Deploying a K3S cluster with meta-virtualization

Bruce Ashfield, Xilinx

Yocto Project Summit, 2021.11

Introduction

Agenda

- **Goals**
- **meta-virtualization**
  - K3S and container runtime support
- **Sample cluster**
  - infrastructure overview
  - image definitions
- **Application build and deployment**
- **Future work**

Goals

- **Not about a particular provisioning / deployment stack**
  - Details are NOT hidden behind multi-step tools
    - just “one simple command”!
- **Demonstrate sample / reference cluster creation**
  - Not expected to be production ready
  - Not optimized images, etc
- **Illustrate how meta-virtualization provides plumbing**
  - Image creation
  - Application build and deployment
- **3rd party K3S tutorials work/apply to our images**

Overview / Background

meta-virtualization: overview

- **Point of integration for ‘virtualization’ technologies**
  - Started June 2012
    - 1500+ commits made by 155+ contributors
  - VMs and containers
  - Core technology + support software
  - Many audiences: Bleeding edge and established tech
  - Baseline for creating OE derived virtualization solutions
- **K3S added in 2020**
  - From Rancher: for unattended, resource-constrained, remote locations or inside IoT appliances

Cluster setup

Test Cluster

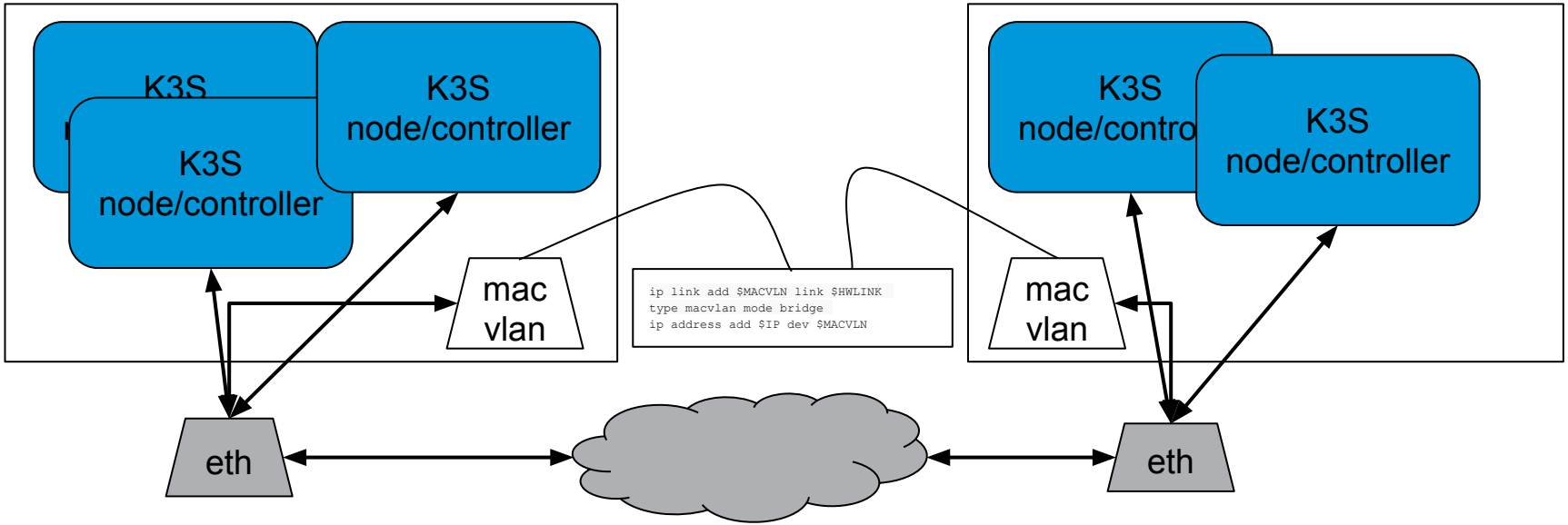
- **Requirements:**
  - Simple and virtualized
  - Full networking available (not slirp, not tap, etc)
  - Can be used outside of OE / yocto environment
  - Uses base images, created from meta-virt
    - No provisioning tools
    - Configuration works “out of the box”
- **Built from master (Nov 2021)**

Test Cluster: ‘physical’ networking

- **Not container overlay networking (that’s different)**
- **Requirements (remember machines are VMs)**
  - VMs and physical machines look the same
  - Direct VM -> physical adapter access
  - Full VM -> VM and VM -> host communications
  - Network wide addressable (i.e. common dhcp server)
  - No bridges or other complex routing / setup
  - Fast

Test Cluster: ‘physical’ networking

- **Solution: MAC vlan and MAC vtap**
  - Including the host on the mac vlan



Machine Launch

- **Virtual Machines**
  - ext4/raw images
  - launched from deploy, no bitbake interactions
  - script integrated with mac vtap
  - peripherals and shared storage attach are possible
  - fast and independent launch
  - ~2GB mem per machine

Images and Cluster Launch

Images

- **K3S Host and K3S Node**
  - Currently packagegroups and recipe RDEPENDS
    - flexibility
    - Future: image definitions in meta-virtualization
  - Default components
    - containerd, runc, etc
  - Very little customization required for basic cluster
- **Demo is via extension of core-image-minimal**
  - poky distro
  - local.conf for tweaks

K3S Controller and Node: Layers

- Same layers for both node types

```
BBLAYERS += " \
/opt/poky/meta-virtualization \
/opt/poky/meta-openembedded/meta-networking \
/opt/poky/meta-openembedded/meta-python \
/opt/poky/meta-openembedded/meta-oe \
/opt/poky/meta-openembedded/meta-filesystems \
/opt/poky/meta-openembedded/meta-perl \
/opt/poky/meta-security \
"
```

Local configuration

- **local.conf**
  - systemd
  - extra disk space
  - distro / image features
  - ssh (debug)
  - package management / PR Serve (optional)
  - Yocto Project BBHASH server (optional)

K3S host: Local configuration

```
CORE_IMAGE_EXTRA_INSTALL += " packagegroup-k3s-host kernel-modules"
CORE_IMAGE_EXTRA_INSTALL += " openssh"
CORE_IMAGE_EXTRA_INSTALL += " ca-certificates"

IMAGE_ROOTFS_EXTRA_SPACE = "2097152"
DISTRO_FEATURES:append = " seccomp"
DISTRO_FEATURES:append = " virtualization k8s"

IMAGE_FEATURES += "virt-unique-hostname"
IMAGE_FEATURES[validitems] += "virt-unique-hostname"

DISTRO_FEATURES:append = " systemd"
VIRTUAL-RUNTIME_init_manager = "systemd"
DISTRO_FEATURES_BACKFILL_CONSIDERED = "sysvinit"

IMAGE_FEATURES += " package-management"
PACKAGE_FEED_URIS="http://10.10.10.129/"
PACKAGE_FEED_BASE_PATHS = "rpm"
```

K3S node: Local configuration

```
CORE_IMAGE_EXTRA_INSTALL += " packagegroup-k3s-node kernel-modules"
CORE_IMAGE_EXTRA_INSTALL += " openssh"

IMAGE_ROOTFS_EXTRA_SPACE = "2097152"
DISTRO_FEATURES:append = " seccomp"
DISTRO_FEATURES:append = " virtualization k8s"

IMAGE_FEATURES += "virt-unique-hostname"
IMAGE_FEATURES[validitems] += "virt-unique-hostname"

DISTRO_FEATURES:append = " systemd"
VIRTUAL-RUNTIME_init_manager = "systemd"
DISTRO_FEATURES_BACKFILL_CONSIDERED = "sysvinit"

IMAGE_FEATURES += " package-management"
PACKAGE_FEED_URIS="http://10.10.10.129/"
PACKAGE_FEED_BASE_PATHS = "rpm"
```

K3S launch: controller

```
% cube-live.sh --macvtap --kernel ./bzImage ./core-image-minimal-qemux86-64.ext4

root@qemux86-64-7b:~# kubectl get nodes
NAME            STATUS   ROLES                  AGE   VERSION
qemux86-64-7b   Ready    control-plane,master   93s   v1.22.3-k3s1
root@qemux86-64-7b:~# uname -a
Linux qemux86-64-7b 5.14.15-yocto-standard #1 SMP PREEMPT Fri Oct 29 01:21:02 UTC 2021 x86_64 x86_64 x86_64 GNU/Linux

root@qemux86-64-7b:~# ip a s
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 86:16:9b:12:55:8d brd ff:ff:ff:ff:ff:ff
    inet 10.10.10.110/24 brd 10.10.10.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::8416:9bff:fe12:558d/64 scope link
       valid_lft forever preferred_lft forever
4: flannel.1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1450 qdisc noqueue state UNKNOWN group default
    link/ether 22:e3:3d:59:7a:80 brd ff:ff:ff:ff:ff:ff
    inet 10.42.0.0/32 scope global flannel.1
       valid_lft forever preferred_lft forever
    inet6 fe80::20e3:3dff:fe59:7a80/64 scope link
       valid_lft forever preferred_lft forever
5: cni0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1450 qdisc noqueue state UP group default qlen 1000
    link/ether e6:33:0e:3a:4d:8b brd ff:ff:ff:ff:ff:ff
    inet 10.42.0.1/24 brd 10.42.0.255 scope global cni0
       valid_lft forever preferred_lft forever
    inet6 fe80::e433:eff:fe3a:4d8b/64 scope link
       valid_lft forever preferred_lft forever
```

K3S launch: worker

```
% cube-live.sh --macvtap --kernel ./bzImage ./core-image-minimal-qemux86-64.ext4

root@qemux86-64-cb:~# uname -a
Linux qemux86-64-cb 5.15.0-yoctodev-standard #1 SMP PREEMPT Tue Nov 23 17:46:25 UTC 2021 x86_64 x86_64 x86_64 GNU/Linux

root@qemux86-64-cb:~# ip a s
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 1a:46:0b:ca:bc:7b brd ff:ff:ff:ff:ff:ff
    inet 10.10.10.190/24 brd 10.10.10.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::1846:bff:feca:bc7b/64 scope link
       valid_lft forever preferred_lft forever
3: sit0@NONE: <NOARP> mtu 1480 qdisc noop state DOWN group default qlen 1000
    link/sit 0.0.0.0 brd 0.0.0.0


root@qemux86-64-cb:~# systemctl status k3s-agent
* k3s-agent.service - Lightweight Kubernetes Agent
     Loaded: loaded (/lib/systemd/system/k3s-agent.service; disabled; vendor preset: disabled)
     Active: inactive (dead)
       Docs: https://k3s.io
```

K3S cluster join

```
root@qemux86-64-7b:~# cat /var/lib/rancher/k3s/server/token
K105b5997548fe4df9e74dfb53b3d7bf095a3f6aa7bd0ba1c504fe76f9bf1e4cee2::server:f7400dfc7310e7b498f7b6e1748d6556
```

```
root@qemux86-64-cb:~# k3s-agent -t K105b5997548fe4df9e74dfb53b3d7bf095a3f6aa7bd0ba1c504fe76f9bf1e4cee2::server:f7400dfc7310e7b498f7b6e1748d6556 -s
https://10.10.10.110:6443

Created symlink /etc/systemd/system/multi-user.target.wants/k3s-agent.service -> /lib/systemd/system/k3s-agent.service.

root@qemux86-64-cb:~# systemctl status k3s-agent
* k3s-agent.service - Lightweight Kubernetes Agent
     Loaded: loaded (/lib/systemd/system/k3s-agent.service; enabled; vendor preset: disabled)
    Drop-In: /etc/systemd/system/k3s-agent.service.d
             `-10-env.conf
     Active: active (running) since Fri 2021-11-26 14:59:32 UTC; 29s ago
```

```
root@qemux86-64-7b:~# kubectl label node qemux86-64-cb node-role.kubernetes.io/worker=worker

root@qemux86-64-7b:~# kubectl get nodes -o wide
NAME           STATUS   ROLES                  AGE     VERSION        INTERNAL-IP    EXTERNAL-IP   OS-IMAGE   KERNEL-VERSION            CONTAINER-RUNTIME
qemux86-64-7b  Ready    control-plane,master   34m     v1.22.3-k3s1   10.10.10.110   <none>        Poky       5.14.15-yocto-standard    containerd://1.5.5-11-g69e5db821.m
qemux86-64-cb  Ready    worker                 2m44s   v1.22.3-k3s1   10.10.10.190   <none>        Poky       5.15.0-yoctodev-standard  containerd://1.5.5-11-g69e5db821.m
```

Application build and deploy

Yocto + meta-virtualization: application build

- **meta-virtualization/recipes-demo**
- **Flask application**
  - GET on port 9000/yocto -> “Hello from Yocto!”
  - GET on port 9000/oe -> “Hello from OpenEmbedded!”
- **Includes deployment yaml**
  - as pod, or service
  - packaged in hello-world-deploy
- **OCI container application image**
  - sets entrypoint to the flask application

Flask App Recipe

```
DESCRIPTION = "Demo flask application"
HOMEPAGE = "https://yoctoproject.org"
LICENSE = "MIT"
LIC_FILES_CHKSUM =
"file://${COREBASE}/meta/COPYING.MIT;md5=3da9cfbcb7
88c80a0384361b4de20420"

SRC_URI = "file://flask-app \
           file://flask-app.yaml \
           file://flask-app-service.yaml"

DEPLOY_TYPE ?= "pod"
NAME ?= "demo"
APPNAME ?= "yocto-app"
CONTAINERNAME ?= "yocto-container"
CONTAINERIMAGE ?= "zeddii/app-container:latest"
CONTAINERPORT ?= "9000"
EXTERNALPORT ?= "10000"
```

```
do_install() {
    for tgt in flask-app.yaml flask-app-service.yaml; do
        sed -i 's%\@NAME\@%${NAME}%g' ${WORKDIR}/$tgt
        sed -i 's%\@APPNAME\@%${APPNAME}%g' ${WORKDIR}/$tgt
        sed -i 's%\@CONTAINERNAME\@%${CONTAINERNAME}%g' ${WORKDIR}/$tgt
        sed -i 's%\@CONTAINERIMAGE\@%${CONTAINERIMAGE}%g' ${WORKDIR}/$tgt
        sed -i 's%\@CONTAINERPORT\@%${CONTAINERPORT}%g' ${WORKDIR}/$tgt
        sed -i 's%\@EXTERNALPORT\@%${EXTERNALPORT}%g' ${WORKDIR}/$tgt
    done

    install -d ${D}${bindir}/
    install -m 755 ${WORKDIR}/flask-app ${D}${bindir}/

    install -d ${D}${sysconfdir}/deploy
    install -m 644 ${WORKDIR}/flask-app.yaml ${D}${sysconfdir}/
    install -m 644 ${WORKDIR}/flask-app-service.yaml ${D}${sysconfdir}/
}

RDEPENDS:${PN} += "python3-core python3-flask"
PACKAGES:prepend = "${PN}-deploy "
FILES:${PN}-deploy = "${sysconfdir}/*"
```

OCI container image recipe

```
SUMMARY = "Basic Application container image"
LICENSE = "MIT"
LIC_FILES_CHKSUM = "file://${COREBASE}/meta/COPYING.MIT;md5=3da9cfbcb788c80a0384361b4de20420"

include container-base.bb

OCI_IMAGE_ENTRYPOINT = "/usr/bin/flask-app"
CONTAINER_SHELL = "busybox"

IMAGE_INSTALL:append = "helloworld-flask"
```

```
% bitbake app-container

% skopeo copy --dest-creds zeddii:<> oci:app-container-qemux86-64-20211126153228.rootfs-oci:latest
docker://zeddii/app-container
Getting image source signatures
Copying blob 4dead1f63075 done
Copying config 3b84c96840 done
Writing manifest to image destination
Storing signatures
```

Application: deploy

```
apiVersion: v1
kind: Pod
metadata:
   name: @NAME@
spec:
   containers:
   - name: @CONTAINERNAME@
     image: @CONTAINERIMAGE@
     ports:
     - containerPort: @CONTAINERPORT@
```

```
root@qemux86-64-7b:~# dnf install helloworld-flask-deploy

root@qemux86-64-7b:~# kubectl apply -f /etc/flask-app.yaml
pod/demo created

root@qemux86-64-7b:~# kubectl label pod/demo new-label=yoctorule
pod/demo labeled
root@qemux86-64-7b:~# kubectl get ingress,svc,pods -o wide
NAME                 TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE    SELECTOR
service/kubernetes   ClusterIP   10.43.0.1    <none>        443/TCP   150m   <none>

NAME       READY   STATUS    RESTARTS   AGE   IP          NODE           NOMINATED NODE   READINESS GATES
pod/demo   1/1     Running   0          13m   10.42.1.4   qemux86-64-cb   <none>           <none>
```

Application: deploy

```
root@qemux86-64-7b:~# kubectl expose pod/demo  --port=9000  --target-port=9000 --type=LoadBalancer --name=yocto-greeter
service/yocto-greeter exposed
root@qemux86-64-7b:~# [ 9274.123931] IPv6: ADDRCONF(NETDEV_CHANGE): vetha24abbb3: link becomes ready
[ 9274.125629] IPv6: ADDRCONF(NETDEV_CHANGE): eth0: link becomes ready
[ 9274.132430] cni0: port 5(vetha24abbb3) entered blocking state
[ 9274.138588] cni0: port 5(vetha24abbb3) entered disabled state
[ 9274.140425] device vetha24abbb3 entered promiscuous mode
[ 9274.142444] cni0: port 5(vetha24abbb3) entered blocking state
[ 9274.143639] cni0: port 5(vetha24abbb3) entered forwarding state

root@qemux86-64-7b:~# kubectl get ingress,svc,pods -o wide
NAME                     TYPE           CLUSTER-IP    EXTERNAL-IP                 PORT(S)          AGE    SELECTOR
service/kubernetes       ClusterIP      10.43.0.1     <none>                      443/TCP          152m   <none>
service/yocto-greeter    LoadBalancer   10.43.4.191   10.10.10.110,10.10.10.190   9000:31458/TCP   9s     new-label=yoctorule

NAME                            READY   STATUS    RESTARTS   AGE   IP          NODE           NOMINATED NODE   READINESS GATES
pod/demo                        1/1     Running   0          15m   10.42.1.4   qemux86-64-cb   <none>           <none>
pod/svclb-yocto-greeter-rmt8t   1/1     Running   0          9s    10.42.1.5   qemux86-64-cb   <none>           <none>
pod/svclb-yocto-greeter-vbp4v   1/1     Running   0          9s    10.42.0.9   qemux86-64-7b   <none>           <none>
```

```
server [~]> curl -X GET http://10.10.10.110:9000/yocto/
Hello from Yocto!
server [~]> curl -X GET http://10.10.10.110:9000/oe/
Hello from OpenEmbedded!
```

3rd Party Application: nginx

```
root@qemux86-64-7b:~# kubectl apply -f https://raw.githubusercontent.com/myannou/k3d-demo/master/nginx.yaml
deployment.apps/nginx created
service/nginx created
root@qemux86-64-7b:~# kubectl expose deployment nginx  --port=8001 --target-port=80 --type=LoadBalancer --name=nginx-service
service/nginx-service exposed

root@qemux86-64-7b:~# kubectl get ingress,svc,pods -o wide
NAME                    TYPE           CLUSTER-IP      EXTERNAL-IP                   PORT(S)          AGE     SELECTOR
service/kubernetes      ClusterIP      10.43.0.1       <none>                        443/TCP          3h13m   <none>
service/yocto-greeter   LoadBalancer   10.43.4.191     10.10.10.110,10.10.10.190     9000:31458/TCP   41m     new-label=yoctorule
service/nginx           ClusterIP      10.43.209.215   <none>                        80/TCP           6m20s   app=nginx
service/nginx-service   LoadBalancer   10.43.39.209    10.10.10.110,10.10.10.190     8001:31980/TCP   6m6s    app=nginx
```

Future work

Future Work

- **Host and Node Image definitions**
  - Ability to vary CRI, networking, etc
  - Deployment tweaks
- **Application build and deploy templates**
- **Similar demo/tests for K8S**
- **Plug K3S servers into CI pipelines**

yocto PROJECT | THE LINUX FOUNDATION