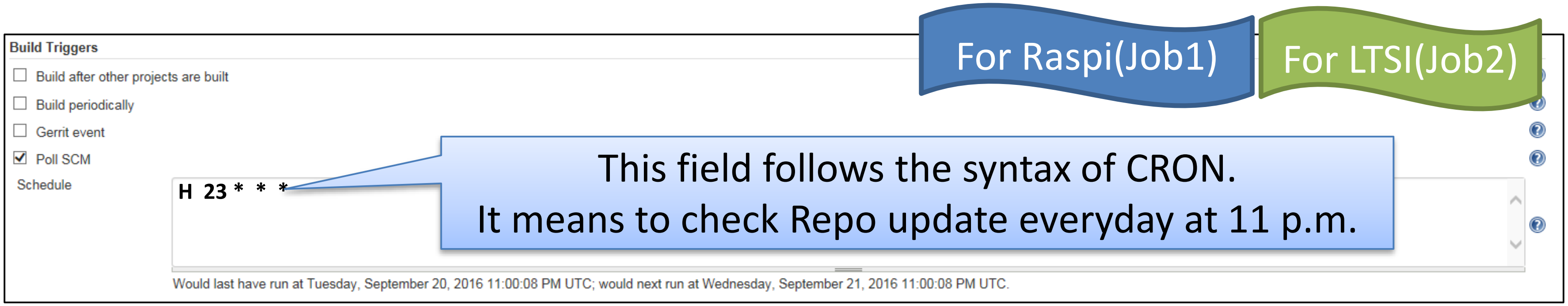
Set Repo URL

Set the Branch:(In this case,
Master branch means 4.1 in Oct. 2016.)

Sub-directory name to check out

① cont. (set schedule to poll repo)

- Set Build Triggers



Build Triggers

☐ Build after other projects are built

☐ Build periodically

☐ Gerrit event

☑ Poll SCM

Schedule

```
H 23 * * *
```

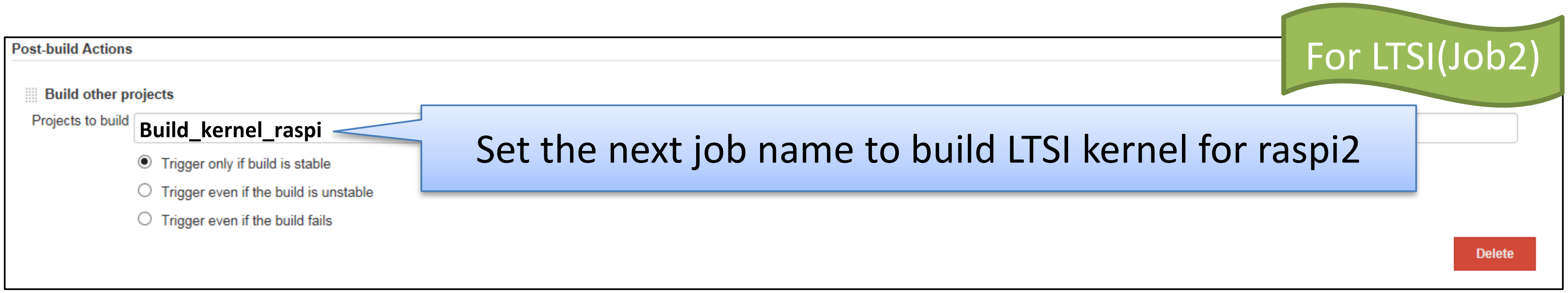
Would last have run at Tuesday, September 20, 2016 11:00:08 PM UTC; would next run at Wednesday, September 21, 2016 11:00:08 PM UTC.

For Raspi(Job1)

For LTSI(Job2)

This field follows the syntax of CRON.
It means to check Repo update everyday at 11 p.m.

- Choose either/both job to kick the kernel build job you like
  - In this situation, Trigger is LTSI update (Job2)



Post-build Actions

Build other projects

Projects to build: **Build_kernel_raspi**

● Trigger only if build is stable

○ Trigger even if the build is unstable

○ Trigger even if the build fails

For LTSI(Job2)

Set the next job name to build LTSI kernel for raspi2

② Build the Kernel

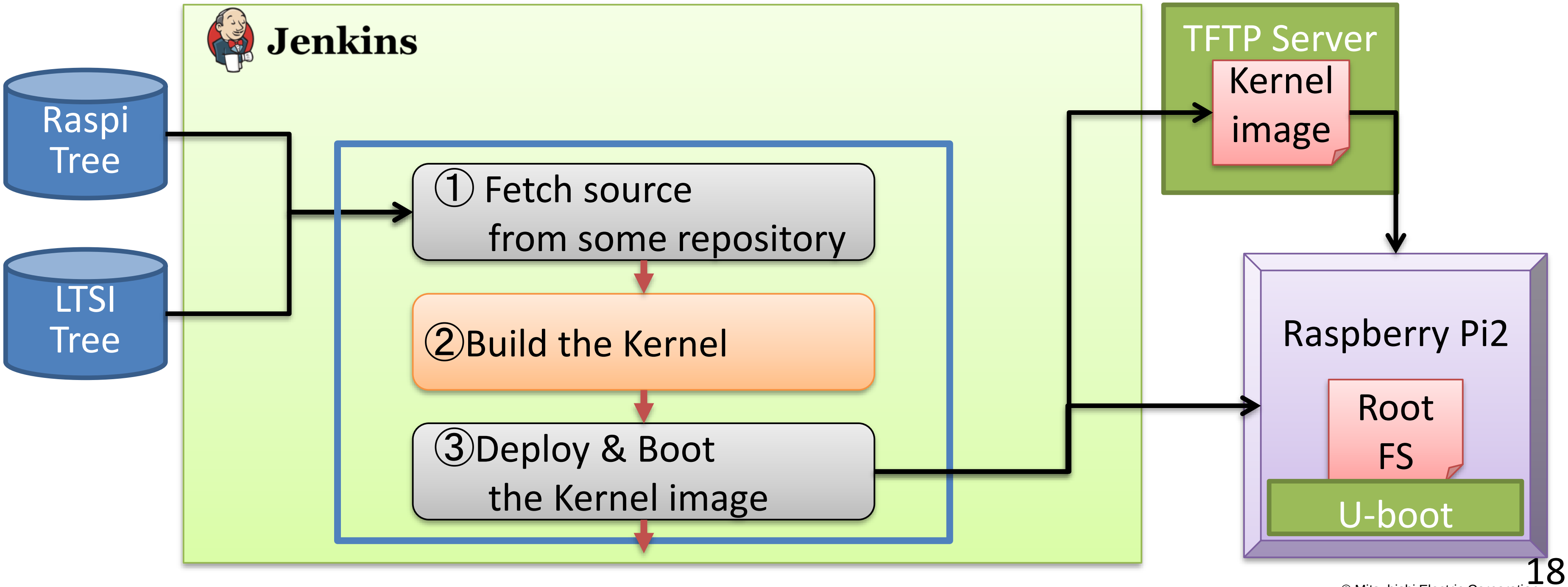
- Create a new job to build Kernel

1) Get the sources from previous jobs

2) Apply the LTSI patches to Raspi kernel

3) Build the LTSI kernel

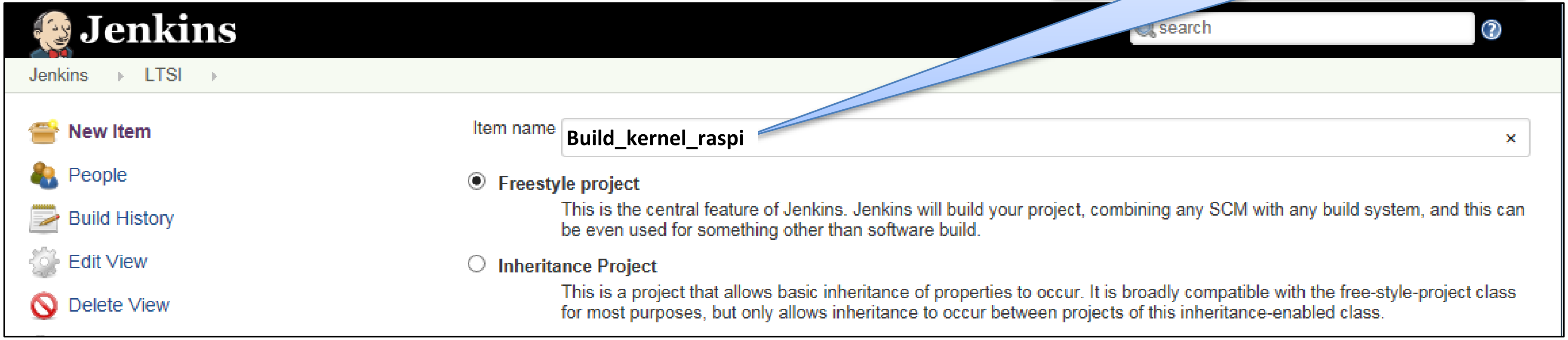
4) Archive the LTSI kernel image and source code



②-1 Get the sources

- Create the new Job

Enter new Job name

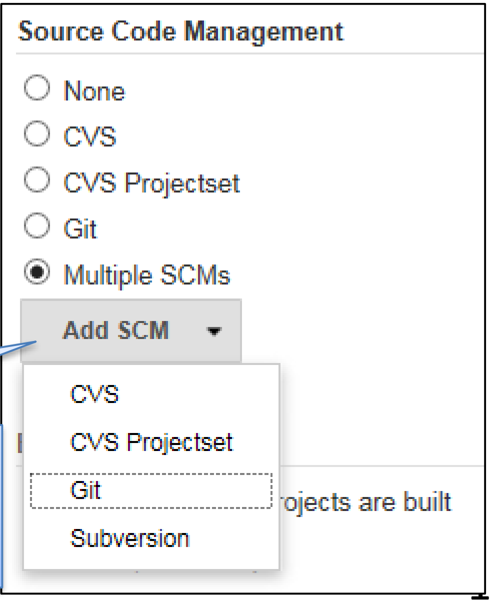


Item name: Build_kernel_raspi ×

● **Freestyle project**
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

○ **Inheritance Project**
This is a project that allows basic inheritance of properties to occur. It is broadly compatible with the free-style-project class for most purposes, but only allows inheritance to occur between projects of this inheritance-enabled class.

- Set the repository
  - Install Multiple SCMs plugin
    - This plugin enables the selection of multiple source code management systems
  - Choose Multiple SCMs



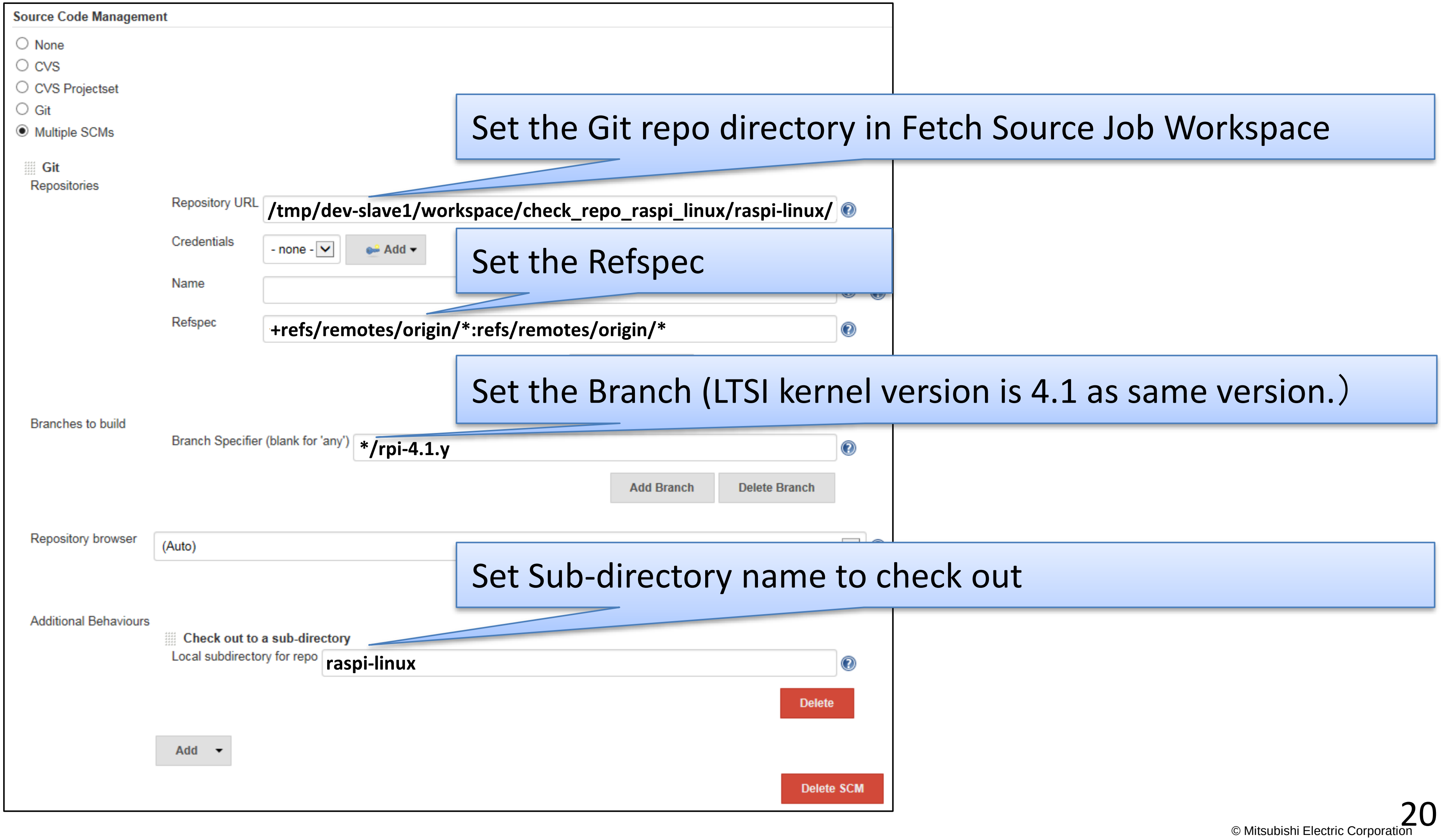
Source Code Management

○ None
○ CVS
○ CVS Projectset
○ Git
● Multiple SCMs

"Select Multiple SCMs" in Source Code Management.
Select Git in "Add SCM" list.

②-1 Get the sources(cont.)

- Set Raspi Git repo of Fetch Source Job



Source Code Management

○ None
○ CVS
○ CVS Projectset
○ Git
● Multiple SCMs

Git
Repositories

Repository URL: /tmp/dev-slave1/workspace/check_repo_raspi_linux/raspi-linux/

Credentials: - none -

Name:

Refspec: +refs/remotes/origin/*:refs/remotes/origin/*

Branches to build

Branch Specifier (blank for 'any'): */rpi-4.1.y

Repository browser: (Auto)

Additional Behaviours

Check out to a sub-directory
Local subdirectory for repo: raspi-linux

Set the Git repo directory in Fetch Source Job Workspace

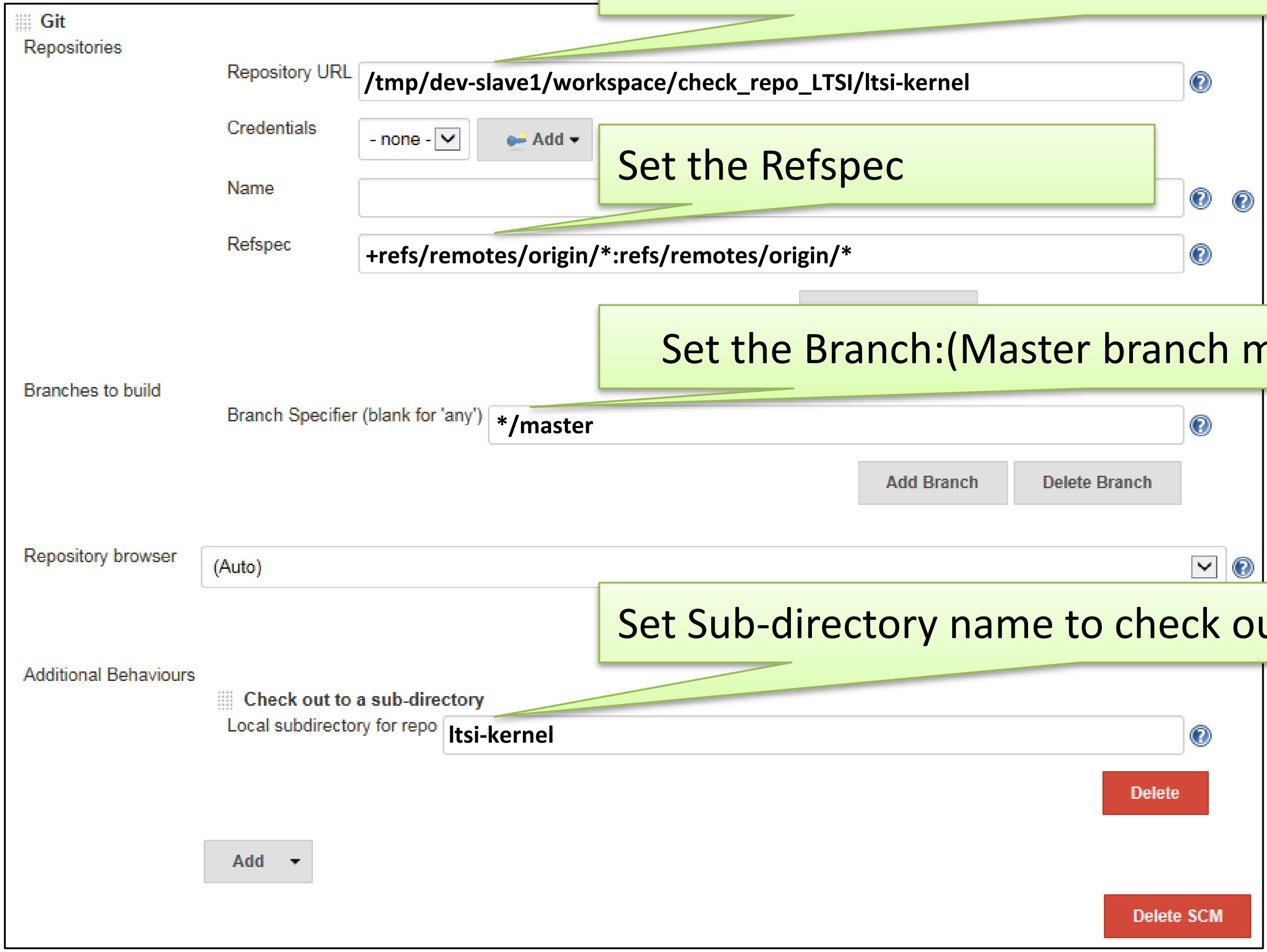
Set the Refspec

Set the Branch (LTSI kernel version is 4.1 as same version.)

Set Sub-directory name to check out

②-1 Get the sources(cont.)

- Set LTSI Git repo of Fetch Source Job



Set the Git repo directory in Fetch Source Job Workspace

Set the Refspec

Set the Branch:(Master branch means 4.1 in Oct. 2016.)

Set Sub-directory name to check out

Git

Repositories

Repository URL: /tmp/dev-slave1/workspace/check_repo_LTSI/ltsi-kernel

Credentials: - none -

Name:

Refspec: +refs/remotes/origin/*:refs/remotes/origin/*

Branches to build

Branch Specifier (blank for 'any'): */master

Repository browser: (Auto)

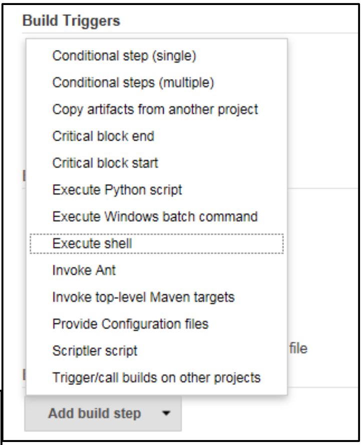
Additional Behaviours

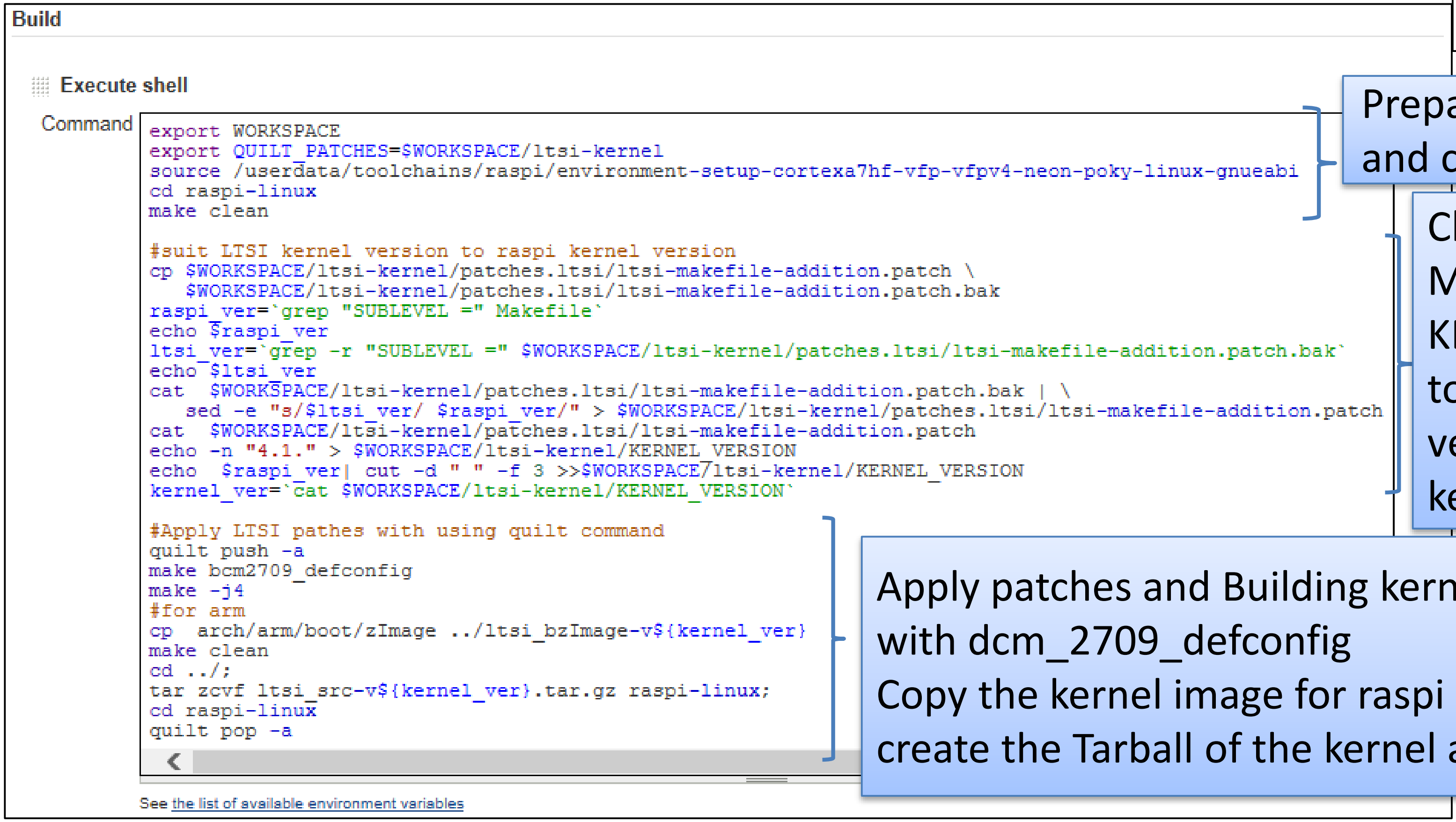
Check out to a sub-directory

Local subdirectory for repo: ltsi-kernel

②-2,3 Applying patches & Build the kernel

- Describe a shell script for building
  - Selecting “Execute shell” in “Add build step”
- Apply patches and Building LTSI kernel


Build Triggers
Conditional step (single)
Conditional steps (multiple)
Copy artifacts from another project
Critical block end
Critical block start
Execute Python script
Execute Windows batch command
Execute shell
Invoke Ant
Invoke top-level Maven targets
Provide Configuration files
Scripter script
Trigger/call builds on other projects
file
Add build step



```
export WORKSPACE
export QUILT_PATCHES=$WORKSPACE/ltsi-kernel
source /userdata/toolchains/raspi/environment-setup-cortexa7hf-vfp-vfpv4-neon-poky-linux-gnueabi
cd raspi-linux
make clean

#suit LTSI kernel version to raspi kernel version
cp $WORKSPACE/ltsi-kernel/patches.ltsi/ltsi-makefile-addition.patch \
   $WORKSPACE/ltsi-kernel/patches.ltsi/ltsi-makefile-addition.patch.bak
raspi_ver=`grep "SUBLEVEL =" Makefile`
echo $raspi_ver
ltsi_ver=`grep -r "SUBLEVEL =" $WORKSPACE/ltsi-kernel/patches.ltsi/ltsi-makefile-addition.patch.bak`
echo $ltsi_ver
cat  $WORKSPACE/ltsi-kernel/patches.ltsi/ltsi-makefile-addition.patch.bak | \
   sed -e "s/$ltsi_ver/ $raspi_ver/" > $WORKSPACE/ltsi-kernel/patches.ltsi/ltsi-makefile-addition.patch
cat  $WORKSPACE/ltsi-kernel/patches.ltsi/ltsi-makefile-addition.patch
echo -n "4.1." > $WORKSPACE/ltsi-kernel/KERNEL_VERSION
echo  $raspi_ver| cut -d " " -f 3 >>$WORKSPACE/ltsi-kernel/KERNEL_VERSION
kernel_ver=`cat $WORKSPACE/ltsi-kernel/KERNEL_VERSION`

#Apply LTSI pathes with using quilt command
quilt push -a
make bcm2709_defconfig
make -j4
#for arm
cp  arch/arm/boot/zImage ../ltsi_bzImage-v${kernel_ver}
make clean
cd ../;
tar zcvf ltsi_src-v${kernel_ver}.tar.gz raspi-linux;
cd raspi-linux
quilt pop -a
```

See the list of available environment variables

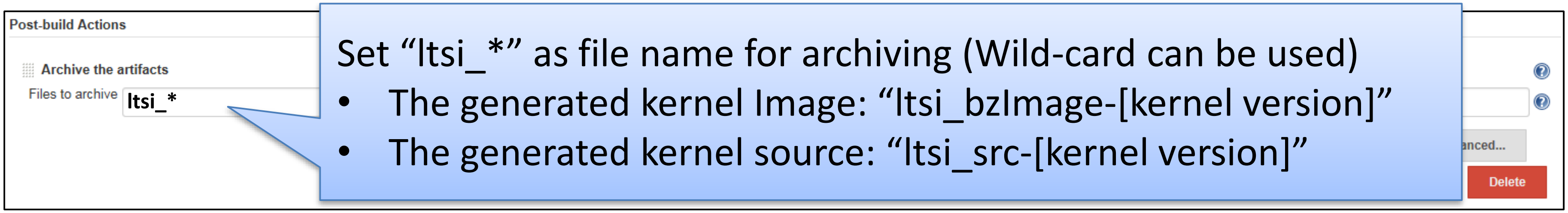
Prepare to use quilt and cross compiler

Change patch of Makefile and KERNEL_VERSION to suit LTSI kernel version to Raspi kernel version

Apply patches and Building kernel of raspi2 with dcm_2709_defconfig
Copy the kernel image for raspi and create the Tarball of the kernel applied patches

②-4 Archive LTSI kernel Image & Source

- Set Post-build Actions
  - Archive the artifacts



Post-build Actions

Archive the artifacts

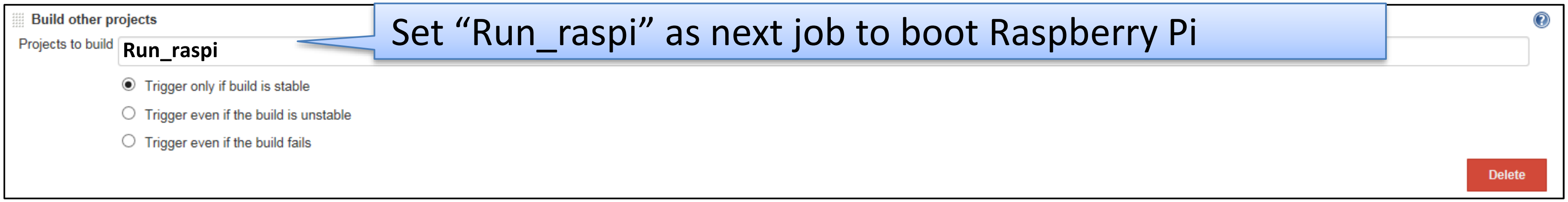
Files to archive: ltsi_*

anced...

Delete

Set “ltsi_*” as file name for archiving (Wild-card can be used)
- The generated kernel Image: “ltsi_bzImage-[kernel version]”
- The generated kernel source: “ltsi_src-[kernel version]”

  - Build other projects



Build other projects

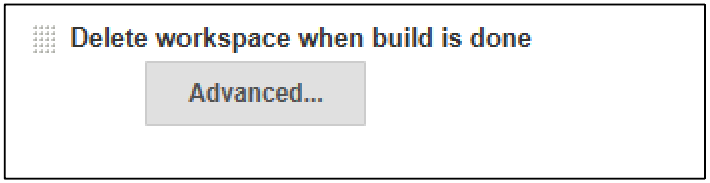
Projects to build: Run_raspi

● Trigger only if build is stable
○ Trigger even if the build is unstable
○ Trigger even if the build fails

Delete

Set “Run_raspi” as next job to boot Raspberry Pi

  - Delete workspace when build is done (optional)
    - Using Workspace Cleanup Plugin



Delete workspace when build is done

Advanced...

Recommend to set “Delete workspace option”, not using the workspace cache

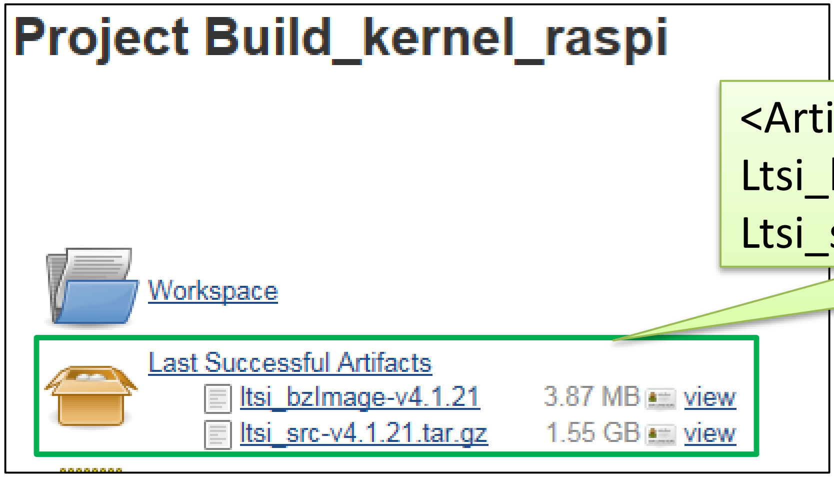
② Build result and the artifacts

- Console Output

```
....
Archiving artifacts
[WS-CLEANUP] Deleting project workspace...[WS-CLEANUP] done
Warning: you have no plugins providing access control for builds, so falling back
to legacy behavior of permitting any downstream builds to be triggered
Triggering a new build of Run_raspi
Finished: SUCCESS
```

Complete this Job!

- The artifacts list



Project Build_kernel_raspi

Workspace

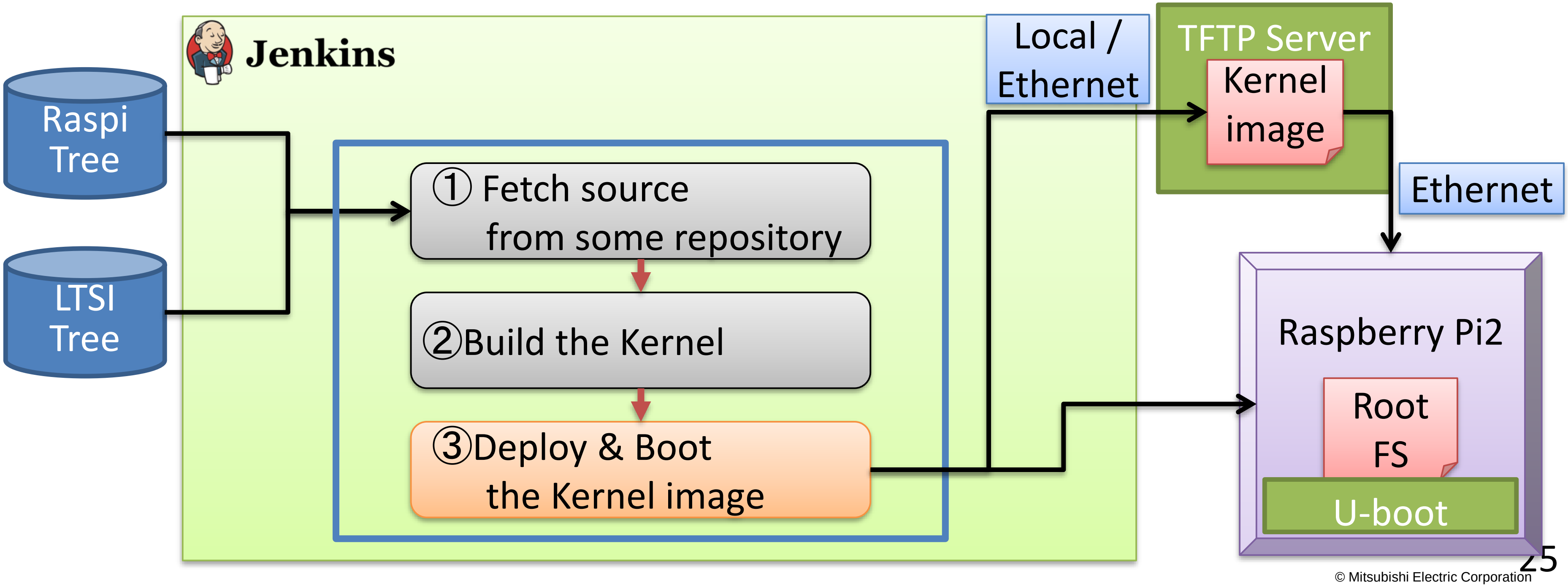
Last Successful Artifacts

ltsi_bzImage-v4.1.21	3.87 MB	view
ltsi_src-v4.1.21.tar.gz	1.55 GB	view

<Artifacts>
Ltsi_bzImage-v4.1.21: the Kernel Image
Ltsi_src-v4.1.21 :Kernel source applied patches

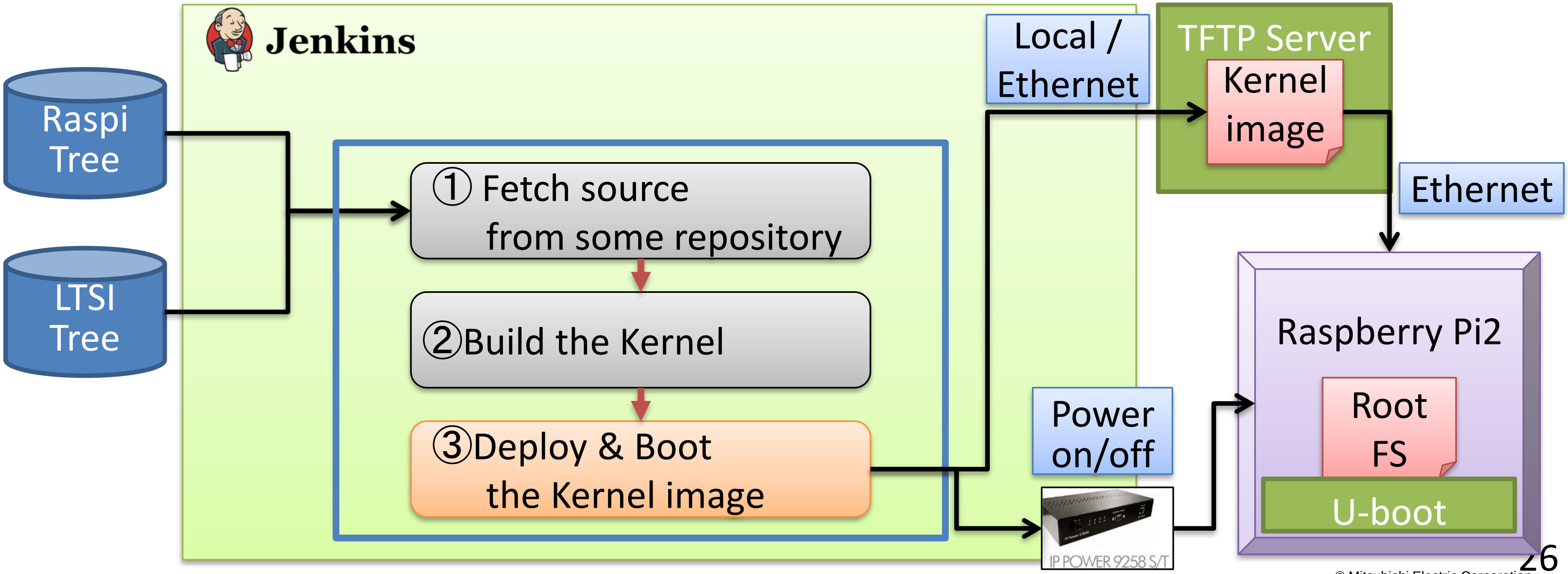
③ Deploy & Boot the kernel

- Need the below preparation for booting automatically
  1) U-boot for enabling Tftpboot on target
  2) Device Tree Binary for booting target if needed like arm, ppc etc
  3) RootFS for booting target if needed (Creating by Yocto)
  4) TFTP Server and NFS Server for booting target remotely



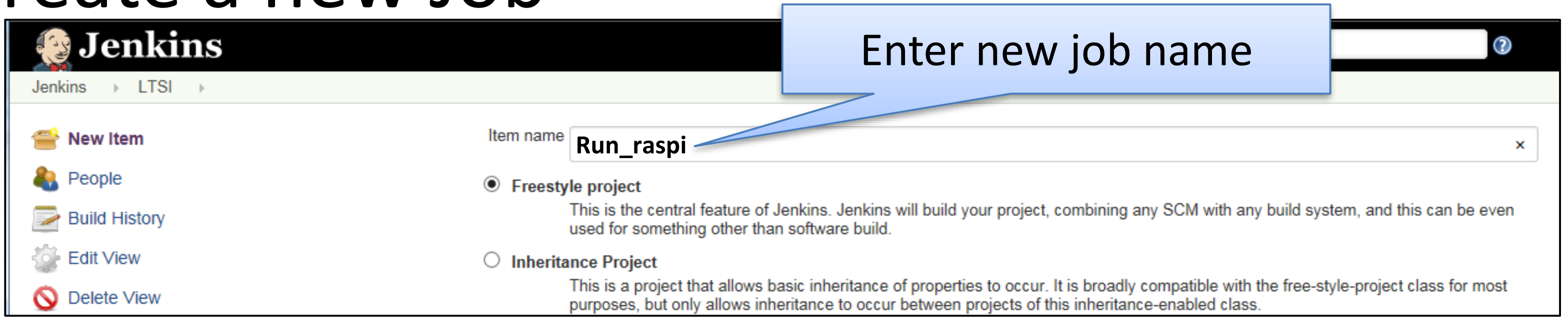
③ Deploy & Boot (cont.)

- Deploy the kernel Image
  - Copy the kernel image from the artifacts
- Boot the Linux
  - Reset a target by remote power supply
  - Boot automatically by Tftpboot of u-boot



③ Deploy & Boot (cont.)

- Create a new Job



Enter new job name

Item name: Run_raspi

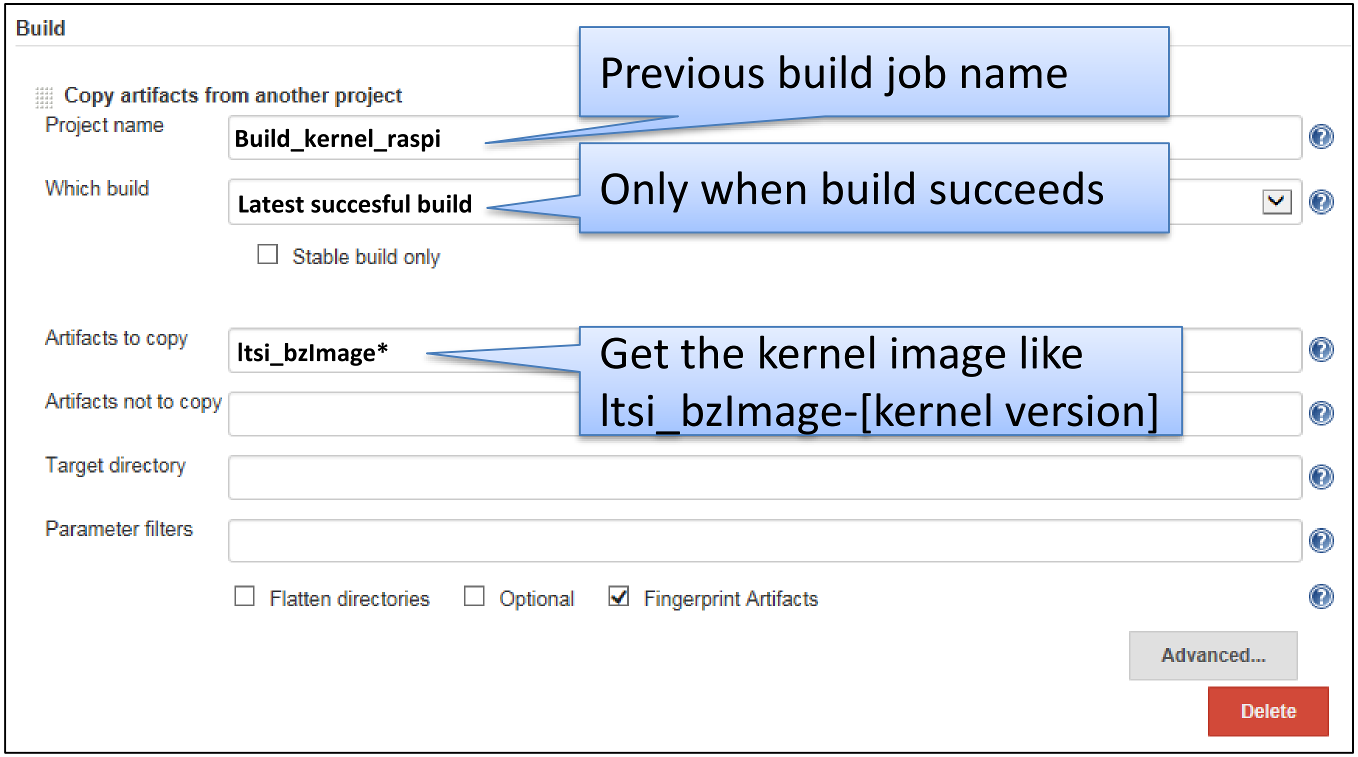
● Freestyle project

This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

○ Inheritance Project

This is a project that allows basic inheritance of properties to occur. It is broadly compatible with the free-style-project class for most purposes, but only allows inheritance to occur between projects of this inheritance-enabled class.

- Copy the artifact from build job to current job WS



Build

Copy artifacts from another project

Project name: Build_kernel_raspi

Previous build job name

Which build: Latest succesful build

Only when build succeeds

□ Stable build only

Artifacts to copy: ltsi_bzImage*

Get the kernel image like ltsi_bzImage-[kernel version]

Artifacts not to copy:

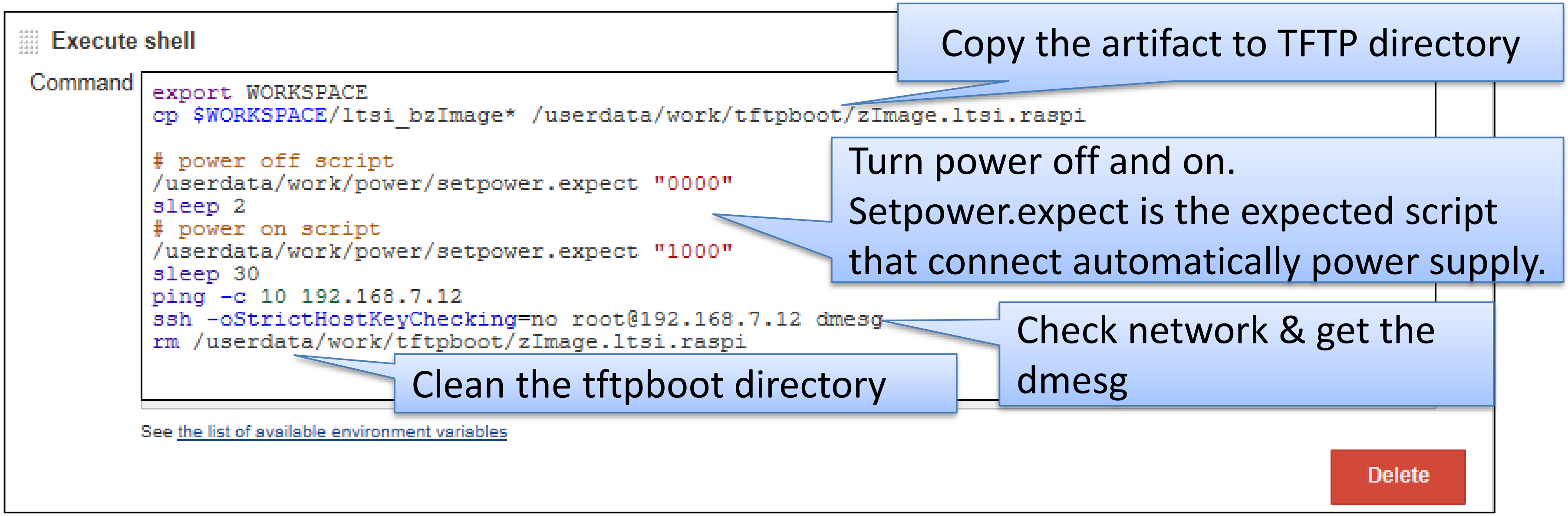
Target directory:

Parameter filters:

□ Flatten directories □ Optional ☑ Fingerprint Artifacts

③ Deploy & Boot (cont.)

- Run boot Shell script
  - 1) Copy the artifact to TFTP directory
  - 2) Turn power off & on with sleep by remote power supply
    - Using telnet with expect command
  - 3) Checking boot (check ping and get dmesg log)



Execute shell

Command

```
export WORKSPACE
cp $WORKSPACE/ltsi_bzImage* /userdata/work/tftpboot/zImage.ltsi.raspi

# power off script
/userdata/work/power/setpower.expect "0000"
sleep 2
# power on script
/userdata/work/power/setpower.expect "1000"
sleep 30
ping -c 10 192.168.7.12
ssh -oStrictHostKeyChecking=no root@192.168.7.12 dmesg
rm /userdata/work/tftpboot/zImage.ltsi.raspi
```

See the list of available environment variables

Copy the artifact to TFTP directory

Turn power off and on.
Setpower.expect is the expected script that connect automatically power supply.

Check network & get the dmesg

Clean the tftpboot directory

③ Result of deploy & boot (cont.)

- Console output of checking network using ping

```
+ ping -c 10 192.168.7.12
PING 192.168.7.12 (192.168.7.12): 56 data bytes
64 bytes from 192.168.7.12: icmp_seq=0 ttl=64 time=0.739 ms
64 bytes from 192.168.7.12: icmp_seq=1 ttl=64 time=0.695 ms
64 bytes from 192.168.7.12: icmp_seq=2 ttl=64 time=1.040 ms
```

Network is working!

- Console output of dmesg on the target board

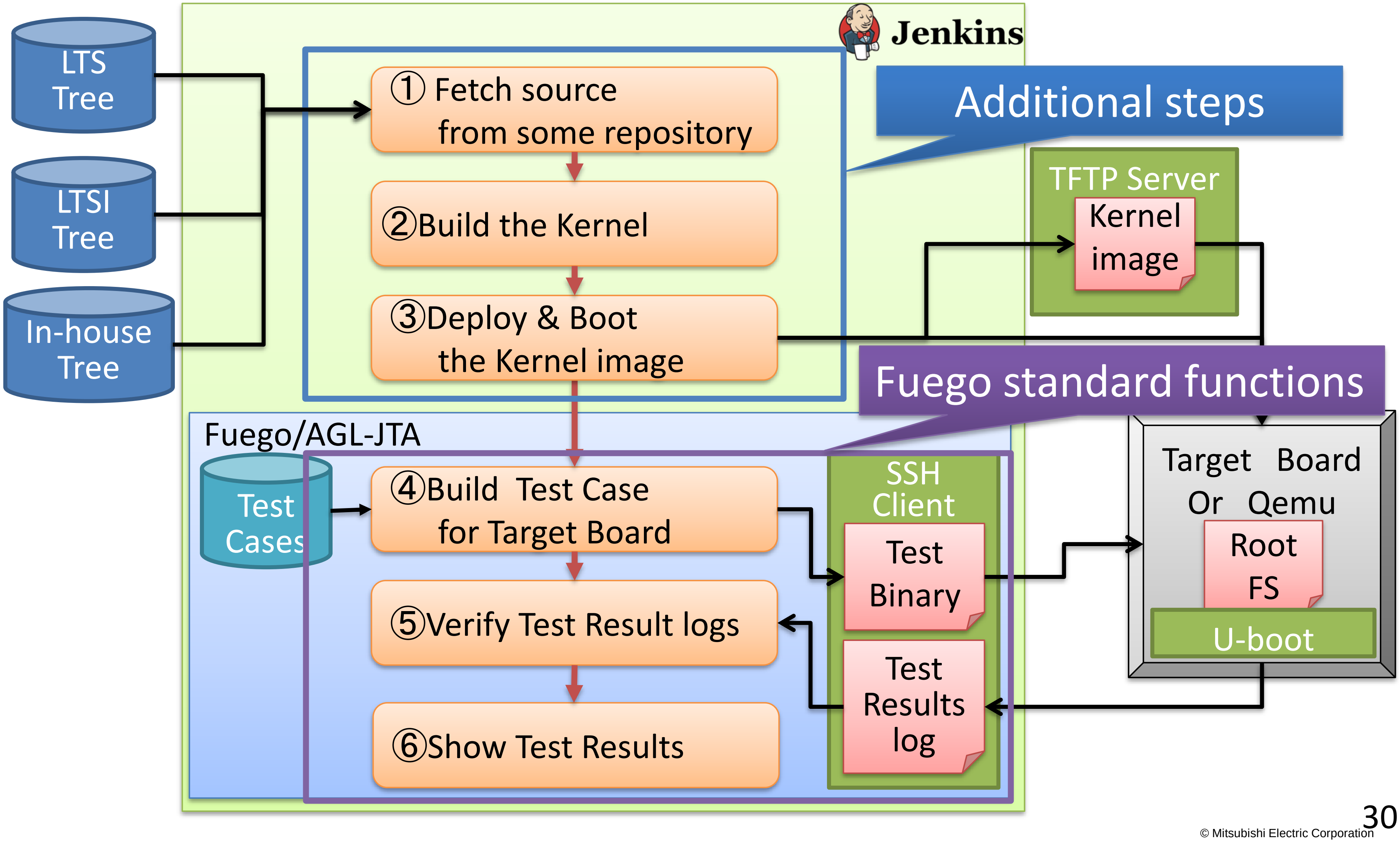
```
+ ssh -oStrictHostKeyChecking=no root@192.168.7.12 dmesg
[ 0.000000] Booting Linux on physical CPU 0xf00
[ 0.000000] Initializing cgroup subsys cpuset
[・・・・
[4.347269] IP-Config: Complete:
[ 4.350506] device=eth0, hwaddr=82:66:35:4c:16:e5, ipaddr=192.168.7.12, mask=255.255.255.0, gw=255.255.255.255
[ 4.361005] host=192.168.7.12, domain=, nis-domain=(none)
[ 4.366849] bootserver=192.168.7.3, rootserver=192.168.7.3, rootpath=
[ 4.374050] uart-pl011 3f201000.uart: no DMA platform data
[ 4.387281] VFS: Mounted root (nfs filesystem) on device 0:15.
[ 4.393682] devtmpfs: mounted
[ 4.397416] Freeing unused kernel memory: 444K (80795000 - 80804000)
[ 5.322347] random: nonblocking pool is initialized
[ 5.565450] udevd[101]: starting version 182
+ rm /userdata/work/tftpboot/zImage.ltsi.raspi [WS-CLEANUP] Deleting project workspace...
[WS-CLEANUP] done
Finished: SUCCESS
```

Get dmesg with ssh

Complete this Job!

Set up is done! Let's try to run tests!

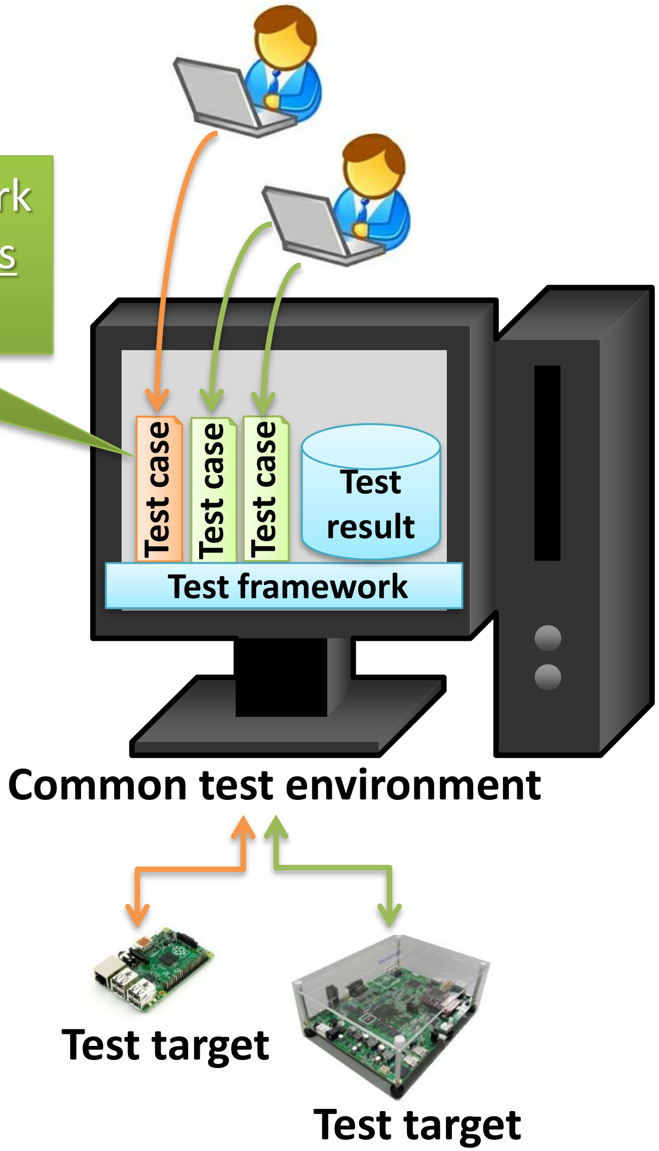
- Created additional steps using Jenkins



Utilizing OSS test suite

- How to utilize OSS test suite?

Creating all test cases is tough work
→ Utilize OSS test suite as much as possible



About OSS test suite

- Waste much time for creating test cases sometimes but, there are many OSS test suite for testing Linux kernel
- Because OSS test suite could be created for specific target or condition, some test case cannot be passed on your test target
- But checking all test case of OSS test suite is tough work also…

Share how to use OSS test suite easily,
using LTP as example

How to use OSS test suite

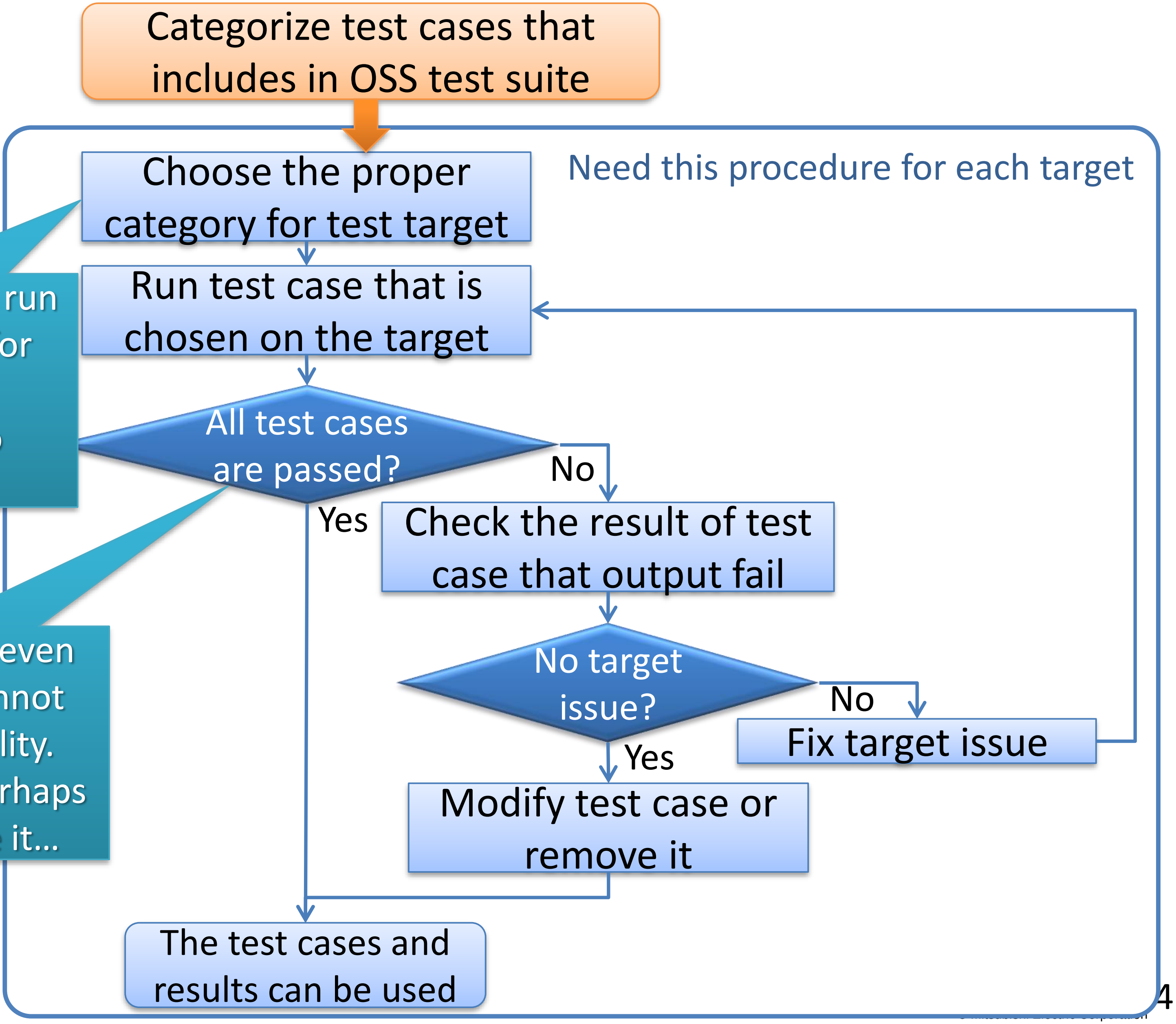
- When running OSS test suite on your target
  - The first time
    - You need to choose test case that can be used for your target

    <u>→ Share how to categorize test case effectively, in case of using OSS test suite</u>

  - From the second time
    - You need to check if the result is acceptable or need further investigation

    <u>→ Share how to check test result effectively</u>

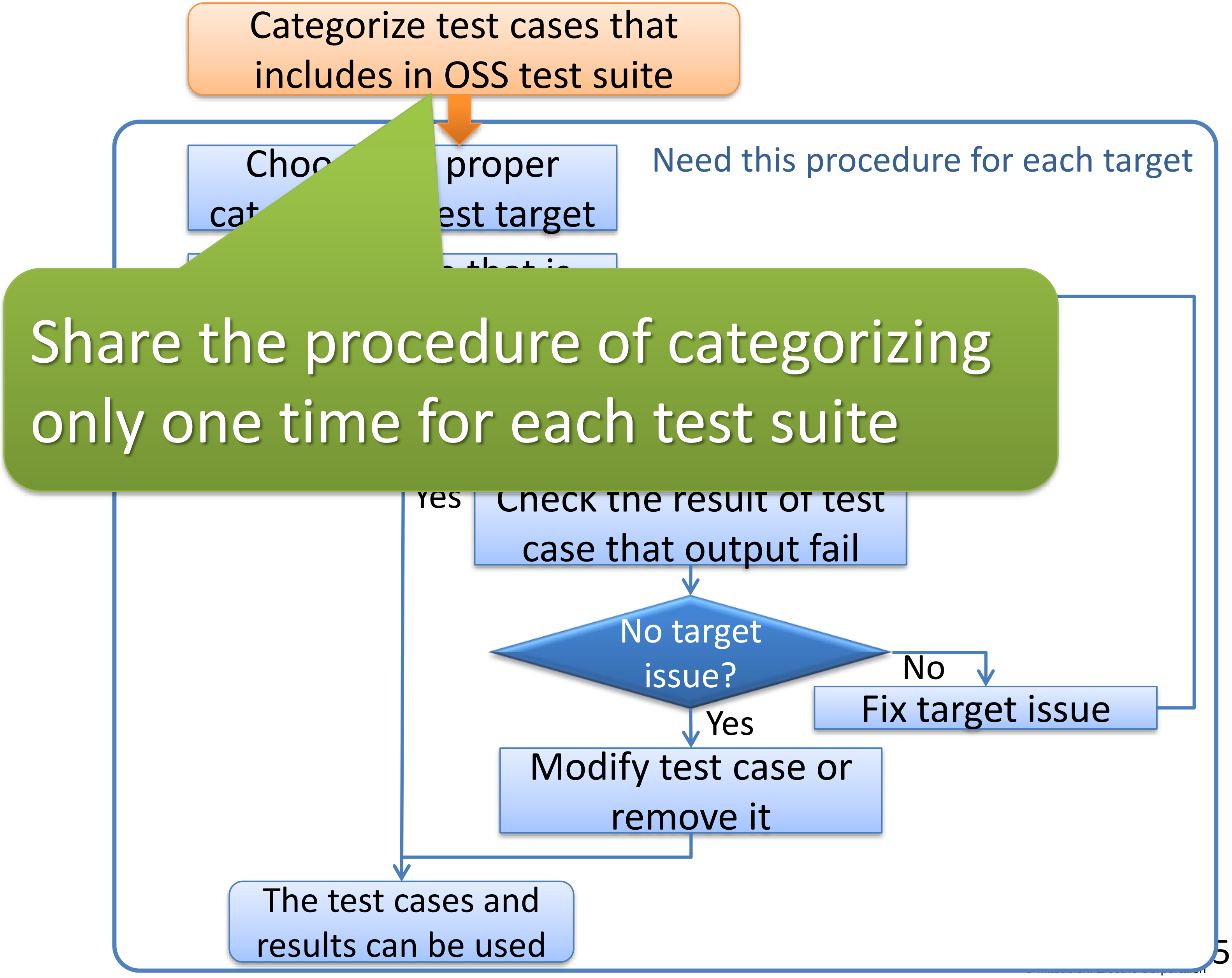
Procedure for the first time


Categorize test cases that includes in OSS test suite
Need this procedure for each target
Choose the proper category for test target
Run test case that is chosen on the target
All test cases are passed?
No
Yes
Check the result of test case that output fail
No target issue?
No
Fix target issue
Yes
Modify test case or remove it
The test cases and results can be used

You need to avoid to run test case that is not for the target because tester wastes time to check the result

You need to check even pass case if you cannot trust test suite quality. But in this case, perhaps you should not use it...

Procedure for the first time



Categorize test cases that includes in OSS test suite

Need this procedure for each target

Choo proper cat est target

that is

Share the procedure of categorizing only one time for each test suite

Yes

Check the result of test case that output fail

No target issue?

No

Fix target issue

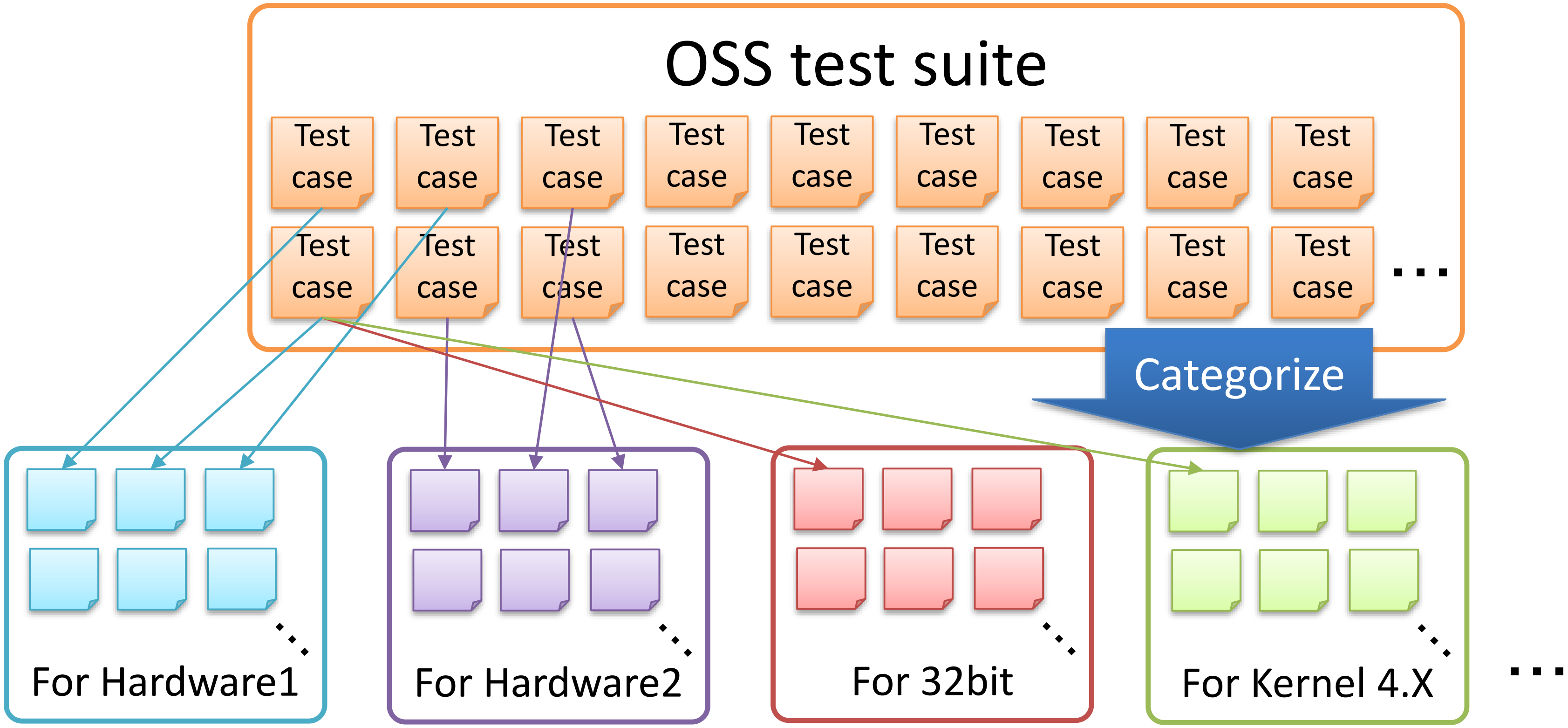
Yes

Modify test case or remove it

The test cases and results can be used

Categorize OSS test suite

- You need to choose test cases that can be used for your target from OSS test suite



Share how to categorize test cases, effectively

How to categorize test case

- Run the test suite that you would like to categorize and compare the result on many targets
- Choose targets in consideration of the below perspectives
  - Hardware difference
  - Bit architecture difference
  - Included package difference
  - Kernel difference

There could be other perspectives.

Case study : categorize LTP test cases

- In consideration of the below perspectives, run LTP and compare the results
  - Hardware difference: *Minnow board vs Raspberry Pi2*
  - Bit architecture difference: *32bit vs 64bit*
  - Included package difference: *minimal vs with GUI*
    - core-image-minimal vs core-image-sato (on Yocto Project)
  - Kernel difference: *3.18 vs 4.1*

Result summary

case	1	2	3	4	5
Hardware	Minnow board (32bit)	Minnow board (64bit)	Raspberry Pi2	Raspberry Pi2	Raspberry Pi2
Kernel	4.1.8	4.1.8	3.18.11	4.1.10	4.1.10
Userland	core-image-sato	core-image-sato	core-image-sato	core-image-sato	core-image-minimal
TPASS	938	868	934	934	933
TWARN	3	3	0	0	0
TCONF	64	134	70	70	70
TFAIL	3	3	3	3	3
TBROK	54	54	55	55	56

- TPASS - Indicates that the test case had the expected result and passed
- TWARN - Indicates that the test case experienced an unexpected or undesirable event that should not affect the test itself such as being unable to cleanup resources after the test finished.
- TCONF - Indicates that the test case was not written to run on the current hardware or software configuration such as machine type, or, kernel version.
- TFAIL - Indicates that the test case had an unexpected result and failed.
- TBROK - Indicates that the remaining test cases are broken and will not execute correctly, because some precondition not met, such as a resource not being available.

Check TWARN/TFAIL

case	1	2	3	4	5
Hardware	Minnow board (32bit)	Minnow board (64bit)	Raspberry Pi2	Raspberry Pi2	Raspberry Pi2
Kernel	4.1.8	4.1.8	3.18.11	4.1.10	4.1.10
Userland	core-image-sato	core-image-sato	core-image-sato	core-image-sato	core-image-minimal
TPASS	938	868	934	934	933
TWARN	3	3	0	0	0
TCONF	64	134	70	70	70
TFAIL	3	3	3	3	3
TBROK	54	54	55	55	56

- TWARN 3 items: Occurred on Minnow board only.
- TFAIL 3 items: The results of all cases are same. There might be no dependency.

Check TBROK

case	1	2	3	4	5
Hardware	Minnow board (32bit)	Minnow board (64bit)	Raspberry Pi2	Raspberry Pi2	Raspberry Pi2
Kernel	4.1.8	4.1.8	3.18.11	4.1.10	4.1.10
Userland	core-image-sato	core-image-sato	core-image-sato	core-image-sato	core-image-minimal
TPASS	938	868	934	934	933
TWARN	3	3	0	0	0
TCONF	64	134	70	70	70
TFAIL	3	3	3	3	3
TBROK	54	54	55	55	56

- The results of each cases are same, excepting the below.
  - 1 item: NOT occurred on Minnow board (32bit).
  - 1 item: Occurred on Raspberry Pi2 only.
  - 1 item: Occurred on core-image-minimal only.

Check TCONF

case	1	2	3	4	5
Hardware	Minnow board (32bit)	Minnow board (64bit)	Raspberry Pi2	Raspberry Pi2	Raspberry Pi2
Kernel	4.1.8	4.1.8	3.18.11	4.1.10	4.1.10
Userland	core-image-sato	core-image-sato	core-image-sato	core-image-sato	core-image-minimal
TPASS	938	868	934	934	933
TWARN	3	3	0	0	0
TCONF	64	134	70	70	70
TFAIL	3	3	3	3	3
TBROK	54	54	55	55	56

- The results of each cases are same, excepting the below.
  - 10 items: NOT occurred on Minnow board (32bit).
  - 1 item: Occurred on Raspberry Pi2 only.
  - 2 items: Occurred on Minnow board only.
  - 66 items: Occurred on Minnow board (64bit) only.
  - 3 items: NOT occurred on Minnow board (64bit).

The details of test case

- The below test cases could be depending on Hardware. (7items)
    - Raspberry Pi2 only
        - clock_getres01 (TCONF)
        - getrusage04 (TBROK)
    - Minnow board only
        - fanotify05, fanotify06 (TCONF)
        - Fanotify01, fanotify02, fanotify04 (TWARN)

- The below test cases could be depending on bit architecture. (69items)
    - Minnow 64bit only
        - bdflush01, chown01_16, chown02_16, chown03_16, chown05_16, fchown01_16, fchown02_16, fchown03_16, fchown05_16, fstatat01, fstatat01_64, getegid01_16, getegid02_16, geteuid01_16, geteuid02_16, getgid01_16, getgid03_16, getgroups01_16, getgroups03_16, getuid01_16, getuid03_16, lchown01_16, lchown02_16, modify_ldt01, modify_ldt02, modify_ldt03, setfsgid01_16, setfsgid02_16, setfsgid03_16, setfsuid01_16, setfsuid02_16, setfsuid03_16, setfsuid04_16, setgid01_16, setgid02_16, setgid03_16, setgroups01_16, setgroups02_16, setgroups03_16, setgroups04_16, setregid01_16, setregid03_16, setregid04_16, setresgid01_16, setresgid02_16, setresgid03_16, setresgid04_16, setresuid01_16, setresuid02_16, setresuid03_16, setresuid04_16, setresuid05_16, setreuid01_16, setreuid02_16, setreuid03_16, setreuid04_16, setreuid05_16, setreuid06_16, setreuid07_16, setuid01_16, setuid02_16, setuid03_16, setuid04_16 (TCONF)
    - Other than Minnow 64bit
        - fork14, getcpu01, mmap15 (TCONF)

- The below test cases could be depending on User land. (1item)
    - core-image-minimal only.
        - Utimensat01 (TBROK)

- The below test cases could be depending on Minnow 32bit. (11items)
    - Other than Minnow 32bit
        - eventfd01, io_cancel01, io_destroy01, io_getevents01, io_setup01, io_submit01, readdir21, sgetmask01, set_thread_area01, ssetmask01 (TCONF)
        - syslog08 (TBROK)

# There is no items that depends on Kernel version.

The details of test case

- The below test cases could be depending on Hardware. (7items)
  - Raspberry Pi2 only
    - clock_getres01 (TCONF)
    - getrusage04 (TBROK)

Categorized test case for:

- Hardware: ARM or Intel or Both
- Bit architecture: 32bit or 64bit or Both
- Included package (Userland)
- Kernel Version

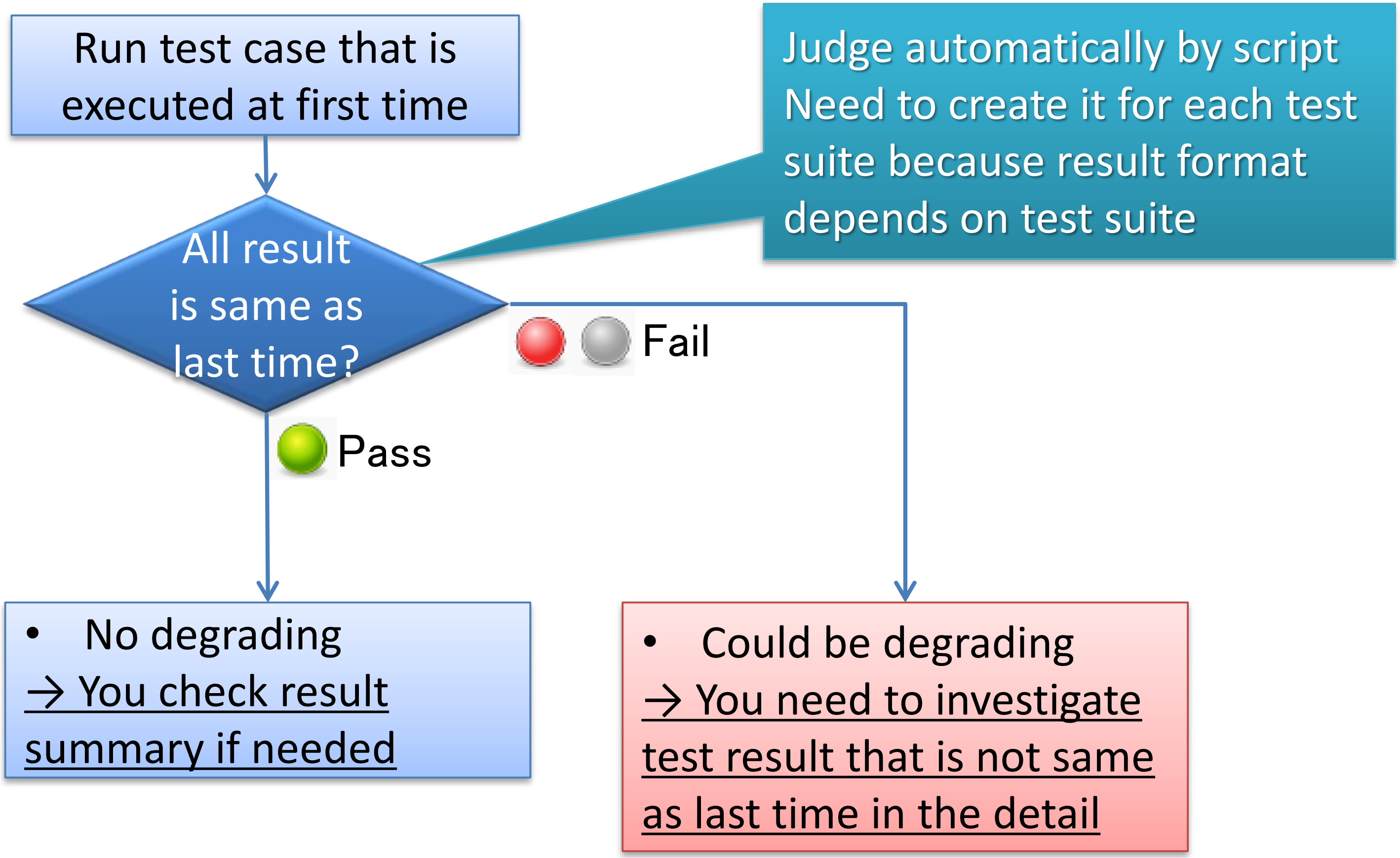
<u>You can choose test cases from the category that suits each target specification</u>

    - syslog08 (TBROK)

# There is no items that depends on Kernel.

Procedure from the second time

- Comparing with the result of last time, you can easily check if there is degrading or not



Run test case that is executed at first time

All result is same as last time?

Judge automatically by script
Need to create it for each test suite because result format depends on test suite

● ● Fail

● Pass

- No degrading
→ You check result summary if needed

- Could be degrading
→ You need to investigate test result that is not same as last time in the detail

Conclusion

- Summary
  - Test framework like Fuego can be utilized for the development using Linux
  - When using Fuego with customization, automated test can be triggered by software update
  - Categorizing test cases and comparing test results can ease using OSS test suite such as LTP

- Future Works
  - Create automated test environment that is triggered by software update using QEMU
  - Consider the way to compare test results with those of last time easily
    - Dependency of result format should be decreased

Reference

- LTSI project :
  - http://ltsi.linuxfoundation.org/
- LTSI Test project:
  - http://ltsi.linuxfoundation.org/ltsi-test-project
  - Test Framework(Fuego):
    - https://bitbucket.org/cogentembedded/jta-public.git
- AGL Test framework(AGL-JTA) :
  - https://wiki.automotivelinux.org/agl-jta
- Linux Test Project
  - http://linux-test-project.github.io/
- Introduction to the Fuego test system By Tim Bird
  - http://events.linuxfoundation.org/sites/events/files/slides/Introduction-to-Fuego.pdf
- Unveil How to Customize LTSI Test For Your Platform
  - http://events.linuxfoundation.org/sites/events/files/slides/ELCE2015-LTSI_Test_Project.pdf

Thank you!!

Questions?