Marcus Wall

From: Marcus Wall
Sent: Tuesday, November 29, 2016 2:20 PM
To: Marcus Wall
Subject: New Member intro WIKI

From: Marcus Wall
Sent: Thursday, November 10, 2016 3:59 PM
To: Marcus Wall <marcus.wall@ericsson.com>
Subject: EE intro Monday, November 07, 2016

EE CI Intro

General

The EE (Execution Environment) is basically a Windriver Linux distribution with /// specifics for our radio base station DU digital units.
The product is very large in terms of MB/lines of code, but has a large 3pp part. The size is particular visible in long build times.
The customer is the G2 projects developing the MSRBS&TCU.

~3 sections of developers including a test team responsible for test scope/strategy & test result follow up.
Test team also is responsible for lab test resources. Test team & Jante CI team works close together and test team also contributes to CI Jenkins repo. Test team & Jante team has a shared designer "API" **jante@mailman.lmera.ericsson.se**

Our CI team is called Jante, it is not so big in number of persons ~3-4
Jante has focus on flow development & the development/setup/availability of the CI tools/infrastructure (see respective heading)
The main CI frontend is our production Jenkins, but we also have a team Wiki:
https://wiki.lmera.ericsson.se/wiki/PLF_CI/Jante
The CI machine is semi-mature in sense that main structure is stable but development is performed i.e
new flows / new external interfaces / capacity increase etc.
Jenkins is used as the "flow execution manager" and uses dedicated functional user **rcsci2**.

EE/Jante is mainly Kista based so even though the CI service as such is 24/7 the real action is working ours CET.

WoW

- 3 week sprints
- Sprint starts/ends @ a regular meeting with stakeholders to summarize passed sprint and look over prio for new sprint.
- prioritized Backlog items in mail MoM from sprint meeting above + openAlm.
- Code review in Gerrit highly recommended but not forced (+2 can be given by commiter)
- Personal workspace on User@esekilxxen2145/repo/
- Manual pre-deliver tests performed in sandbox Jenkins (fem010) with external interfaces stubbed out in job code.

Code base/development

EE - Pre 16b
Main customer repository, where **designer** works, is the rcs-ee in gerritforge (it has a number of **sub-modules**)
https://gerritforge.lmera.ericsson.se/gerrit/#/admin/projects/rcs-ee
rcs-ee also includes tests, that post 16a was broken out to a repo of its own.
gerritforge will be terminated and repos moved to Gerrit central before end of -16

EE – 16b onwards
Main customer repository is the rcs-yocto (it has a number of **sub-repos**)
https://gerrit.ericsson.se/#/admin/projects/rbs_platform/rcs/rcs-yocto
This is where the **designers** work.

And this is the test repo, mainly black box test controlled by **test team**:
https://gerrit.ericsson.se/#/admin/projects/rbs_platform/rcs/rcs-yocto/rcs-test

CI repo
This is "our" CI repo, it is one-track "master" as opposed to design & test that is branched for each ///project (16b, 17a etc)
https://gerrit.ericsson.se/#/admin/projects/platform/jenkins-conf/fem005-eiffel002
CI code is mainly written in bash & groovy.
Central is the jobdsl plugin that is used to generate Jenkins job xmls from high level groovy code.
Central is also the buildflow plugin used to write flow orchestration jobs.

Products

RCS do change/step their product numbers every project but only when function or structure is altered to motivate this. (The use of wideband verification R-states makes it possible to keeps same product for a long time.)

Payload

One load module, CXC, per target: tcu/dusX2/dusX3(dusx3 17a onwards)
i.e

https://fem005-eiffel002.rnd.ki.sw.ericsson.se:8443/jenkins/job/ee-yocto-delivery-master-build/lastSuccessfulBuild/artifact/tcu-delivery/

A number of interface containers, CXAs:
https://fem005-eiffel002.rnd.ki.sw.ericsson.se:8443/jenkins/job/ee-yocto-delivery-master-build/lastSuccessfulBuild/artifact/out_arch/

Build

One Jenkins build job shared by all flows
default is build "all" targets = dusx2, dusx3, tcu, qemux86, nl, nl-cpm2
4 Ups produced small/big x dus/tcu
SSTATE buffers providing build avoidance is kept to speed up build.
Invoked in the yocto build a signing of the Linux is performed.
The signing service is in Finland (LMF) eppki@ericsson.com

The three steps in Jenkins build job:
1 Rcs-yocto build of all targets (using Jenkins_build all) CXCs and CXAs
2 produce black EE-CXPs, take EE-cxp on baseline and swap CXC for the new one from [step1]
3 Produce UPs same as above, swap old CXP for new one from[step 2]

Packaging/delivery

Actual delivery is a Jenkins job exporting EE payload to MiddleWare in clearcase.

All final packaging and revision handling is done outside our machine as we deliver downstream to MiddleWare.
Our CXCs are package into a CXP together with another CXC, Net loader. NetLoader is outside EE src baseline.

Pre 17a packaging
Tcu and dusX2 is put in a EE CXP that in turn is put in another bundle CXP together with MiddleWare CXP.
So we get a nested CXP. This bundle CXP is then delivered to Node.
In other words MiddleWare & EE is delivered together pre 17a to Node level.

17a onwards packaging
The EE CXPs are delivered directly to Node, however still done by MiddleWare.
The DusX2 and TCU are still in a wrapper CXP but not in a bundle with MW CXPs.
The DusX3 CXP don't have a wrapper CXP.

Flows

We have a number of flows but the central one is the **deliveryFlow.**
Most flows are instanced by project track (Jenkins job name includes name if track)
but often contains project independent build steps.
https://wiki.lmera.ericsson.se/wiki/PLF_CI/Jante/FlowOverview

ReviewFlow

Triggers on magic word "testme"

Builds everything and performs a slim installation on all main targets.
Awards +1 if successful.
i.e for master branch:
https://fem005-eiffel002.rnd.ki.sw.ericsson.se:8443/jenkins/job/ee-yocto-review-master-reviewFlow/

DeliveryFlow

Triggers on magic word “respect” as gerrit comment (if +1+2 present for patchset)
Builds everything and performs full test scope.
Merges contribution to remote if successful. In other words input to delivertFlow is a gerrit patchset, output is a new remote HEAD.
Delivers by importing payload to MiddleWare clearcase if successful.

Other Flows

Contflow – runs whenever no design contributions are queded. Provided stats on system/testcase stability.
Intel/RTE prewash – prewash of 3pp, under development to become a part of normal CD.
RCSEE-FW – New flow under delivery, purpose is to deliver the UBOOT directly to Node.

FEM flow

Our own flow for the fem code.
Flow is executed in sandbox, trigger on magic word “respect”.
No automatic testing as the EE product flows
Three jobs involved,
https://fem010-eiffel002.rnd.ki.sw.ericsson.se:8443/jenkins/view/jobdsl

jobdsl-review , sets reviewers to all members of team.
jobdsl-delivery-trigger, sanity check of patchset
jobdsl-delivery, actual sumbit

Specific Flow “Sugar” functionality

Build reuse – Before building check if successful build of same SHA already exists. Typically between review & deliveryFlow reuses.

SpecFlow – To save time while waiting in que start building on the possible outcomes of the one ahead of you. (new or same remote HEAD)

STP Health check - After an STP was used in testrun, check it before returning to pool (status_dirty -> status_verified)

Fpga baseline CL0 -> early test of new fpga versions, only prewash ran at night.

Designer test – job where designers can run test from test scope on own HW.

Test case white filter – Holds known and excepted failures that should fail test run.

External baseline handling – Lift the external baseline as a single delta, tried every night.

Tools

Jenkins
Git/gerrit
IB/MIA
ARM Nexus
Insight
Clearcase

Infra Resources

STPs
All test are performed on real target HW. Delivery testFlow runs ~20-30 parallel target test.
Capacity is/should be about two parallel testFlow (they are however never run in parallel)
All STPs are tied to a Jenkins slave and scheduled by slave tags.

VDIs
The stp slaves are executed on one of the generic VDIs
Named Linux-XXX in computer page:
https://fem005-eiffel002.rnd.ki.sw.ericsson.se:8443/jenkins/computer/

Build machines
Three 40core power machines with two build slots each:
https://fem005-eiffel002.rnd.ki.sw.ericsson.se:8443/jenkins/label/YoctoBuild/