





IC-OS

 Date	
 Slides	
 Courses	<u>IDP</u>
 Lecture video link	

IC-OS

IC-OS contains:

- **SetupOS:** Responsible for booting a new replica node and installing HostOS and GuestOS.
- **HostOS:** The operating system that runs on the host machine. Its main responsibility is to launch and run the GuestOS in a virtual machine. In terms of its capabilities, it is intentionally limited by design.
- **GuestOS:** The operating system that runs inside a virtual machine on the HostOS. The core IC protocol is executed within the GuestOS.
- **Boundary-guestOS:** The operating system that runs on boundary nodes.

The Ubuntu-based IC OS is built by:

- creating a root filesystem image using docker -- this is based on the official Ubuntu docker image and simply adds the OS kernel plus our required services to it.
- converting this root filesystem into filesystem images for `+/+` and `+/boot+` via `+mke2fs+`

IC-OS is created in 2 steps:

1. build a docker image
2. then transformed into "bare-metal" or "virtual-metal" images

This approach allows for a minimal, controlled, and well understood system - which is key for a secure platform.

Docker Image:

There are two dockerfile for each of the OS

1. Dockerfile.base

The Dockerfile.base takes care of installing all upstream Ubuntu packages because the versions of these packages can change at any given time

In order to maintain build determinism, once a week, the CI pipeline builds a new base image for each OS.

2. Dockerfile

The Dockerfile builds off the published base image and configures and assembles the main disk-image.

The docker image is then transformed into a bootable "bare-metal" or "virtual-metal" VM image for use outside containerization.

Boundary-guestOS

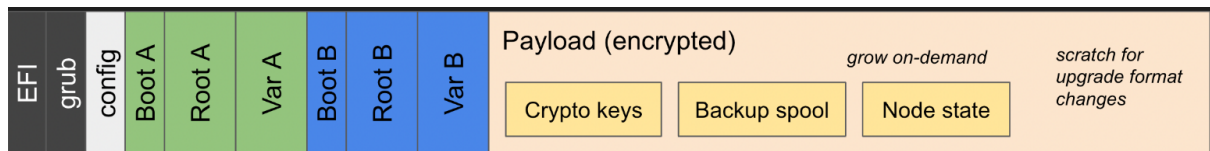
The following essential IC-specific service are started in the IC-OS boot sequence:

- Mount `/boot` filesystems (including `/boot/config`)
- Save machine-id
- Set up ssh host keys
- Set up node exporter keys
- Config injection
- Set up ssh account keys
- Generate network configuration
- Set up hostname
- IPv6 address monitor / retry

- Start node exporter
- Start nginx
- Start boundary-node-control-plane
- Start ic_router_control_plane_watcher
- Start icx_proxy
- Start danted socks-proxy

GuestOS

disk layout:



the following essential IC-specific service are started in the IC-OS boot sequence:

- Mount `+/boot+` filesystems (including `+/boot/config+`)
- Save machine-id
- Set up ssh host keys
- Set up node exporter keys
- Set up key for block device encryption
- Set up and mount `+/var+` filesystem
- Set up of logical volume manager
- One-time set up of filesystems in LVs
- Mount of file systems in LVs
- IC node config injection
- Set up ssh account keys
- Generate network configuration
- Set up hostname

- IPv6 address monitor / retry
- Upgrade data store
- nftables reload on change
- Start journalbeat
- Start node exporter

Setup OS

SetupOS is used for the operating system installing the IC-OS stack.

The sequence of the scripts is defined in the main installation script, `setupos.sh`. The order of execution is as follows:

```
hardware.sh # Verifies the system's hardware components
network.sh  # Tests network connectivity and reachability of the NNS
disk.sh     # Purges existing LVM configurations and partitions
hostos.sh   # Installs and configures the HostOS operating system
guestos.sh  # Installs and configures the GuestOS operating system
devices.sh  # Handles the HSM
```

Host OS

The term HostOS is used for the operating system running on the physical machine. The purpose of this system is to enable virtualization for any node running on top.

Installation process

Install `virsh`

1. Install the required packages

```
apt install libvirt-daemon-system libvirt-clients bridge-utils virtinst virt-manager
```

2. add the user to the `kvm` and `libvirt` group

```
adduser your_username libvirt
adduser your_username kvm
```

Download the disk image and the iso image for the virtual machine. For the first stage, I tried to create a VM for ubuntu using the `virsh`

1. download the `.iso` file for ubuntu

```
wget https://releases.ubuntu.com/20.04/ubuntu-20.04.6-live-server-amd64.iso
```

2. Install the osinfo-db-tools to add the server name
(<https://askubuntu.com/questions/1070500/why-doesnt-osinfo-query-os-detect-ubuntu-18-04>)

```
apt install osinfo-db-tools
```

2. Download the osinfo-db and import it.

```
`wget -O "/tmp/osinfo-db.tar.xz" "https://releases.pagure.org/libosinfo/osinfo-db-20200325.tar.xz"`
`osinfo-db-import --local "/tmp/osinfo-db.tar.xz"``
```

2. start the default network (<https://www.xmodulo.com/network-default-is-not-active.html>)

```
virsh net-start default
```

2. start the VM using :

```
virt-install --name server-01 \
--os-type linux \
--os-variant ubuntu20.04 \
--ram 1024 \
--disk /kvm/disk/server-01.img,device=disk,bus=virtio,size=10,format=qcow2 \
--graphics vnc,listen=0.0.0.0 \
--noautoconsole \
--hvm \
```

```
--cdrom /kvm/iso/ubuntu-20.04.6-live-server-amd64.iso \  
--boot cdrom,hd
```

This is letting me install ubuntu as a kvm. However I am not able to figure out how to install it a