Introducing **resinOS**

An Operating System Tailored for Containers and Built for the Embedded World

Andrei Gherzan / Petros Angelatos

October 2016

About us

Andrei Gherzan

- Lead engineer of resinOS
- Maintainer of meta-raspberrypi, meta-chip

Petros Angelatos

- Founder / CTO
- Ported Docker to ARM

resin OS

Agenda

- Mission
- History
- Architecture
- Features
- Development tools
- Future

Mission

- Be the embedded OS of choice for containers in IoT
- Create a community around containers for IoT
- Modern security features
- Minimal footprint
- Production ready

History - resin.io

- Started 4 years ago
- Modern devops practices to the embedded world
- Naturally leaned towards containers
- Ported Docker to ARMv6
- Ported Docker to ARMv5
  - Fixes upstreamed

History - resinOS

- Needed an OS for our platform
  - Tried a modified Arch
  - Tried a modified TinyCore
- Both had important shortcomings

History - resinOS

- Started in January 2014 as internal project
- Used Yocto as a base
- Open sourced in July 2015
- Currently under very active development
- It’s been running in production for 2.5 years

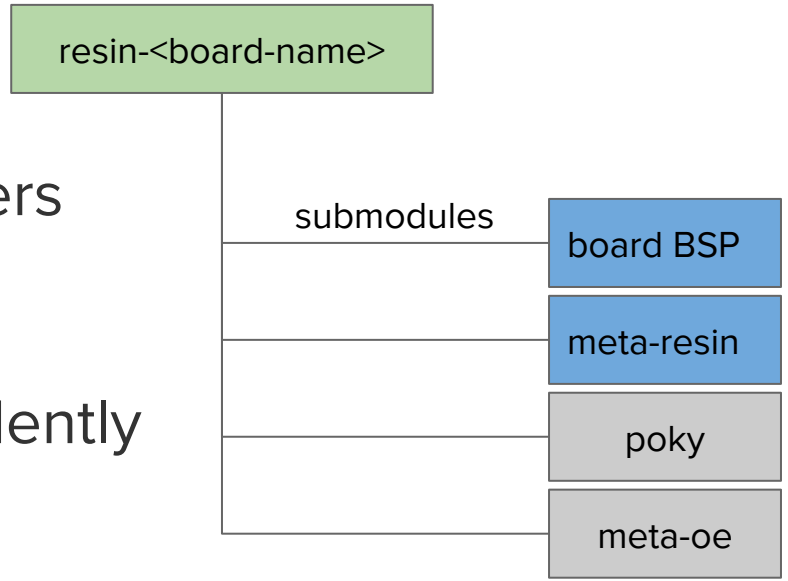
Architecture

Yocto

- Why we chose yocto
  - Minimal
  - Low footprint
  - Build system allows for easy patching
  - Board vendors usually supply Yocto BSP
    - Easier device support

Yocto layer architecture

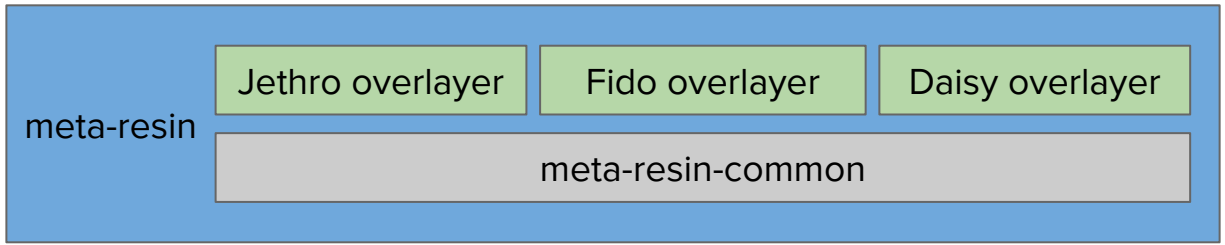
- One repo per board
- Submodules for dependent layers
  - Tried repo tool
  - Tried monolithic repo
- Each board can move independently



https://github.com/resin-os?query=resin-

meta-resin

- Main resinOS layer
- Automatic aufs patching
- BSP independent kernel configuration
- Can prepopulate docker images
- Kernel headers for out-of-tree module development



https://github.com/resin-os/meta-resin

resin OS

Build system

- Environment defined in a Dockerfile
  - Predictable host configuration
- Docker image artifacts
  - You can use the OS as a container
  - `resin/resinos:<version>-<board>`

https://github.com/resin-os/resin-yocto-scripts

resin OS

Partition layout

- Separate rootfs and root state
  - We know exactly which services write to disk
- Dual root partition
- data partition auto-expands on first boot

Read-only root

- Forced us to investigate all writes
- Configuration stored in state partition
  - Network configuration
  - Random seed
  - Clock at shutdown
- Some state is stored in tmpfs
  - DHCP leases
  - Limited logs

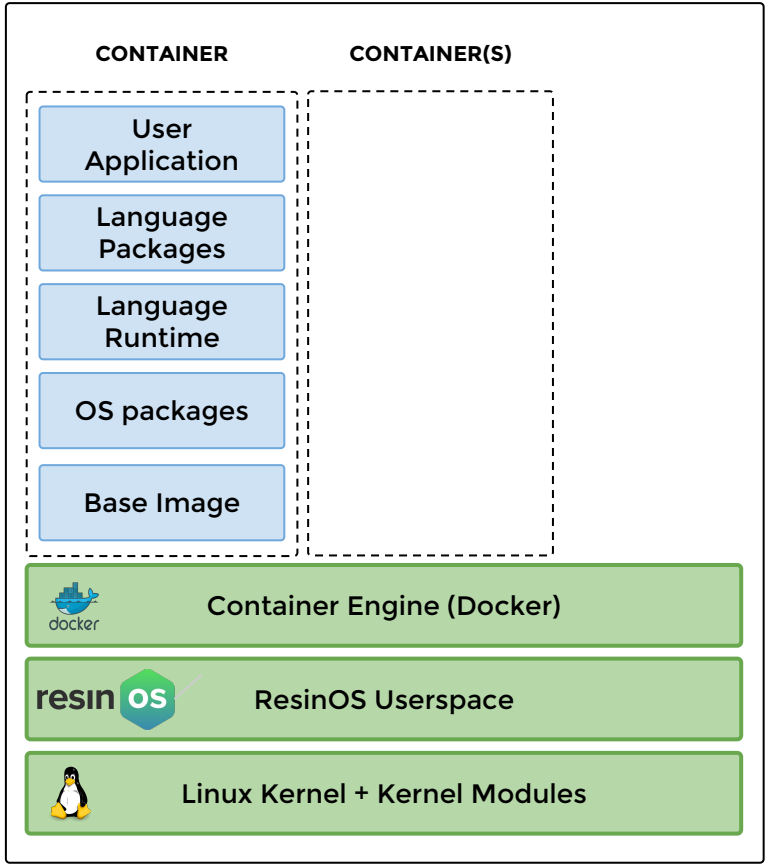
Read-only root

- Cleaner separation
- OTA updates are much easier
- Enables diff based updates
- We can’t leave state behind

Reliability

- Compartmentalisation of failures
  - Device can survive data partition corruption
  - Most I/O activity happens in there
- Root partition is never written to while in use
- We strive to do atomic operations everywhere

Runtime


CONTAINER
CONTAINER(S)
User Application
Language Packages
Language Runtime
OS packages
Base Image
docker
Container Engine (Docker)
resin OS
ResinOS Userspace
Linux Kernel + Kernel Modules

Ingredients

- Systemd
- NetworkManager
- ModemManager
- dropbear
- dnsmasq
- docker
- avahi

Systemd

- Leverage a lot of systemd features
  - Adjusting OOM score for critical services
  - Running services in separate mount namespaces
  - Very easy dependency management
  - NTP
- Socket activation for SSH
  - Saves RAM since ssh is running only when needed

Networking

- DNS is hard
  - dnsmasq
  - Integration of Docker with host's dnsmasq
- NetworkManager
  - Excellent D-Bus API
- ModemManager
  - Excellent D-Bus API
  - Lots of documentation

Docker

- AUFS driver
  - Allows support for NAND based devices
- Currently on docker 1.10.3
  - Backported stability patches
- Journald logging driver
  - Avoids SD card wear
- Seccomp enabled

Log management

- All logs end up in journald
- In RAM 8MB buffer by default
- Configurable log persistence
- Journald allows for structured logs
  - Container logs are annotated with metadata
- Easy to send logs to a central location to store and process

Features

Two stage flashing

- Some boards have internal storage
- Image for these boards is a flasher
  - Automatic copying to internal storage
  - Feedback through LEDs

Host OS updates

- So many options
- It’s one of our biggest focus areas
- resinhup is our current approach
  - Takes advantage of dual root partition
  - Validates everything before changing the state
  - It’s still experimental

https://github.com/resin-os/resinhup/

resin OS

Dual root partition method

- Used by
  - CoreOS, ChromiumOS, Ubuntu Snappy
  - Brillo, Mender.io
- But wastes a lot of space
- We’re experimenting with more advanced approaches
  - ostree
  - docker

ResinHUP

- Integration with docker
- It uses docker to pull the OS image
  - It then unpacks and applies it
- Leveraging important docker features
  - Signed images
  - Programmatic API for fetching
  - Open question: can unify containers and host?

https://github.com/resin-os/resinhup/

resin OS

Automatic emulated testing

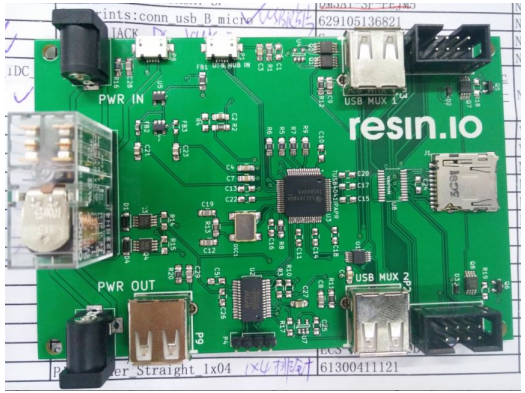
- We support virtual QEMU boards
- Automated basic testing on every PR
  - Booting
  - Networking
- Integrated with our Jenkins

https://github.com/resin-io/autohat

resin OS

Automatic hardware testing

- Manual testing doesn't scale
  - Currently 22 boards
- We built a board that instruments boards
  - GPIO
  - Provisioning
  - SD muxing
  - Wifi testing



https://github.com/resin-io/autohat-rig

resin OS

Device support

ARMv7

- Raspberry Pi 2
- Raspberry Pi 3
- Samsung Artik 5
- SamsungArtik 10
- Beaglebone Black
- Beaglebone Green
- Beaglebone Green Wireless
- Odroid C1/C1+
- Odroid XU4
- SolidRun Hummingboard i2
- Boundary Devices Nitrogen6x
- Parallella Board
- VIA 820 board
- Zynq zc702
- TS4900 single and Quad

ARM64

- Coming soon

ARMv6

- RPI Zero
- RPI model 1 A+

ARMv5

- TS7700

X86_32

- Intel Edison

X86_64

- Intel NUC

Device support

- Easy to add new boards
- Meta-resin handles
  - Userspace
  - Image generation
  - Kernel configuration

Development tools

Development tools

- How do you..
  - Configure network credentials?
  - Provision a device?
  - Develop on the board?
  - Get logs?

Development mode

- Development images have
  - Open SSH server
  - Docker socket exposed over TCP
  - mDNS exposed metadata
- Device is at <hostname>.local

Resin Device Toolbox

- Image configuration
- Wifi credentials
- Hostname
- Persistent logging

```
$ rdt configure ~/Downloads/resinos-dev.img
? Network SSID super_wifi
? Network Key super_secure_password
? Do you want to set advanced settings? Yes
? Device Hostname resin
? Do you want to enable persistent logging? no
Done!
```

resin OS

Resin Device Toolbox

- Automatically detects removable storage
- Won’t wipe your drive!
- Validates after writing

```
$ sudo rdt flash ~/Downloads/resinos-dev.img
? Select drive /dev/disk3 (7.9 GB) - STORAGE DEVICE
? This will erase the selected drive. Are you sure? Yes
Flashing [========================] 100% eta 0s
Validating [========================] 100% eta 0s
```

resin OS

Resin Device Toolbox

- Docker development
- Finds device in local network
- Continously syncs code into the container
- Rebuilds when necessary

```
$ rdt push --source .
* Building..
- Stopping and Removing any previous 'myapp' container
- Removing any existing container images for 'myapp'
- Building new 'myapp' image
```

resin OS

Base Images

- More than 500 images for each supported device type
- Debian, Fedora, Alpine
- Nodejs, python, golang, Java
- Follow docker conventions

https://github.com/resin-io-library/base-images

resin OS

Future

Future

- Roadmap includes..
  - Compressed RAM
  - Docker 1.12
  - Hardware watchdog integration
  - Secure Boot
  - ramoops integration
  - ...
- We interested in your thoughts
- There is lots of room for innovation

Open source

- Website - https://resinos.io/
- Github - https://github.com/resin-os
- Gitter - https://gitter.im/resin-os/chat
- Apache 2 Licence

Questions?