Migrating a x86 based Device to resinOS

The process described in the following has been tested in most but not all parts. Nevertheless I believe that there are no major obstacles left.
Testing has been done on a vanilla Ubuntu 14.04 installation on a x86_64 VirtualBox. For other OSses a modified configuration might have to be used.

The migration process consists of 2 stages.

Migration Stage 1

This stage is initiated by copying the migrate-stage1 script & config file to the device and running it with root privileges.

The stage1 script first makes sure the environment is suitable for migration:
- check operating system and kernel version
- check for existance, use and size of swap
- check availabiltity of required tools
- more to be defined

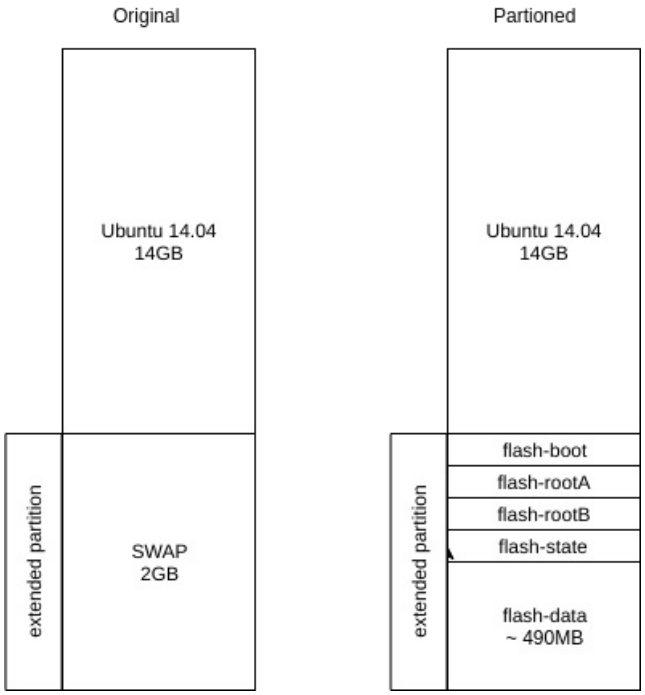
If these checks succeed the script will start preparing the drive.

The script requires a flasher image which will be installed in swap space and booted from to get to stage 2. The flasher image I am working with is a modified standard image for intel-nuc installation. Depending on available space this image can be placed on the device together with the stage1 script or downloaded later. If the image is not present the migrate script requires exact information about the layout of the flasher image as reported by parted:

```
thomas@ganymede:~/$ parted reduced-full.img unit s print
Model:  (file)
Disk /home/thomas/reduced-full.img: 1048576s
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:

Number  Start     End       Size     Type      File system  Flags
 1      8192s     90111s    81920s   primary   fat16        boot, lba
 2      90112s    991231s   901120s  primary   ext4
 3      991232s   999423s   8192s    primary   ext4
 4      999424s   1040383s  40960s   extended               lba
 5      1007616s  1015807s  8192s    logical
 6      1024000s  1040383s  16384s   logical
```

After disabling swap and deleting the swap partition, the stage1 script will use the above information to replicate the layout of the flasher image in the swap space, while enlarging the flash-data partition to take up all trailing space on the device. This will lead to the following partition layout:



After formatting and mounting the flash-data partition we have ~490MB available space to download a flasher-image.
I have modified the standard intel-nuc flasher image by gzipping the contained resinOS image and exchanging the flasher script. This way the image has been shrunken from 2.6GB to 512MB and can be compressed to 320 MB so it will fit into the available space.

It should now be possible to dd the contents of the image partition wise to resin-boot and resin-rootA piping directly from gunzip to dd. The other partitions do not appear to be relevant for the boot process but can be initialized the same way if needed, with the exception of resin-data that is being used. The downloaded image can now be deleted to make space for other activities on resin-data.

On ubuntu 14.04 we can now create a grub2 config in /etc/grub.d/ for our flash-boot partition and install it with `update-grub`
The last step for stage1 is to call `grub-reboot` to which gives us one shot at booting our configuration. All that is left is to reboot the system.

If the system fails to boot into the flasher image it will often be stuck in the boot loader. After reboot it will come up undamaged with the previous configuration except for the swap partition . Ubuntu 14.04 will ignore a missing swap device.

So far the migrate-stage1 script has been working quite reliably.
Following features have not yet been implemented and/or tested.
- preparing the partitions without a flasher-image
- delayed download of the flasher-image
- flashing the data to the partitions from the zipped image

The current implementation works with an uncompressed flasher image and creates and copies partitions directly. The system reboots into the flasher image successfully.

Migration Stage 2

The stage2 script is a modified resin-init-flasher script. This stage has been tested manually but has not yet finished succesfully.

The standard resin-init-flasher script flashes the resinOS image to the installation device and then mounts and modifies it. As we have booted from the device this is problematic. Also once the data is flashed we cannot backup the device any more.

The strategy used to avoid this, is to copy the resinOS image contained in the flasher image to the disk space of the former root partition and loop mount relevant parts of it, that can then be modified.

As the image takes up 1.8GB in uncompressed state we might have to free some space in the ubuntu root partition. What to delete is probably best configured after inspecting the general device setup.

After the image is uncompressed we can loop mount the boot partition and most of the logic of the resin-init-flasher script can run unmodified. After this the image should be ready to boot, once it has been flashed to the boot device.

The next step ist o loop mount the resin-data partition of the resinOS image which contains about 850MB of free space. A customer supplied backup-script can backup relevant data to the resin-data partition.

When both these processes have run we can unmount and remove the resinOS image from the root partition by gzipping it back to its space in flash-rootA or to flash-data which should have about 490MB of free space. After adding data to the image we might experience a growth of the compressed image file, so the resin-data partition is probably the way to go.

This is a part of the process that is hard to calculate in advance. If we backup too much data we might run out of storage space when recompressing the image. There might be ways to get around this but that would most probably involve shrinking and repartitioning the former root partition which might require tools that are not currently present in the resin-flasher image.

After unmounting the original root partition we can flash the resinOS image to the boot device by gunzipping it and piping it to dd. After this we should be able to reboot into resinOS.

Most of these these actions have been manually tested in the resinOS booted from the flasher image.
Yet to be tested is the modification of the resinOS image by the resin-init-flasher script. Because of this I have not yet been able to boot into the installed resinOS successfully.