This is the follow along tutorial for the T-Stack demo shown at zuberlin - running a validator in TDX using the yocto minimal VM.

Setting up the yocto minimal VM

Follow the Getting Started

instructions here, making sure all dependencies are installed.

Create an empty directory to hold your working files.

mkdir validator_demo cd validator_demo

Clone the Yocto meta layer source using yocto manifest as show below.

repo init -u https://github.com/flashbots/yocto-manifests.git -b tdx

Fetch all the repositories.

repo sync

Setup the Yocto OE Init scripts by sourcing setup

script.

source setup

Build the image by using the provided Makefile

. Note that this will take time.

make build

Adding lighthouse to the image

Create the directory for the lighthouse bitbake recipe

mkdir srcs/poky/meta-evm/recipes-nodes/lighthouse

While yocto is nice for reproducible builds, it is not straight forward to add rust projects into it's native build system. We're going to build lighthouse statically outside of yocto - assuming this can be done in a reproducible way - and then our yocto bitbake recipe will only copy the binary into the image.

git clone https://github.com/sigp/lighthouse cd lighthouse RUSTFLAGS="-C target-feature=+crt-static" cargo build --target x86_64-unknown-linux-gnu --release cd ../

Copy the statically linked binary into the recipe directory

cp lighthouse/target/x86_64-unknown-linux-gnu/lighthouse srcs/poky/meta-evm/recipes-nodes/lighthouse

Add the lighthouse recipe

cat < srcs/poky/meta-evm/recipes-nodes/lighthouse/lighthouse_dev.bb DESCRIPTION = "Copy binary to the image" LICENSE = "CLOSED" FILESEXTRAPATHS:prepend := "${THISDIR}:" BINARY = "lighthouse" SRC_URI += "file://${BINARY}" SRC_URI += "file://init" S = "${WORKDIR}"

INITSCRIPT_NAME = "${BINARY}" INITSCRIPT_PARAMS = "defaults 99"

inherit update-rc.d

do_install() { install -d ${D}${bindir} install -m 0777 ${BINARY} ${D}${bindir} install -d ${D}${sysconfdir}/init.d cp init ${D}${sysconfdir}/init.d/${BINARY} chmod 755 ${D}${sysconfdir}/init.d/${BINARY} } FILES_${PN} += "${bindir}" INHIBIT_PACKAGE_DEBUG_SPLIT = "1" INHIBIT_PACKAGE_STRIP = "1" EOF

Add an init script to run the lighthouse validator client on startup

cat < srcs/poky/meta-evm/recipes-nodes/lighthouse/init

BEGIN INIT INFO

Provides: lighthouse

Required-Start: $remote_fs $syslog $networking

Required-Stop: $remote_fs $syslog

Default-Start: 2 3 4 5

Default-Stop: 1

Short-Description: Start and stop the lighthouse daemon

END INIT INFO

PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin DAEMON=/usr/bin/lighthouse NAME=lighthouse_vc DESC="Validator Client" PIDFILE=/var/run/lighthouse_vc.pid

start() { echo -n "Starting $DESC: " mount -o remount,size=90% /var/volatile start-stop-daemon -S -p $PIDFILE -N -10 -b -a /bin/sh -- -c "exec ${DAEMON} \ vc \ --beacon-nodes https://www.lightclientdata.org/ \ 2>&1 | tee /tmp/log" echo "$NAME." }

stop() { echo -n "Stopping $DESC: " start-stop-daemon -K -x "$DAEMON" -p $PIDFILE echo "$NAME." } case "$1" in start) start ;; stop) stop ;; restart|reload) stop start ;; *) N=/etc/init.d/$NAME echo "Usage: $N {start|stop|restart|reload}" >&2 exit 1 ;; esac exit 0 EOF

Now add lighthouse

to the final image by adding it to the initramfs dependencies in the file srcs/poky/meta-confidential-compute/recipes-core/images/cvm-initramfs.bb

diff --git a/recipes-core/images/cvm-initramfs.bb b/recipes-core/images/cvm-initramfs.bb index 97bed6b..638cc02 100644 --- a/recipes-core/images/cvm-initramfs.bb +++ b/recipes-core/images/cvm-initramfs.bb @@ -5,7 +5,7 @@ first 'init' program more efficiently. core-image-tiny-initramfs doesn't \ actually generate an image but rather generates boot and rootfs artifacts \ that can subsequently be picked up by external image generation tools such as wic."

-CVM_DEPS = "busybox-mdev e2fsprogs-resize2fs parted init-ifupdown initscripts base-files base-passwd netbase busybox-udhcpd" +CVM_DEPS = "lighthouse busybox-mdev e2fsprogs-resize2fs parted init-ifupdown initscripts base-files base-passwd netbase busybox-udhcpd"

PACKAGE_INSTALL = "e2fsprogs-mke2fs ca-certificates sysvinit busybox-udhcpd dropbear ${CVM_DEPS} ${VIRTUAL-RUNTIME_base-utils} ${ROOTFS_BOOTSTRAP_INSTALL}"

Finally rebuild the image

make build

Different image flavours can now be found in this directory:

ls -l srcs/poky/build/tmp/deploy/images/tdx/ ... lrwxrwxrwx 2 ubuntu ubuntu 45 Jun 12 16:28 cvm-image-azure-tdx.rootfs.wic -> cvm-image-azure-tdx.rootfs-20240612162808.wic lrwxrwxrwx 2 ubuntu ubuntu 51 Jun 12 16:28 cvm-image-azure-tdx.rootfs.wic.qcow2 -> cvm-image-azure-tdx.rootfs-20240612162808.wic.qcow2 lrwxrwxrwx 2 ubuntu ubuntu 49 Jun 12 16:28 cvm-image-azure-tdx.rootfs.wic.vhd -> cvm-image-azure-tdx.rootfs-20240612162808.wic.vhd ...

Run the VM locally

cd srcs/poky source oe-init-build-env ln -s "$PWD/tmp/work/x86_64-linux/qemu-helper-native/1.0/recipe-sysroot-native/usr/bin/qemu-system-x86_64" "$PWD/tmp/work/x86_64-linux/qemu-helper-native/1.0/recipe-sysroot-native/usr/bin/tdx"

runqemu cvm-image-azure wic nographic kvm ovmf qemuparams="-m 8G, -net nic,model=virtio -net user"

Run the VM in TDX

Make sure your on a TDX capable bare metal machine. Ubuntu 24.04 required.

git clone https://github.com/canonical/tdx

Follow the steps to setup the host

sudo ./tdx/setup-tdx-host.sh

Start the VM

TD_IMG=$PWD/srcs/poky/build/tmp/deploy/images/tdx/cvm-image-azure-tdx.rootfs.wic.qcow2 ./tdx/guest-tools/run_td.sh

Connect to the VM. SSH is enabled for development reasons, but no access should be allowed in a production environment.

ssh -p 10022 root@localhost

Verify lighthouse is running

ps | grep lighthouse tail -f /tmp/log