ONL NOS Installation with Ansible

Created by:
Ron Wilhelmson
ron.wilhelmson@deltaww.com
Version:
Draft 1

Introduction

The basic installation of Network Operating Systems can be automated with scripts and the use of Ansible Playbooks. This guide will outline the installation of Open Network Linux – ONL - on Delta Networks switches.

Objective

The objective of this guide is to document the basic steps required to install a verified ONL installer remotely in an automated way using Ansible and configuration scripts.. Complementary information for basic setup and Ansible use can be referenced at Getting Started with Ansible.

Pre-install Connectivity and Setup

Network and Systems required

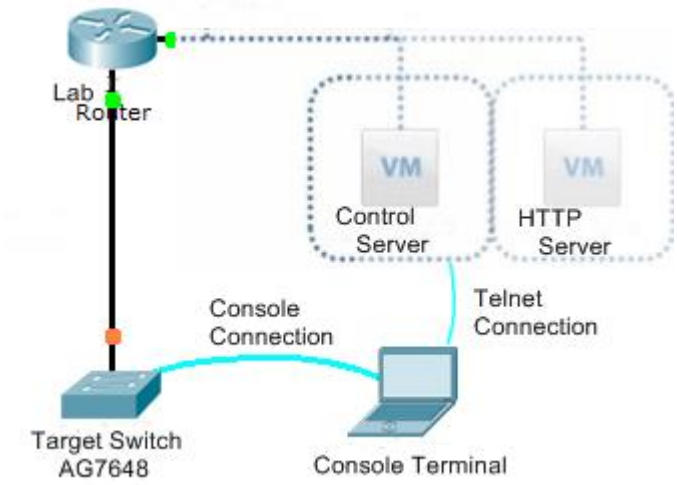
The basic systems required for building and running Ansible playbooks consist of the following:

Control Server – Linux server that runs Ansible and contains playbooks.

Target Switch – Switch that is to be configured running ONIE and may be many.

Web Server – HTTP location where update, license, and install files are located.

Console Terminal – PC with console connection to switch and telnet to control server.



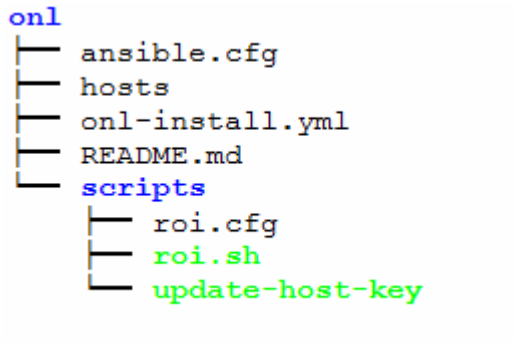
Network Diagram

Note: Please reference Getting Started with Ansible for additional details on system, switch, and server setups.

Ansible Files

The following describes the primary files you will need to get the ONIE playbook running.

Ansible Directory and Files



ansible.cfg

The ansible.cfg file contains all of the configuration variables that can be set and Ansible will read this configuration file when it is initiated

hosts

The hosts file is a reference file used to define the switches and systems that Ansible will be executing tasks on.

Example hosts file:

```
# This is the default ansible 'hosts' file.
#
# It should live in /etc/ansible/hosts
#
#   - Comments begin with the '#' character
#   - Blank lines are ignored
#   - Groups of hosts are delimited by [header] elements
#   - You can enter hostnames or ip addresses
#   - A hostname/ip can be a member of multiple groups
#
# DPR Lab Agema Switch AG7648
#
[switches]
10.62.10.40
[http_server]
10.62.10.22
[control_server]
10.62.10.22
```

playbook

Ansible playbooks are written in the YAML language. For this ONL ONIE Install playbook our playbook will have the following sections:

`hosts` – defines the systems to pass commands to. In this case it is "`localhost`"
`tasks` – commands that will be executed by the playbook

For this playbook we are also using the `user: root` and `sudo: yes` as required.

Note: Please see the references at the end of this document for additional details on commands and modules.

Ansible Playbook NOS Installation

The following YAML format playbook is used to initiate the installation of the ONL NOS installer.

ONIE Install Playbook

The following playbook along with a helper script and associated configuration file are utilized to initiate and run the ONL NOS installer on the target switch running ONIE. Edits to the configuration file and helper script may be required for your specific network details and the version of installer file being loaded. The actual playbook is simple and uncluttered. The playbook initiates the helper shell script with the command task. The Remote ONIE Install script, `roi.sh`, initiates a download of the ONL installation file from the HTTP server to the target switch and then executes it. The playbook, script, and configuration file are shown below.

ONIE Install Playbook: onl-install.yml

```
---

- hosts: localhost
  user: root
  sudo: yes

  tasks:
  - name: Execute script
    command: '/bin/bash ./scripts/roi.sh'
```

Ansible utilizes SSH to connect to the target switch to execute the commands defined in the playbook tasks. It does this by creating a series of tasks and then uses `sftp`, by default, to put the Ansible created commands in the `.ansible/tmp` directory on the target switch. Once the command file is on target system Ansible issues an SSH command to execute the temporary command file and then cleans up by deleting the temporary file. The target switch though in our case has ONIE installed and running. ONIE by default does not support `sftp` transfers so it is difficult for Ansible to move the tasks file to the target switch for execution. Therefore we have a helper script, roi.sh, which is called by the command: task as shown below and sources the associated roi.cfg file for specific details for the installation.

Helper Script onl-install: roi.sh

```
#!/bin/bash

#  Copyright (C) 2018 Ron Wilhelmson <ron.wilhelmson@deltaww.com>
#
#

##
## remote-onie-install
##
## History: 12APR2018 Ron.Wilhelmson Initial Creation
##          04MAY2018 Ron.Wilhelmson Update for NOS install options cumulus or ONL
##          21JUN2018 Ron.Wilhelmson Update for ONL install option

# Dependencies: roi.cfg in same directory

# Set environment variables from roi.cfg for switch and server names/IPs
source /root/ansible/scripts/roi.cfg

echo ""
echo "Target switch set to: $target_switch"
echo ""
echo "HTTP server set to: $http_server"
echo ""
echo "NOS to be installed on Target switch: $NOS_install"

echo "Sending install command to switch"


case $NOS_install in
   cumulus) /usr/bin/ssh -a -l root $target_switch /bin/onie-nos-install
http://"$http_server"/cumulus-linux-3.5.0-bcm-amd64.bin
      echo ""
      ;;

   ONL) /usr/bin/ssh -a -l root $target_switch /bin/onie-nos-install
http://"$http_server"/onie-ONL-installer-x86_64-ag7648
      echo ""
      ;;

   ocnos) /usr/bin/ssh -a -l root $target_switch /bin/onie-nos-install
http://"$http_server"/DELTA_AGC7648A-OcNOS-1.3.2.137-DC_MPLS_ZEBM-S0-P0-installer
      ;;

   onl) {
        sleep 10
```

```
        echo /bin/install_url http://"$http_server"/AG9032v1/ONL-2.0.0_ONL-OS_2018-01-
17.0840-6f80df8_AMD64_INSTALLED_INSTALLER
        sleep 60
        echo exit
        } | telnet $target_switch

esac

echo "Waiting for onie to download and install"

sleep 60

echo "Rebooting switch"

/usr/bin/ssh -l root $target_switch /sbin/reboot
```

Config File remote-onie-install: roi.cfg

```
#  Copyright (C) 2018 Ron Wilhelmson <ron.wilhelmson@deltaww.com>
#
#

##
## remote-onie-install variable definitions
##
## History: 12APR2018 Ron.Wilhelmson Initial Creation
##          04MAY2018 Ron.Wilhelmson Updated to set NOS install variable

# Reliancies: roi.sh

# set network operating system (NOS) to be installed on target switch
# current options are: cumulus
#                      ONL

NOS_install=ONL

# set the switch and server names/IPs according to your systems

# target_switch is the switch that NOS is being installed on

target_switch=10.62.10.40

# http_server is the URL server where the NOS bin and license files are located

http_server=10.62.10.22
```

Playbook Execution

The Ansible command to run this playbook installer is as follows:

```
ansible-playbook remote-onie-install.yml --ask-pass -v
```

Note: The password that will be requested, with the `--ask-pass` option, is for the ONIE root account which is blank so a return is required when the playbook runs and asks for "SSH password:" and the `-v` option increases the verbosity of the execution and provides a more detailed response.

The following Ansible playbook execution shows the output generated and the resulting information generated on the control server through a terminal connection from your PC or directly if working on a workstation.

Example Playbook Output: onl-install.yml

```
root@DPR-LABVM-01:~/ansible# ansible-playbook onl-install.yml --ask-pass --ask-sudo -v
Using /root/ansible/ansible.cfg as config file
SSH password:
SUDO password[defaults to SSH password]:

PLAY [localhost]
*********************************************************************************
************

TASK [Gathering Facts]
*********************************************************************************
******
ok: [localhost]

TASK [Execute script to install ONL]
*********************************************************************
changed: [localhost] => changed=true
  cmd:
  - /bin/bash
  - ./scripts/roi.sh
  delta: '0:02:10.016686'
  end: '2018-06-22 12:43:44.969652'
  rc: 0
  start: '2018-06-22 12:41:34.952966'
  stderr: |-
    Connection closed by foreign host.
    ./scripts/roi.sh: line 44: echo: write error: Broken pipe
  stderr_lines:
  - Connection closed by foreign host.
  - './scripts/roi.sh: line 44: echo: write error: Broken pipe'
  stdout: |2-

    Target switch set to: 10.62.10.42

    HTTP server set to: 10.62.10.22

    NOS to be installed on Target switch: onl
    Sending install command to switch
    Trying 10.62.10.42...
    Connected to 10.62.10.42.
    Escape character is '^]'.

    To check the install status inspect /var/log/onie.log.
    Try this:  tail -f /var/log/onie.log

    ** Installer Mode Enabled **
    ONIE:/ # /bin/install_url http://10.62.10.22/AG9032v1/ONL-2.0.0_ONL-OS_2018-01-1
```

```
    7.0840-6f80df8_AMD64_INSTALLED_INSTALLER
    Info: Fetching http://10.62.10.22/AG9032v1/ONL-2.0.0_ONL-OS_2018-01-17.0840-
6f80df8_AMD64_INSTALLED_INSTALLER ...
    Connecting to 10.62.10.22 (10.62.10.22:80)
    installer            31% |*********                       | 53859k  0:00:02
ETAinstaller             97% |****************************** |   164M  0:00:00
ETAinstaller            100% |*******************************|   168M  0:00:00 ETA
    installer: computing checksum of original archive
    installer: checksum is OK
    installer: extracting pad
    1+0 records in
    1+0 records out
    512 bytes (512B) copied, 0.000058 seconds, 8.4MB/s
    installer: copying file before resetting pad
    installer: resetting pad
    1+0 records in
    1+0 records out
    512 bytes (512B) copied, 0.000042 seconds, 11.6MB/s
    installer: extracting shar into /tmp/sfx-bZgxSV
    installer: invoking installer installer.sh
    Archive:  /tmp/sfx-bZgxSV/onie-installer.zip
      inflating: onl-loader-initrd-amd64.cpio.gz
    65051 blocks
    Archive:  /tmp/sfx-bZgxSV/onie-installer.zip
      inflating: preinstall.sh
    Hello from preinstall
    Chroot is /tmp/sfx-bZgxSV/initrd-mLDROO
    57267 blocks
    57267 blocks
    57267 blocks
    mke2fs 1.42.7 (21-Jan-2013)
    ext2fs_check_if_mount: Can't check if filesystem is mounted due to missing mtab
file while determining whether /dev/sda3 is mounted.
    fs_types for mke2fs.conf resolution: 'ext4', 'small'
    Discarding device blocks:   1024/131072             done
    Filesystem label=ONL-BOOT
    OS type: Linux
    Block size=1024 (log=0)
    Fragment size=1024 (log=0)
    Stride=0 blocks, Stripe width=0 blocks
    32768 inodes, 131072 blocks
    6553 blocks (5.00%) reserved for the super user
    First data block=1
    Maximum filesystem blocks=67371008
    16 block groups
    8192 blocks per group, 8192 fragments per group
    2048 inodes per group
    Superblock backups stored on blocks:
            8193, 24577, 40961, 57345, 73729

    Allocating group tables:  0/16     done
    Writing inode tables:  0/16     done
    Creating journal (4096 blocks): done
    Writing superblocks and filesystem accounting information:  0/16     done

    mke2fs 1.42.7 (21-Jan-2013)
```

```
    ext2fs_check_if_mount: Can't check if filesystem is mounted due to missing mtab
file while determining whether /dev/sda4 is mounted.
    fs_types for mke2fs.conf resolution: 'ext4', 'small'
    Discarding device blocks:   1024/131072             done
    Filesystem label=ONL-CONFIG
    OS type: Linux
    Block size=1024 (log=0)
    Fragment size=1024 (log=0)
    Stride=0 blocks, Stripe width=0 blocks
    32768 inodes, 131072 blocks
    6553 blocks (5.00%) reserved for the super user
    First data block=1
    Maximum filesystem blocks=67371008
    16 block groups
    8192 blocks per group, 8192 fragments per group
    2048 inodes per group
    Superblock backups stored on blocks:
            8193, 24577, 40961, 57345, 73729

    Allocating group tables:  0/16     done
    Writing inode tables:  0/16     done
    Creating journal (4096 blocks): done
    Writing superblocks and filesystem accounting information:  0/16     done

    mke2fs 1.42.7 (21-Jan-2013)
    ext2fs_check_if_mount: Can't check if filesystem is mounted due to missing mtab
file while determining whether /dev/sda5 is mounted.
    fs_types for mke2fs.conf resolution: 'ext4'
    Discarding device blocks:   4096/262144             done
    Filesystem label=ONL-IMAGES
    OS type: Linux
    Block size=4096 (log=2)
    Fragment size=4096 (log=2)
    Stride=0 blocks, Stripe width=0 blocks
    65536 inodes, 262144 blocks
    13107 blocks (5.00%) reserved for the super user
    First data block=0
    Maximum filesystem blocks=268435456
    8 block groups
    32768 blocks per group, 32768 fragments per group
    8192 inodes per group
    Superblock backups stored on blocks:
            32768, 98304, 163840, 229376

    Allocating group tables: 0/8   done
    Writing inode tables: 0/8   done
    Creating journal (8192 blocks): done
    Writing superblocks and filesystem accounting information: 0/8   done

    mke2fs 1.42.7 (21-Jan-2013)
    ext2fs_check_if_mount: Can't check if filesystem is mounted due to missing mtab
file while determining whether /dev/sda6 is mounted.
    fs_types for mke2fs.conf resolution: 'ext4'
    Discarding device blocks:     4096/30897408 3674112/30897408
7868416/3089740812062720/3089740815732736/3089740819927040/3089740823597056/3089740827
791360/30897408                 done
```

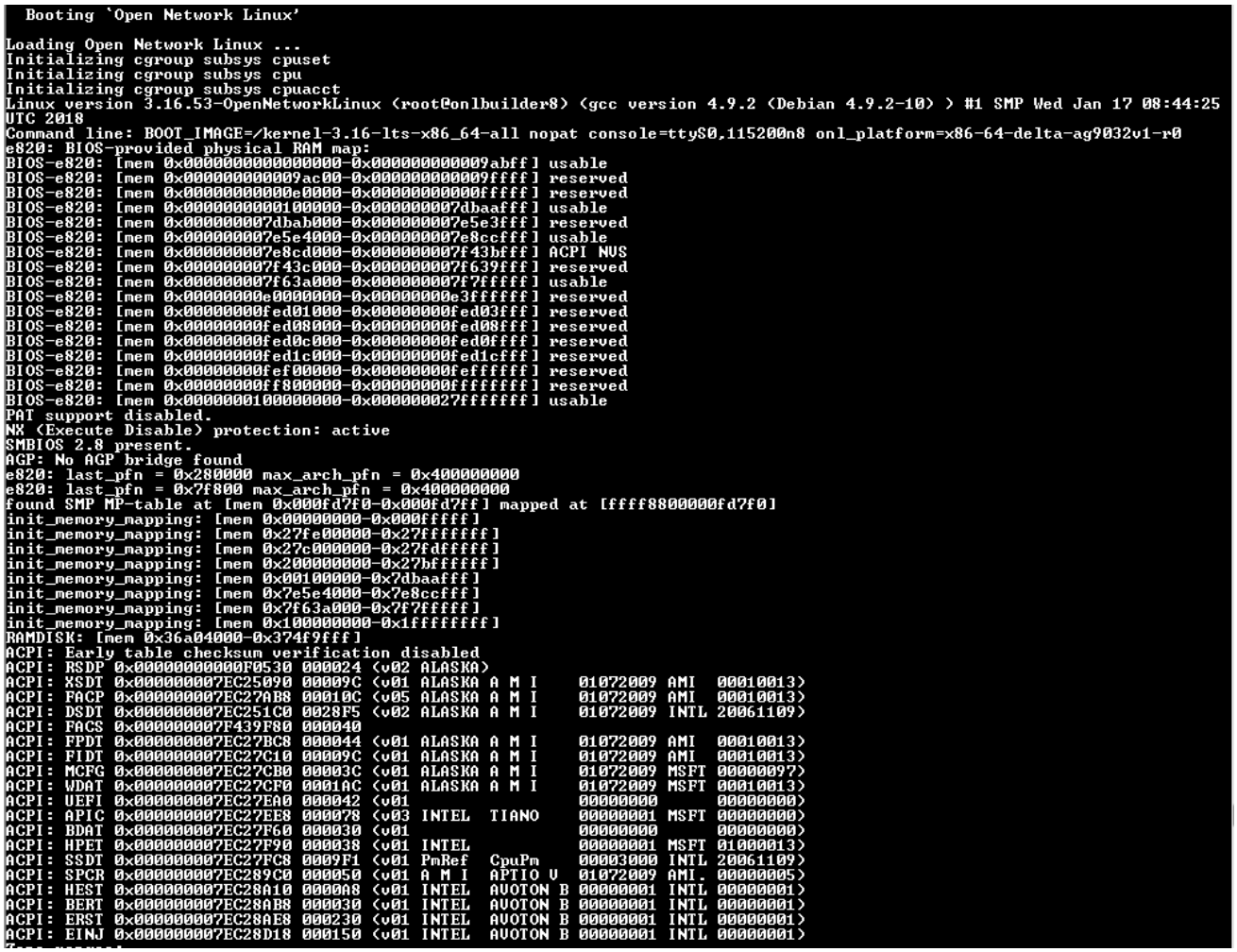
```
    Filesystem label=ONL-DATA
    OS type: Linux
    Block size=4096 (log=2)
    Fragment size=4096 (log=2)
    Stride=0 blocks, Stripe width=0 blocks
    7725056 inodes, 30897408 blocks
    1544870 blocks (5.00%) reserved for the super user
    First data block=0
    Maximum filesystem blocks=4294967296
    943 block groups
    32768 blocks per group, 32768 fragments per group
    8192 inodes per group
    Superblock backups stored on blocks:
            32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208,
            4096000, 7962624, 11239424, 20480000, 23887872

    Allocating group tables:   0/943136/943       done
    Writing inode tables:   0/943       done
    Creating journal (32768 blocks): done
    Writing superblocks and filesystem accounting information:   0/943       done

    Installing for i386-pc platform.
    Installation finished. No error reported.
    + . /tmp/sfx-bZgxSV/postinst-XhMB9i
    + set +x
    Archive:  /tmp/sfx-bZgxSV/onie-installer.zip
      inflating: postinstall.sh
    Hello from postinstall
    Chroot is /tmp/sfx-bZgxSV/initrd-mLDROO
    Hit CR to continue, CTRL-D or CTRL-C to stop... Terminated
    ONIE:/ # Waiting for onie to download and install
    Rebooting switch
  stdout_lines: <omitted>

PLAY RECAP
*********************************************************************************
******************
localhost                  : ok=2    changed=1    unreachable=0    failed=0

root@DPR-LABVM-01:~/ansible#
```

The console connection to the target switch will display the following screens when the playbook is executed:

```
ONIE:/ # discover: installer mode detected.
Stopping: discover... done.
ONIE: Executing installer: http://10.62.10.22/AG9032v1/ONL-2.0.0_ONL-OS_2018-01-17.0840-6f80df8_AMD64_INSTALLED_INSTALLER
Found installer tmpfs on /tmp/sfx-bZgxSV (/tmp) using opts rw,noatime
Unpacking ONL installer files...
Extracting from /tmp/sfx-bZgxSV/onie-installer.zip: onl-loader-initrd-amd64.cpio.gz ...
Extracting initrd to /tmp/sfx-bZgxSV/initrd-mLDROO
Setting up /dev
Setting up /run
Setting up mounts
Examining /dev/sda5 --> ONL-IMAGES
Examining /dEXT3-fs (sda2): error: couldn't mount because of unsupported optional features (240)
ev/sda4 --> ONL-EXT2-fs (sda2): error: couldn't mount because of unsupported optional features (240)
CONFIG
Examining /dev/sda3 --> ONL-BOOT
Examining /dev/sda2 --> ONIE-BOOT
Found ONIE-BOOT at /dev/sda2
Mounting ONIE-BOOT (/dev/sda2) as /mnt/onie-boot
Launching ONL installer
Unmounting /mnt/onie-boot (--force)
Extracting from /tmp/sfx-bZgxSV/onie-installer.zip: preinstall.sh ...
Invoking pre-install actions
Examining /dev/sda5 --> ONL-IMAGES
Examining /dev/sda4 --> ONL-CONFIG
Examining /dev/sda3 --> ONL-BOOT
Examining /dev/sda2 --> ONIE-BOOT
Found ONIE boot partition at /dev/sda2
Examining /dev/sda6 --> ONL-DATA
Found a clean GPT partition table
getting installer configuration
+ blkid
found ONIE boot device /dev/sda2
found ONIE boot mounted at /mnt/onie-boot
found ONIE initrd at /mnt/onie-boot/onie/initrd.img-4.1.28-onie
extracting initrd /mnt/onie-boot/onie/initrd.img-4.1.28-onie
+ /bin/mkdir /tmp/sfx-bZgxSV/chroot-vpWUHn.d
+ xz -dc /mnt/onie-boot/onie/initrd.img-4.1.28-onie | cpio -imd
preparing chroot in /tmp/sfx-bZgxSV/chroot-vpWUHn.d
+ mount -v -t proc proc /tmp/sfx-bZgxSV/chroot-vpWUHn.d/proc
+ mount -v -t sysfs sysfs /tmp/sfx-bZgxSV/chroot-vpWUHn.d/sys
+ mount -v -t devpts devpts /tmp/sfx-bZgxSV/chroot-vpWUHn.d/dev/pts
extracting initrd /mnt/onie-boot/onie/initrd.img-4.1.28-onie
+ /bin/mkdir /tmp/sfx-bZgxSV/chroot-fEOS0w.d
+ xz -dc /mnt/onie-boot/onie/initrd.img-4.1.28-onie | cpio -imd
preparing chroot in /tmp/sfx-bZgxSV/chroot-fEOS0w.d
+ mount -v -t proc proc /tmp/sfx-bZgxSV/chroot-fEOS0w.d/proc
+ mount -v -t sysfs sysfs /tmp/sfx-bZgxSV/chroot-fEOS0w.d/sys
+ mount -v -t devpts devpts /tmp/sfx-bZgxSV/chroot-fEOS0w.d/dev/pts
+ chroot /tmp/sfx-bZgxSV/chroot-fEOS0w.d /bin/sh -c IFS=;onie-sysinfo -h
un-mounting mounts points in chroot /tmp/sfx-bZgxSV/chroot-fEOS0w.d
+ umount -v /tmp/sfx-bZgxSV/chroot-fEOS0w.d/sys
+ umount -v /tmp/sfx-bZgxSV/chroot-fEOS0w.d/proc
+ umount -v /tmp/sfx-bZgxSV/chroot-fEOS0w.d/dev/pts
cleaning up chroot in /tmp/sfx-bZgxSV/chroot-fEOS0w.d
+ /bin/rm -fr /tmp/sfx-bZgxSV/chroot-fEOS0w.d
un-mounting mounts points in chroot /tmp/sfx-bZgxSV/chroot-vpWUHn.d
+ umount -v /tmp/sfx-bZgxSV/chroot-vpWUHn.d/sys
+ umount -v /tmp/sfx-bZgxSV/chroot-vpWUHn.d/proc
+ umount -v /tmp/sfx-bZgxSV/chroot-vpWUHn.d/dev/pts
cleaning up chroot in /tmp/sfx-bZgxSV/chroot-vpWUHn.d
+ /bin/rm -fr /tmp/sfx-bZgxSV/chroot-vpWUHn.d
ONL Installer @ONLVERSION@
ONL installer running chrooted.
Detected platform x86-64-delta-ag9032v1-r0
trying a GRUB based installer
+ blkid
found ONIE boot device /dev/sda2
```

Once the installer completes downloading from the HTTP server it will install and reboot with the default option to load ONL:

When the ONL software completes loading the switch will boot into Open Network Linux:


Booting 'Open Network Linux'
Loading Open Network Linux ...
Initializing cgroup subsys cpuset
Initializing cgroup subsys cpu
Initializing cgroup subsys cpuacct
Linux version 3.16.53-OpenNetworkLinux (root@onlbuilder8) (gcc version 4.9.2 (Debian 4.9.2-10)) #1 SMP Wed Jan 17 08:44:25 UTC 2018
Command line: BOOT_IMAGE=/kernel-3.16-lts-x86_64-all nopat console=ttyS0,115200n8 onl_platform=x86-64-delta-ag9032v1-r0
e820: BIOS-provided physical RAM map:
BIOS-e820: [mem 0x0000000000000000-0x000000000009abff] usable
BIOS-e820: [mem 0x000000000009ac00-0x000000000009ffff] reserved
BIOS-e820: [mem 0x00000000000e0000-0x00000000000fffff] reserved
BIOS-e820: [mem 0x0000000000100000-0x000000007dbaafff] usable
BIOS-e820: [mem 0x000000007dbab000-0x000000007e5e3fff] reserved
BIOS-e820: [mem 0x000000007e5e4000-0x000000007e8ccfff] usable
BIOS-e820: [mem 0x000000007e8cd000-0x000000007f43bfff] ACPI NVS
BIOS-e820: [mem 0x000000007f43c000-0x000000007f639fff] reserved
BIOS-e820: [mem 0x000000007f63a000-0x000000007f7fffff] usable
BIOS-e820: [mem 0x00000000e0000000-0x00000000e3ffffff] reserved
BIOS-e820: [mem 0x00000000fed01000-0x00000000fed03fff] reserved
BIOS-e820: [mem 0x00000000fed08000-0x00000000fed08fff] reserved
BIOS-e820: [mem 0x00000000fed0c000-0x00000000fed0ffff] reserved
BIOS-e820: [mem 0x00000000fed1c000-0x00000000fed1cfff] reserved
BIOS-e820: [mem 0x00000000fef00000-0x00000000feffffff] reserved
BIOS-e820: [mem 0x00000000ff800000-0x00000000ffffffff] reserved
BIOS-e820: [mem 0x0000000100000000-0x000000027fffffff] usable
PAT support disabled.
NX (Execute Disable) protection: active
SMBIOS 2.8 present.
AGP: No AGP bridge found
e820: last_pfn = 0x280000 max_arch_pfn = 0x400000000
e820: last_pfn = 0x7f800 max_arch_pfn = 0x400000000
found SMP MP-table at [mem 0x000fd7f0-0x000fd7ff] mapped at [ffff8800000fd7f0]
init_memory_mapping: [mem 0x00000000-0x000fffff]
init_memory_mapping: [mem 0x27fe00000-0x27fffffff]
init_memory_mapping: [mem 0x27c000000-0x27fdfffff]
init_memory_mapping: [mem 0x200000000-0x27bffffff]
init_memory_mapping: [mem 0x00100000-0x7dbaafff]
init_memory_mapping: [mem 0x7e5e4000-0x7e8ccfff]
init_memory_mapping: [mem 0x7f63a000-0x7f7fffff]
init_memory_mapping: [mem 0x100000000-0x1fffffffff]
RAMDISK: [mem 0x36a04000-0x374f9fff]
ACPI: Early table checksum verification disabled
ACPI: RSDP 0x00000000000F0530 000024 (v02 ALASKA)
ACPI: XSDT 0x000000007EC25090 00009C (v01 ALASKA A M I 01072009 AMI 00010013)
ACPI: FACP 0x000000007EC27AB8 00010C (v05 ALASKA A M I 01072009 AMI 00010013)
ACPI: DSDT 0x000000007EC251C0 0028F5 (v02 ALASKA A M I 01072009 INTL 20061109)
ACPI: FACS 0x000000007F439F80 000040
ACPI: FPDT 0x000000007EC27BC8 000044 (v01 ALASKA A M I 01072009 AMI 00010013)
ACPI: FIDT 0x000000007EC27C10 00009C (v01 ALASKA A M I 01072009 AMI 00010013)
ACPI: MCFG 0x000000007EC27CB0 00003C (v01 ALASKA A M I 01072009 MSFT 00000097)
ACPI: WDAT 0x000000007EC27CF0 0001AC (v01 ALASKA A M I 01072009 MSFT 00010013)
ACPI: UEFI 0x000000007EC27EA0 000042 (v01 00000000 00000000)
ACPI: APIC 0x000000007EC27EE8 000078 (v03 INTEL TIANO 00000001 MSFT 00000000)
ACPI: BDAT 0x000000007EC27F60 000030 (v01 00000000 00000000)
ACPI: HPET 0x000000007EC27F90 000038 (v01 INTEL 00000001 MSFT 01000013)
ACPI: SSDT 0x000000007EC27FC8 0009F1 (v01 PmRef CpuPm 00003000 INTL 20061109)
ACPI: SPCR 0x000000007EC289C0 000050 (v01 A M I APTIO V 01072009 AMI. 00000005)
ACPI: HEST 0x000000007EC28A10 0000A8 (v01 INTEL AVOTON B 00000001 INTL 00000001)
ACPI: BERT 0x000000007EC28AB8 000030 (v01 INTEL AVOTON B 00000001 INTL 00000001)
ACPI: ERST 0x000000007EC28AE8 000230 (v01 INTEL AVOTON B 00000001 INTL 00000001)
ACPI: EINJ 0x000000007EC28D18 000150 (v01 INTEL AVOTON B 00000001 INTL 00000001)

```
* Open Network Linux Loader
*
*        Version: ONL-2.0.0
*             Id: 2018-01-17.08:40-6f80df8
*
*       Platform: x86-64-delta-ag9032v1-r0
*            ma1: 00:18:23:30:e6:3a
*
************************************************************
[ boot-config ]
NETDEV=ma1
BOOTMODE=INSTALLED
SWI=images::latest


Press Control-C now to enter the interactive loader shell.

[ Starting Autoboot ]
[ Configuring Interfaces ]
IPv6: ADDRCONF(NETDEV_UP): ma1: link is not ready
8021q: adding VLAN 0 to HW filter on device ma1
[ BOOTMODE is installed. ]
DEBUG:swiget:+ blkid
INFO:swiget:found 'latest' swi /mnt/onl/images/ONL-2.0.0_ONL-OS_2018-01-17.0840-6f80df8_AMD64.swi
[ Un-populated /mnt/onl/data, will (re-)image ]
extracting SWI /mnt/onl/images/ONL-2.0.0_ONL-OS_2018-01-17.0840-6f80df8_AMD64.swi --> /mnt/onl/data/swiprep-0PDBHL
igb 0000:00:14.0 ma1: igb: ma1 NIC Link is Up 1000 Mbps Full Duplex, Flow Control: None
IPv6: ADDRCONF(NETDEV_CHANGE): ma1: link becomes ready
extracting rootfs /mnt/onl/data/swiprep-0PDBHL/rootfs.sqsh --> /mnt/onl/data
Parallel unsquashfs: Using 4 processors
13249 inodes (15156 blocks) to write


create_inode: socket /mnt/onl/data/dev/log ignored

created 11141 files
created 1980 directories
created 2096 symlinks
created 1 devices
created 2 fifos
recording SWI /mnt/onl/images/ONL-2.0.0_ONL-OS_2018-01-17.0840-6f80df8_AMD64.swi --> /tmp/swiprep-4oWXgM
[ Trying dir:data:/... ]
creating 1048576k of tmpfs in /tmp/boot-tmpfs-DHQVKS
Mounting dir:data:/
DEBUG:swimount:+ blkid
DEBUG:swimount:+ mount -o rw,remount /mnt/onl/data
EXT4-fs (sdb6): re-mounted. Opts: data=ordered
INIT: version 2.88 booting
2018-06-23 00:44:26.864 INFO    system-upgrade: ************************************************************
2018-06-23 00:44:26.865 INFO    system-upgrade: * System Compatibility Version Check
2018-06-23 00:44:26.865 INFO    system-upgrade: ************************************************************
2018-06-23 00:44:26.865 INFO    system-upgrade:
2018-06-23 00:44:26.866 INFO    system-upgrade: Current System Compatibility Version: 2
2018-06-23 00:44:26.867 INFO    system-upgrade:    Next System Compatibility Version: 2
2018-06-23 00:44:26.867 INFO    system-upgrade:
2018-06-23 00:44:26.867 INFO    system-upgrade: System version 2 is current.
2018-06-23 00:44:26.868 INFO    system-upgrade:
2018-06-23 00:44:26.868 INFO    system-upgrade: ************************************************************
2018-06-23 00:44:27.646 INFO    loader-upgrade: ************************************************************
2018-06-23 00:44:27.647 INFO    loader-upgrade: * Loader Upgrade Check
2018-06-23 00:44:27.647 INFO    loader-upgrade: ************************************************************
2018-06-23 00:44:27.647 INFO    loader-upgrade:
2018-06-23 00:44:27.649 INFO    loader-upgrade: Current Loader Version: ONL-2.0.0,2018-01-17.08:40-6f80df8
2018-06-23 00:44:27.649 INFO    loader-upgrade:    Next Loader Version: ONL-2.0.0,2018-01-17.08:40-6f80df8
2018-06-23 00:44:27.649 INFO    loader-upgrade:
2018-06-23 00:44:27.649 INFO    loader-upgrade: Loader version ONL-2.0.0,2018-01-17.08:40-6f80df8 is current.
2018-06-23 00:44:27.650 INFO    loader-upgrade:
2018-06-23 00:44:27.650 INFO    loader-upgrade: ************************************************************
```

The install completes when the following ONL login screen is displayed. At this prompt you can access the ONL environment with user ID **root** and password **onl**:

```
Using makefile-style concurrent boot in runlevel S.
Starting the hotplug events dispatcher: udevdsystemd-udevd[687]: starting version 215
.
Synthesizing the initial hotplug events...done.
Waiting for /dev to be fully populated...gpio_ich: GPIO from 452 to 511 on gpio_ich
done.
Setting kernel variables ...done.
Setting up resolvconf...done.
Configuring network interfaces...done.
INIT: Entering runlevel: 2
Using makefile-style concurrent boot in runlevel 2.
Starting enhanced syslogd: rsyslogd.
Starting Fault Agent: faultd.
Starting ONLP SNMP Agent: onlp-snmpd.
Starting ONLP Platform Agent: onlpd.
Starting OpenBSD Secure Shell server: sshd.
Starting SNMP services::  snmpdnf_conntrack: automatic helper assignment is deprecated and it will be removed soon. Use the
iptables CT target to attach helpers instead.
Warning: Could not probe for any interfaces
Stopping watchdog keepalive daemon....
Starting watchdog daemon....

Open Network Linux OS ONL-2.0.0, 2018-01-17.08:40-6f80df8

localhost login:
```

Example Playbooks and Projects:

The following git repository has example playbooks created for this guide and other applications that may be useful:

https://github.com/DeltaProducts/Getting-Started-with-Ansible

References:

Ansible User Guide
http://docs.ansible.com/ansible/devel/user_guide/intro_getting_started.html

Ansible Installation Guide

http://docs.ansible.com/ansible/latest/installation_guide/intro_installation.html

Ansible ansible-playbook Guide:

https://docs.ansible.com/ansible/2.4/ansible-playbook.html

Ansible Commands

http://docs.ansible.com/ansible/latest/modules/list_of_commands_modules.html

Ansible Modules

http://docs.ansible.com/ansible/latest/modules/list_of_all_modules.html

Tera Term Guide

https://ttssh2.osdn.jp/index.html.en

Open Network Install Environment (ONIE) Installation Guide

https://github.com/DeltaProducts/SolutionCenter/blob/master/ONIE%20recovery%20from%20bootable%20USB.pdf

Apache Web Server Setup Guide

https://www.digitalocean.com/community/tutorials/how-to-install-the-apache-web-server-on-ubuntu-16-04