OPEN. FOR BUSINESS.

OCP SUMMIT

Open
Firmware

Linuxboot continuous integration

Jean-Marie Verdun/ITRenew

Guillaume Giamarchi/ITRenew

OPEN. FOR BUSINESS

OCP SUMMIT

Introduction

This is Work In Progress, feel free to collaborate

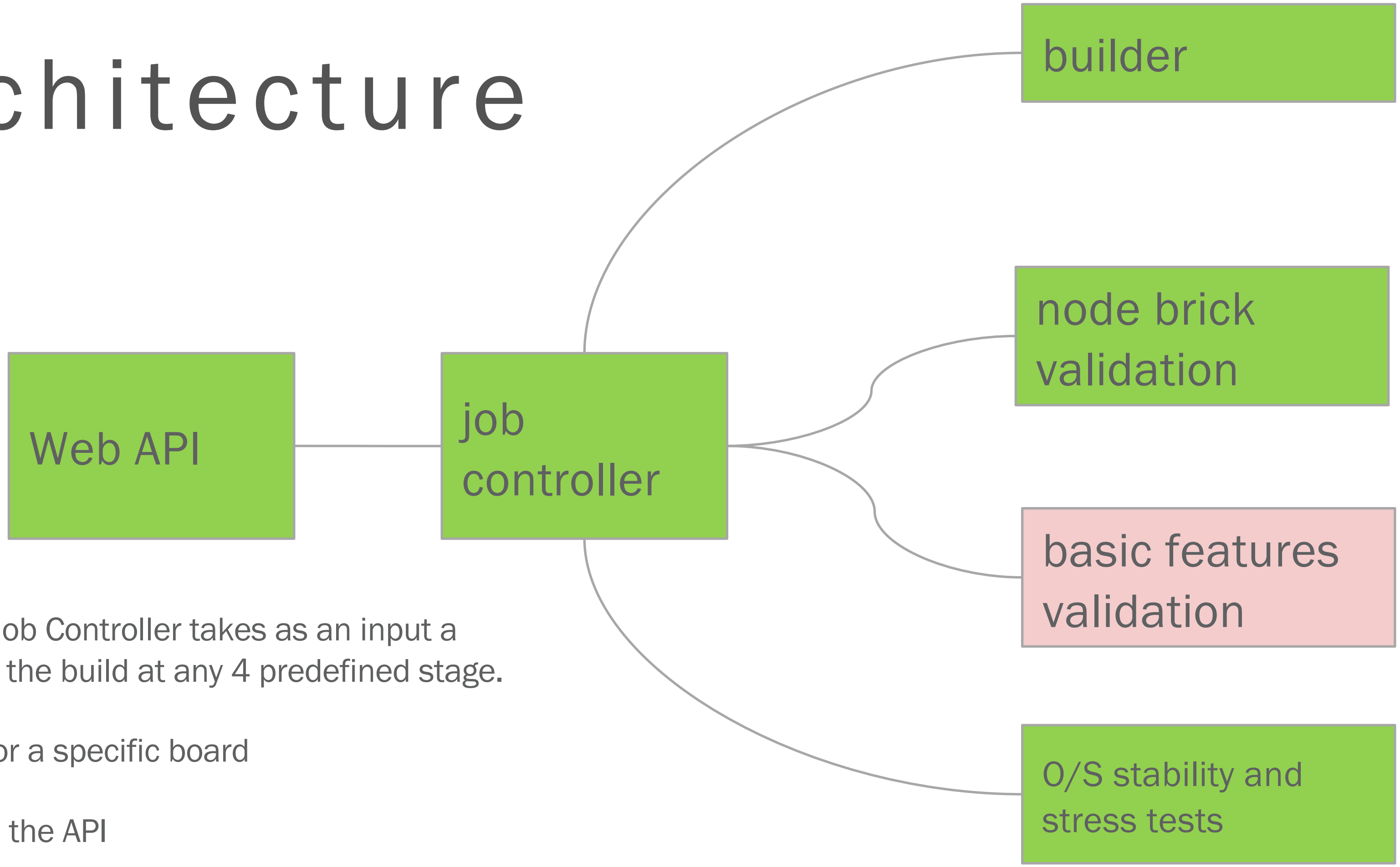
Linuxboot Continuous Integration platform aims to provide

1. Replicable build environment for linuxboot images
2. Fully automated testings at firmware level on real hardware
3. Multiple hardware generation support (currently focusing on OCP nodes)

Architecture

Active dev

Future dev



CI must be used from a Web API. Job Controller takes as an input a job description file which can stop the build at any 4 predefined stage.

Output is a status and a rom file for a specific board

Job status can be queried through the API

Implementation

1. Automatic redeployment managed through Ansible scripting
2. Automatic redundancy and scalability based on hardware availability
3. Slurm batch scheduler to manage run queue, unique job ID, and jobs output
4. KVM used to sand box builds into Ubuntu Xenial VMs
5. Basic setup requires:
   a. An ansible master node
   b. A Slurm controller node
   c. A Slurm batch node
   d. All of them sharing the same subnet

Web API

Goals

1. Get as an input a github repository address with a unique commit ID
2. Provide job control
   a. Launch
   b. Kill
   c. List
3. Provide jobs status feedback
   a. Build log file

Initial implementation

Written in Go

Support

- Job launch
- Job status query
- Job log

Job controller - Goals

1. Allocate and manage build nodes resources
2. Preset build environment
3. Store jobs status
4. Controlled using a hidden file which contains jobs description that will override default values

Job controller

1. Allocate job through Slurm batch scheduler
2. Setup a virtual machine (based on Ubuntu Xenial) when node is allocated with predefined characteristics using KVM and virsh
   a. VM storage is seating in memory (about 40 GB)
   b. VM have access to the Internet and can run apt command to setup build environment
3. Setup remote access to the VM from the slurm compute node
4. Copy the relevant files into the VM and setup build environment
5. Initiate job execution

Builder

Goals

1. Build a fully functional linuxboot image based on job parameters

Initial implementation

1. Based on osresearch/heads build environment
   a. Requires initial board ROM
   b. Can apply various patches to the kernel
   c. Can build NERF (go based) user environment
   d. Can control/extract final DXE drivers integrated within linuxboot ROM

node brick validation

Goals

1. Validate that a newly built image doesn't brick a node (aka that we can talk to the firmware through serial and successfully execute basic command)

Initial implementation

1. Based on initial qemu launch of the ROM
2. Based on real hardware setup with a ROM emulator connected to the board

basic features validation

1. Validate that a newly built image is properly detecting hardware
2. Validate that a newly built image is able to install an O/S and boot it through
   a. PXEboot or any other network boot capability
   b. local boot on AHCI and NVMe storage

basic features validation

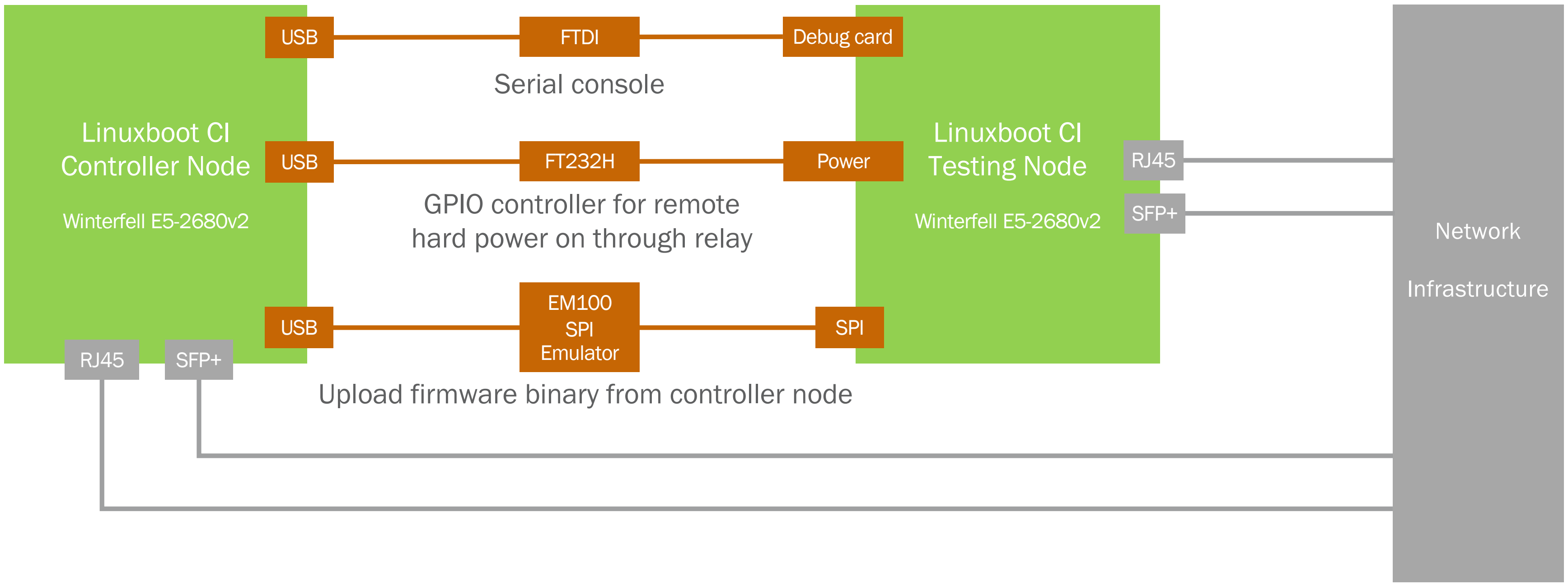
1. O/S image installer built for linuxboot
   a. Ubuntu Xenial netinstaller or local boot kernel fails to boot properly on linuxboot (the kernel hangs)
      i. We regenerate a full ISO image based a valid original Ubuntu Xenial ISO and disable EFI support within the installer kernel
      ii. That image is also pre-configuring console output either to ttyS0 (local serial) or ttyS4 (SoL) on Winterfell machine
      iii. That new image is also bootable through a PXEboot process which is automatically configured on the slurm controller node.

O/S stability and tests

Goals

1. Validate that a newly built image is able to:
   a. Run Linux at full operational mode
      i. Properly detect hardware behavior
      ii. Detect and can manage without error PCIe subsystems
      iii. Can run various workload without system error and within an acceptable performance goal
         1. Run the Linpack/pysthone benchmark at speed
         2. Run Networking benchmark at speed
         3. Run bonnie++ at speed

Demo



Linuxboot CI
Controller Node
Winterfell E5-2680v2
USB
FTDI
Debug card
Serial console
USB
FT232H
Power
GPIO controller for remote
hard power on through relay
USB
EM100
SPI
Emulator
SPI
Upload firmware binary from controller node
RJ45
SFP+
Linuxboot CI
Testing Node
Winterfell E5-2680v2
RJ45
SFP+
Network
Infrastructure

Scaling the public CI

Current nodes are based on Winterfell machines:

- Dual Xeon 2680v2 / 64 GB RAM / 3 TB HDD / 1Gbps network - Cost : 750 $US (with racks)
- 20 servers are allocated
  - 4 machines are used for research development activity
  - 4 machines are used for development and industrialization
  - 4 machines are used for integration testings
  - 8 machines are used for production

Scaling the public CI

Estimated CI requirements to scale the project:

- Increase servers count with various backend model
  - Keep current infrastructure for Winterfell
  - For each new generation needs at least 4 nodes
    - 1 for brick testing
    - 3 for O/S setup and workload testing
    - Add 2 build servers (Winterfell class)
  - Integrate Leopard, Yosemite, Tioga Pass
  - Upgrade winterfell node with SSD/NVMe storage ?
- Secure long term hosting - 10k$ / month per rack (connectivity / power

OCP SUMMIT

OPEN.

FOR BUSINESS.