Technical Showcase

CE Workgroup Linux Foundation / Embedded Linux Conference Europe

Using docker and yocto as a base for multi-arch fleet deployment

--- Andrei Gherzan – *resin.io* ---

What is demonstrated

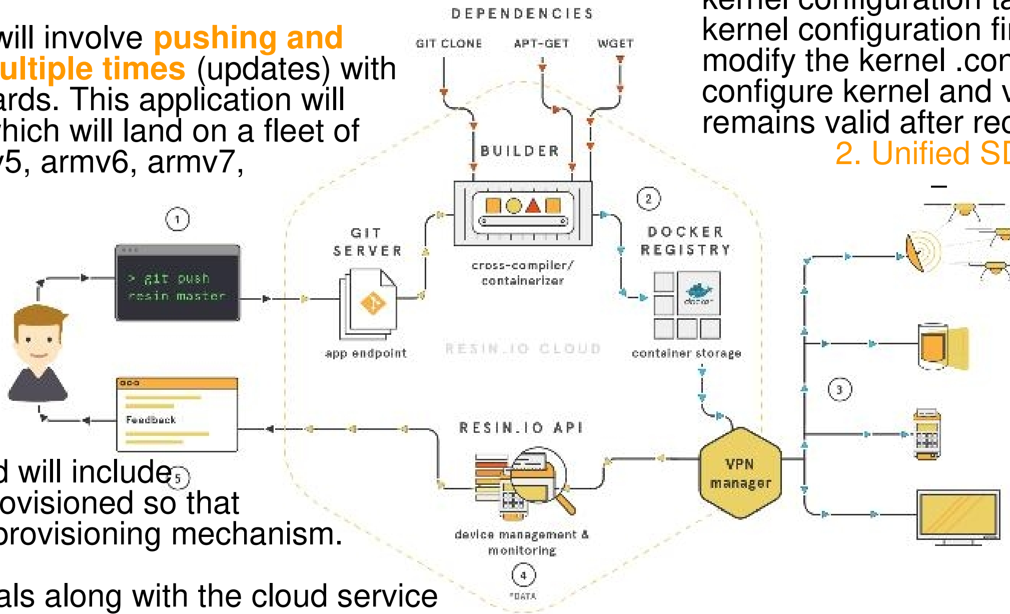
Demonstrate how embedded software can be managed throughout it’s **entire lifecycle from provisioning a device to application deployment** in an easy and automated way.

The practical demonstration will involve **pushing and deploying an application multiple times** (updates) with minor changes to various boards. This application will be a single code repository which will land on a fleet of devices of varying arch (armv5, armv6, armv7, x86_32, x86_64).

Users will be able to get a feeling of the deployment process by **modifying the application and pushing updates to all the devices**.

The entire setup will be made out of **24+ devices** and will include some boards that won’t be provisioned so that users will be able to test the provisioning mechanism.

All the equipment and materials along with the cloud service hosting are provided by resin.io. The devices software stack is **yocto based** and the custom layers for customization, support and improvements are **publicly available**.



```
> git push resin master
```

Hardware Information

Raspberry Pi, Raspberry Pi 2, BeagleBone Black, Intel Edison, Intel NUC and maybe others.

What was improved

1. Unified Linux Kernel configuration
   - defines a couple of tasks which are hooked into the kernel configuration task chain as it follows: after kernel configuration finishes (kernel.bbclass), we modify the kernel .config file generated, re-configure kernel and verify that our configuration remains valid after reconfiguration.
2. Unified SD card images
   - SD card generation bbclass which, based on different variables, is instructed to copy different files from deploy directory to SD card and come with a valid machine specific SD image. Our images inherit this bbclass so each machine bbappends the images only with their specific configuration. Ex: kernel image names, bootloader deployment locations etc.
3. Other customizations and support packages available at meta-resin.

Source code or detail technical information availability

https://github.com/resin-io/meta-resin
https://resin.io/blog/charting-an-open-source-path-for-resin-io/