Create an AMI

This module provides instructions to create an Amazon Machine Image (AMI).

Create an AMI

The EC2 hosts on which the XRd runs must be configured to meet XRd's requirements.

To ensure high XRd Router performance and maintain consistency across the cluster, you must generate an Amazon Machine Image (AMI) that meets all XRd's requirements. This AMI can then be employed as a template for launching all worker nodes within the cluster.

An AMI can be shared among all the worker nodes in a single region, so you can save time by following these instructions only once. You can then copy the AMIs to different regions.

Find the Base AMI

The base AMI that is used to create an AMI for XRd worker nodes must be Amazon EKS optimized Amazon Linux (based on Amazon Linux 2023). Amazon provides separate images for every supported version of Kubernetes.

You can find the latest AMI for each Kubernetes version in each region by running the following command.

```
aws ssm get-parameter \
--name "<base AMI name>" \
--region <region> \
--query "Parameter.Value" \
--output text
```

Variables:

- `<base AMI name>`: Replace this with the full parameter name, e.g., `/aws/service/eks/optimized-ami/<k8s-version>/amazon-linux-2023/x86_64/standard/recommended/image_id`.
- `<region>`: Replace this with the AWS region you are targeting (e.g., `us-east-1`).

Note The Kubernetes version specified here must be the same as the version that was used to create the cluster.

Make a note of the AMI ID, `<al2023-ami-id>`.

For XRd Control Plane, you can use the base AMI with sysctl settings applied to the user data file when creating the worker node. For details, see the Create a Worker Node section.

Create an EC2 Instance

This EC2 instance must have access to the internet for installing packages and pulling in resources. If not, you must provide all these resources locally. Run the following command to create an EC2 instance.

```
aws ec2 run-instances --image-id <al2023-ami-id> --count 1 \
  --instance-type m6in.16xlarge \
  --key-name <key-pair-name> \
  --security-group-ids <security-group-ids> \
  --subnet-id <subnet-id>
```

You must specify the security group and subnet IDs so that the EC2 node has access to the internet. These are not necessarily the security group and subnet IDs created earlier. `<key-pair-name>` is the name of the key-pair which you must have created earlier.

Make a note of the instance ID returned from the command, `<ami-instance-id>`.

When the EC2 instance is created, connect to the EC2 instance over SSH using the specified SSH key and the username, `ec2-user`.

Copy XRd Resources

A set of resources on the XRd is available on the EKS public GitHub at xrd-packer.

Note HashiCorp Packer is a tool used for building virtual machine images, and the **xrd-packer** GitHub repository provides resources and Packer templates to create an AMI suitable for XRd vRouter. The manual steps described in this document use the resources from the repository as the source for some of the host OS settings, but the Packer tool is not used. The README file in the **xrd-packer** GitHub repository provides a description on how to build an AMI using Packer and the provided templates, as an alternative to the manual steps provided in this document.

Copy all of the resources under the **files** folder to the EC2 instance, mirroring the structure in the repository. For example,

- Copy `files/etc/modprobe.d/igb_uio.conf` to `/etc/modprobe.d/igb_uio.conf` on the EC2 instance
- Copy `files/etc/tuned/xrd-eks-node-variables.conf` to `/etc/tuned/xrd-eks-node-variables.conf` on the EC2 instance

If you have access to the internet, you can accomplish this task by running the following commands:

```
sudo dnf install -y git
git clone https://github.com/ios-xr/xrd-packer
sudo cp -r xrd-packer/files/* /
```

Clean up the unnecessary data by running the following commands:

```
sudo dnf remove -y git
rm -rf xrd-packer
```

Tuning

TuneD is a daemon for monitoring and tuning of system devices. The files copied in the previous step, contain a TuneD profile tailored for XRd hosts. This profile sets up some kernel parameters and tuning options to help XRd achieve high performance.

Note Install TuneD from source, because it is not included in the Amazon Linux 2023 repositories.

Perform the following steps to install TuneD from the source.

Install the prerequisite packages using the following commands:

```
sudo dnf install -y \
    dbus \
    ethtool \
    gawk \
    polkit \
    python-configobj \
    python-decorator \
    python-gobject \
    python-linux-procfs \
    python-perf \
    python-pyudev \
    util-linux \
    virt-what \
    make
```

Download the TuneD v2.24.1 source tarball using the following commands:

```
mkdir tuned
curl -L https://github.com/redhat-performance/tuned/archive/refs/tags/v2.24.1.tar.gz | tar
 -xz -C tuned --strip-components 1
```

Install TuneD using the following command:

```
sudo make -C tuned install
```

Take note that TuneD attempts to install a desktop file; skipping this causes the above **make** command to fail with the following output:

```
# desktop file
install -dD /usr/share/applications
desktop-file-install --dir=/usr/share/applications tuned-gui.desktop
make: desktop-file-install: Command not found
make: *** [install] Error 127
```

This is expected and can be ignored.

You can safely remove the TuneD sources afterward using the following command:

```
rm -rf tuned
```

After installing TuneD, specific files copied from the XRd resources necessitate adjustments for individual deployments. Update the file: /etc/tuned/xrd-eks-node-variables.conf with the appropriate isolated cores and required number of hugepages.

To deploy on a specific instance, the file must include the following configurations:

- m6in.16xlarge instance

```
isolated_cores=19-31
hugepages_gb=6
```

- m5[n].12xlarge instance

```
isolated_cores=11-23
hugepages_gb=6
```

- m5[n].24xlarge instance

```
isolated_cores=11-23
hugepages_gb=12
```

Note When running on an m5[n].24xlarge instance, the number of hugepages is doubled from the Cisco IOS XRd requirement. This is because the configured hugepages are distributed across all non-uniform memory access (NUMA) nodes in the system, whereas Cisco IOS XRd necessitates all its hugepages to be on the single NUMA node it operates on.

Lastly, initiate the TuneD service, and select the Cisco IOS XRd profile by executing the following command:

```
sudo systemctl start tuned
sleep 10
sudo tuned-adm profile xrd-eks-node
```

Note Running the `profile` command immediately after starting the service will prevent it from taking effect. Allow for a reasonable delay before running the `profile` command.

Boot Command Line Arguments

To ensure XRd functions correctly, configure the following command-line arguments:

```
grubby --update-kernel ALL --args "nohz_full=<isolated_cores> nohz=on \
    skew_tick=1 intel_pstate=disable tsc=reliable nosoftlockup \
    isolcpus=<isolated_cores> irqaffinity=<non-isolated cores> \
    hugepages=<hugepages_gb> rcu_nocbs=<isolated_cores> hugepagesz=1G \
    default_hugepagesz=1G rcupdate.rcu_normal_after_boot=1"
```

For `<isolated_cores>` and `<hugepages_gb>`, use the values from the TuneD section.

For `<non-isolated cores>`, specify all remaining cores, excluding the isolated ones. For example:

- `0-15` on m6in.16xlarge instance.
- `0-11, 24-47` on m5.24xlarge or m5n.24xlarge instances.

These arguments replicate those in the `realtime-virtual-guest` TuneD profile. The hugepage arguments have been added because the TuneD bootloader plugin cannot be used in Amazon Linux 2023.

Kernel Parameters

Setting kernel parameters (sysctl settings) on the host is either required or recommended when running Cisco IOS XRd containers. These parameters address the needs of both the XRd Control Plane and XRd vRouter platforms.

Replace the contents of `/etc/sysctl.conf` file with the following parameters:

```
fs.inotify.max_user_instances=65536
fs.inotify.max_user_watches=65536
kernel.randomize_va_space=2
net.core.rmem_max=67108864
net.core.wmem_max=67108864
net.core.rmem_default=67108864
net.core.wmem_default=67108864
net.core.netdev_max_backlog=300000
net.core.optmem_max=67108864
net.ipv4.udp_mem=1124736 10000000 67108864
```

Note `fs.inotify.max_user_watches=64000` - Indicates the maximum number of files you are allowed to watch for changes. You will get notified when there is a file change.

Configuring Core Dumping

Core files must be collected for detailed debugging. To prevent the node's disk from being exhausted, you must configure the system carefully. The recommended solution is to use the `systemd-coredump` tool.

To configure the kernel to use the `systemd-coredump` tool when a process crashes, include the following line to `/etc/sysctl.conf` file.

```
kernel.core_pattern=|/lib/systemd/systemd-coredump %P %u %g %s %t 9223372036854775808 %h
```

Copy the following contents to `/etc/systemd/coredump.conf`.

```
[Coredump]
Storage=external # (this is the default)
Compress=yes     # (this is the default)
MaxUse=32G       # (twice the sum of RAM given to XRd)
KeepFree=16G     # (the sum of RAM given to XRd)
ProcessSizeMax=32G  # (required if < v251 of systemd)
ExternalSizeMax=32G # (required if < v251 of systemd)
```

These values are applicable for a single XRd vRouter installation, described in Install XRd vRouter.

For the XRd Control Plane installation, described in Install XRd Control Plane, the copied values must be `MaxUse=12G` and `KeepFree=6G`.

In this configuration, `systemd-coredump` is configured to write the core files to `/var/lib/systemd/coredump` on the host.

Building and Installing the Interface Driver

You must not use the interface driver bundled with Amazon Linux 2023, as it does not support an ENAv2 interface feature.

Perform the following steps to build and install the interface driver that has Write Combining support.

1. Install the required packages

   ```
   sudo dnf install -y kernel-devel-$(uname -r) git
   ```

2. Clone the DPDK kernel module repository, and build `igb_uio`.

   ```
   git clone git://dpdk.org/dpdk-kmods
   cd dpdk-kmods
   git switch e721c733cd24206399bebb8f0751b0387c4c1595 --detach
   make -C linux/igb_uio
   ```

Note A 'Skipping BTF generation' warning is expected, it does not affect the performance of the built driver.

3. When the the driver is built, copy the interface driver kernel module into the appropriate location and register it.

   ```
   sudo cp linux/igb_uio/igb_uio.ko /lib/modules/"$(uname -r)"/kernel/drivers/uio
   sudo depmod -a
   cd ..
   ```

4. Clean up the packages.

   ```
   rm -rf dpdk-kmods
   sudo dnf remove -y kernel-devel-$(uname -r) git
   sudo dnf autoremove -y
   ```

5. Load the kernel module with the following:

   ```
   sudo modprobe uio
   sudo modprobe igb_uio wc_activate=1
   ```

6. Set the kernel module to load on boot, which is handled by copying the files earlier.

Reboot and Check

Reboot the EC2 instance to allow all the settings to take effect.

When the EC2 instance is rebooted, copy the `host-check` script from the **xrd-tools** repository at xrd-tools.

The `host-check` tool verifies if the host environment is configured correctly for running XRd containers. For more information on the usage of `host-check` tool, see the README file.

Run the `host-check` script.

This is sample output when you run the script for the XRd vRouter. You can run the script similarly for the XRd control plane.

```
[ec2-user@ip-172-31-3-90 ~]$ ./host-check -p xrd-vrouter
==============================
Platform checks - xrd-vrouter
==============================
PASS -- CPU architecture (x86_64)
 PASS -- CPU cores (64)
 PASS -- Kernel version (6.1)
 PASS -- Base kernel modules
         Installed module(s): dummy, nf_tables
 PASS -- Cgroups (v2)
 PASS -- Inotify max user instances
         65536 - this is expected to be sufficient for 16 XRd instance(s).
```

```
 PASS -- Inotify max user watches
         65536 - this is expected to be sufficient for 16 XRd instance(s).
 PASS -- Socket kernel parameters (valid settings)
 PASS -- UDP kernel parameters (valid settings)
 INFO -- Core pattern (core files managed by the host)
 PASS -- ASLR (full randomization)
 INFO -- Linux Security Modules (No LSMs are enabled)
 PASS -- Kernel module parameters
         Kernel modules loaded with expected parameters.
 PASS -- CPU extensions (sse4_1, sse4_2, ssse3)
 PASS -- RAM
         Available RAM is 239.5 GiB.
         This is estimated to be sufficient for 47 XRd instance(s), although memory
         usage depends on the running configuration.
         Note that any swap that may be available is not included.
 PASS -- Hugepages (6 x 1GiB)
 PASS -- Interface kernel driver
         Loaded PCI drivers: vfio-pci, igb_uio
 INFO -- IOMMU
         IOMMU not supported in AWS.
 PASS -- PCI devices
         The following PCI device(s) are available:
         ens5 (0000:00:05.0)
 PASS -- Shared memory pages max size (17179869184.0 GiB)
 PASS -- Real-time Group Scheduling (disabled in kernel config)
AWS host checks
-----------------------
 PASS -- Cmdline arguments
         Expected cmdline args found with isolated cores 19-31
==================================================================
Host environment set up correctly for xrd-vrouter
==================================================================
```

Delete the Cloud Folder

For the AMI to run the User Data passed into an EC2 instance, you must delete the `/var/lib/cloud` folder.

```
sudo rm -rf /var/lib/cloud
```

Take a Snapshot

Stop the EC2 instance, by running the following command:

```
aws ec2 stop-instances --instance-ids <ami-instance-id>
```

Note This command returns immediately, but it may take a minute for the container to be fully stopped.

To check if the EC2 instance is fully stopped, run the following command and ensure that the instance state is "stopped":

```
ec2 describe-instances --instance-ids <ami-instance-id> --query
'Reservations[0].Instances[0].State.Name
```

Once the EC2 instance has stopped, take a snapshot using the following command:

```
aws ec2 create-image --instance-id <ami-instance-id> --name xrd-vrouter-ami
```

Make a note of the AMI ID, `<xrd-ami-id>`.

Terminate the EC2 instance.

```
aws ec2 terminate-instances --instance-id <ami-instance-id>
```