Bridge vs Macvlan

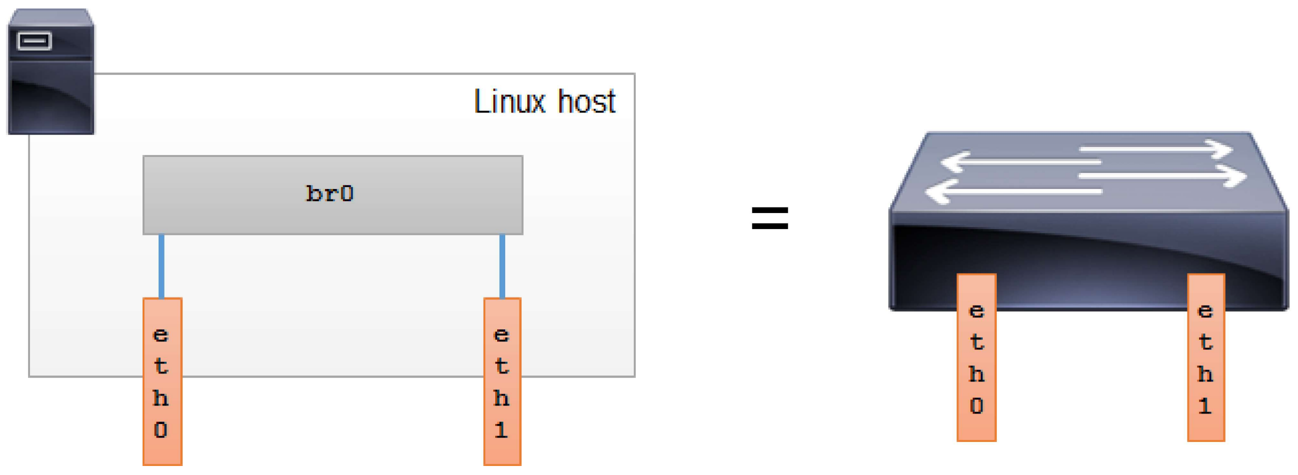
Bridge

A bridge is a Layer 2 device that connects two Layer 2 (i.e. Ethernet) segments together. Frames between the two segments are forwarded based on the Layer 2 addresses (i.e. MAC addresses). Although the two words are still often used in different contexts, **a bridge is effectively a switch** and all the confusion started 20+ years ago for marketing purposes.

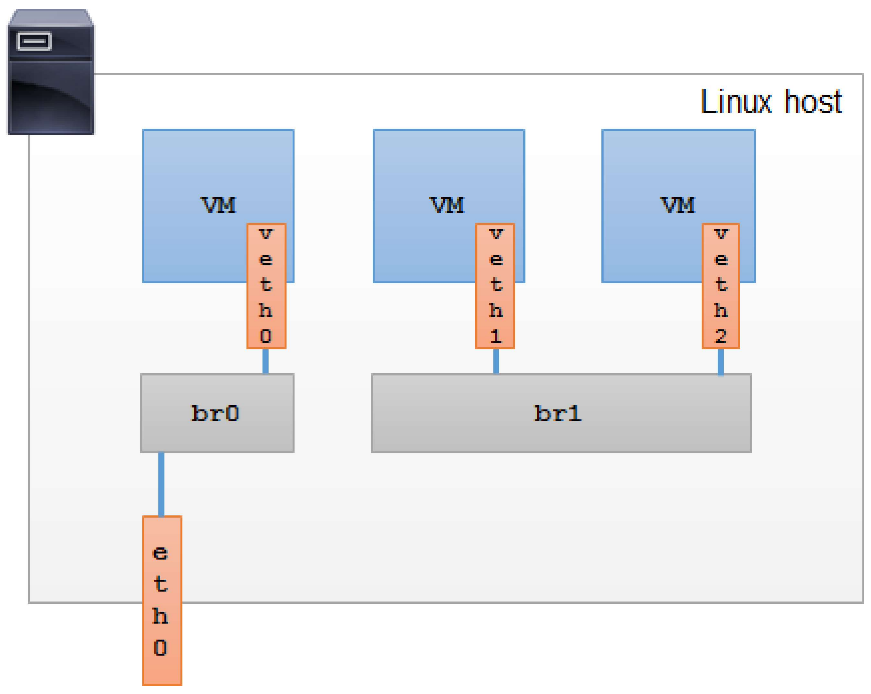
> Switching was just a fancy name for bridging, and that was a 1980s technology – or so the thinking went.

A bridge makes forwarding decisions based on the MAC address table. Bridge learns MAC addresses by looking into the Frames headers of communicating hosts.

A bridge can be a physical device or implemented entirely in software. Linux kernel is able to perform bridging since 1999. By creating a bridge, you can connect multiple physical or virtual interfaces into a single Layer 2 segment. A bridge that connects two physical interfaces on a Linux host effectively turns this host into a physical switch.



Switches have meanwhile became specialized physical devices and software bridging had almost lost its place. However, with the advent of virtualization, virtual machines running on physical hosts required Layer 2 connection to the physical network and other VMs. Linux bridging provided a well proven technology and entered it’s Renaissance. A bridge can connect virtual Ethernet interfaceswith each other or virtual Ethernet interfaces with a physical Ethernet device, connecting them into a single Layer 2 device.



You can verify the bridge configuration of your Linux host with the **brctl** utility which is part of the **bridge-utils** package on most distributions.

```
# brctl show
bridge name  bridge id          STP enabled  interfaces
br0          8000.080006ad34d1  no           eth0
                                             veth0
br1          8000.080021d2a187  no           veth1
                                             veth2
```

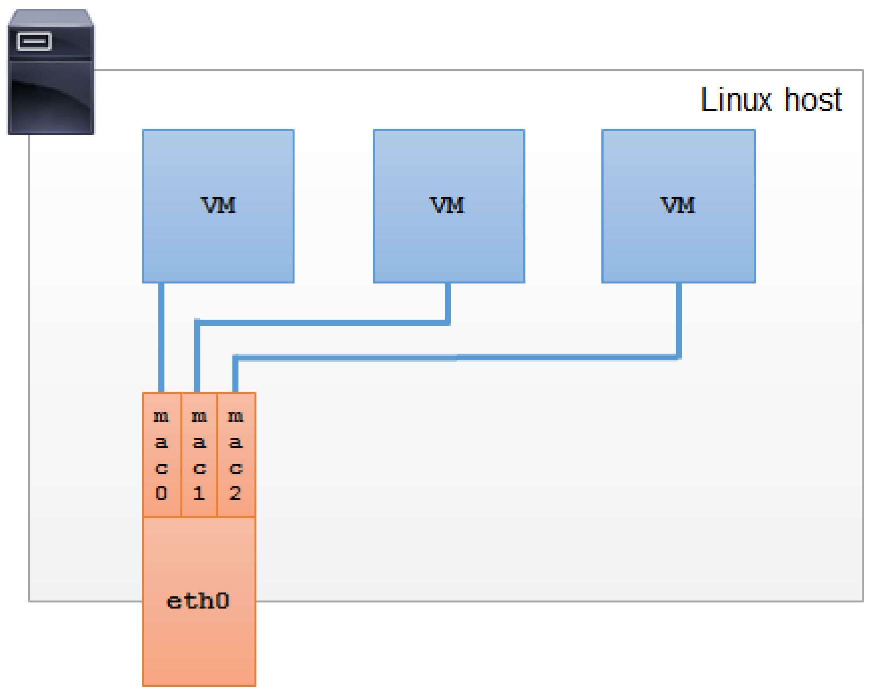
Bridges have potential to create Layer 2 loops, so you have an option to run STP if required.

Macvlan

Macvlan, MACVLAN or MAC-VLAN allows you to configure multiple Layer 2 (i.e. Ethernet MAC) addresses on a single physical interface. Macvlan allows you to configure sub-interfaces (also termed slave devices) of a parent, physical Ethernet interface (also termed upper device), each with its own unique (randomly generated) MAC address, and consequently its own IP address. Applications, VMs and containers can then bind to a specific sub-interface to connect directly to the physical network, using their own MAC and IP address.

Mavlan sub-interfaces are not able to directly communicate with the parent interface, i.e. VMs cannot directly communicate with the host. If you require VM-host communication, you should add another macvlan sub-interface and assign it to the host.

Macvlan sub-interfaces use a **mac0@eth0** notation, to clearly identify the sub-interface and it's parent interface. Sub-interface state is bound to its parent's state – if eth0 is down, so is the mac0@eth0.

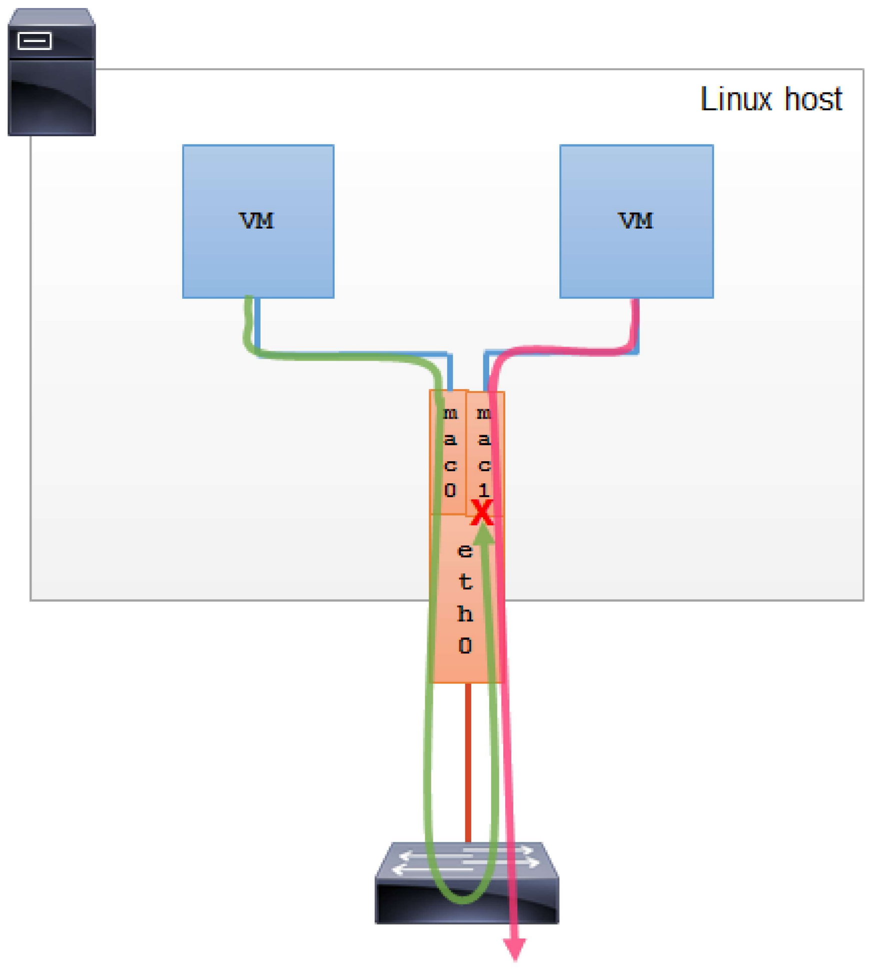


Macvlan modes

Each sub-interface can be in one of 4 modes that affect possible traffic flows.

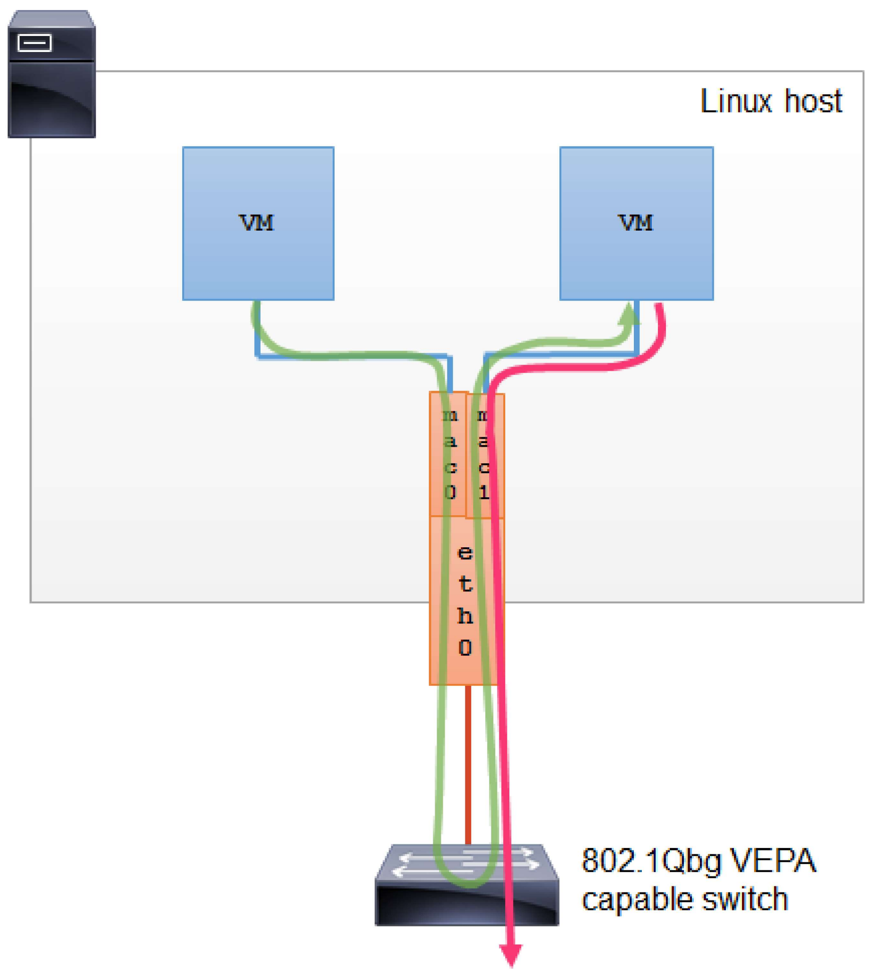
Macvlan private

Sub-interfaces on the same parent interface cannot communicate with each other. All frames from sub-interfaces are forwarded out through the parent interface. Even if physical switch reflects the frame sourced from one sub-interface and destined to another sub-interface, frame gets dropped.



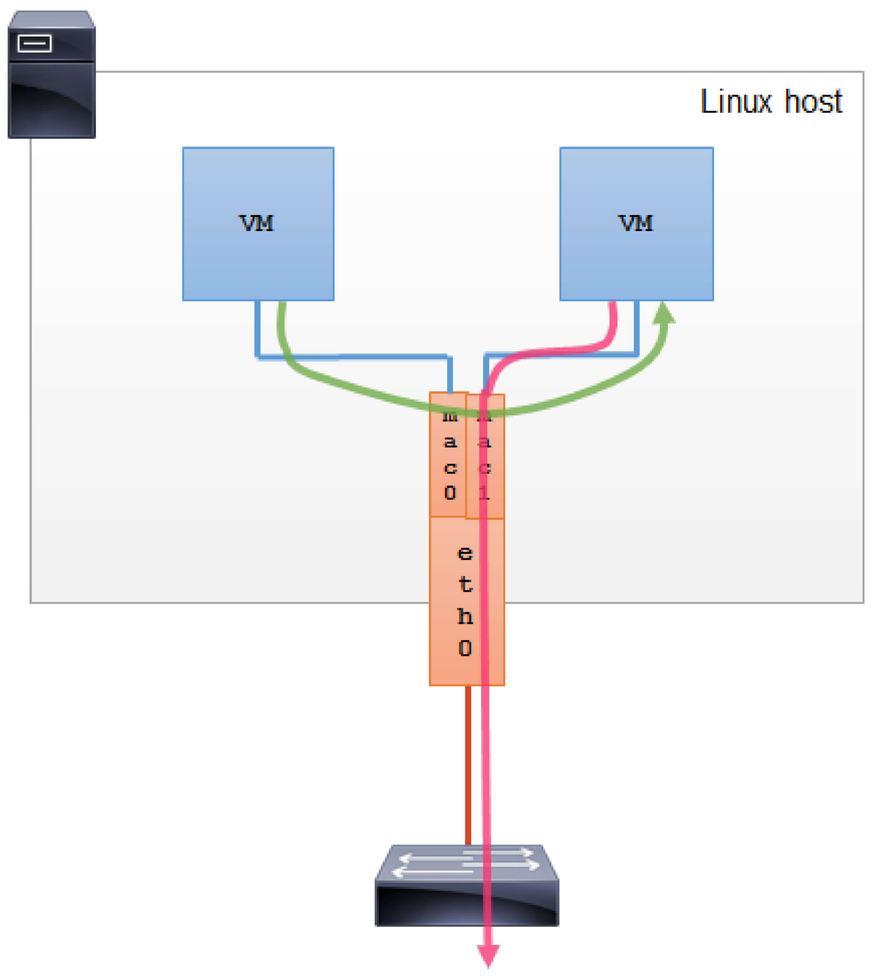
Macvlan VEPA

All frames from sub-interfaces are forwarded out through the parent interface. VEPA mode requires an IEEE 802.1Qbg aka Virtual Ethernet Port Aggregator physical switch. VEPA capable switch returns all frames where both source and destination are local to the macvlan interface. Consequently macvlan subinterfaces on the same parent interface are capable to communicate with each other through a physical switch. Broadcast frames coming in through the parent interface get flooded to all macvlan interfaces in VEPA mode. VEPA mode is useful when you are enforcing policies on physical switch and you want all VM-to-VM traffic to traverse the physical switch.



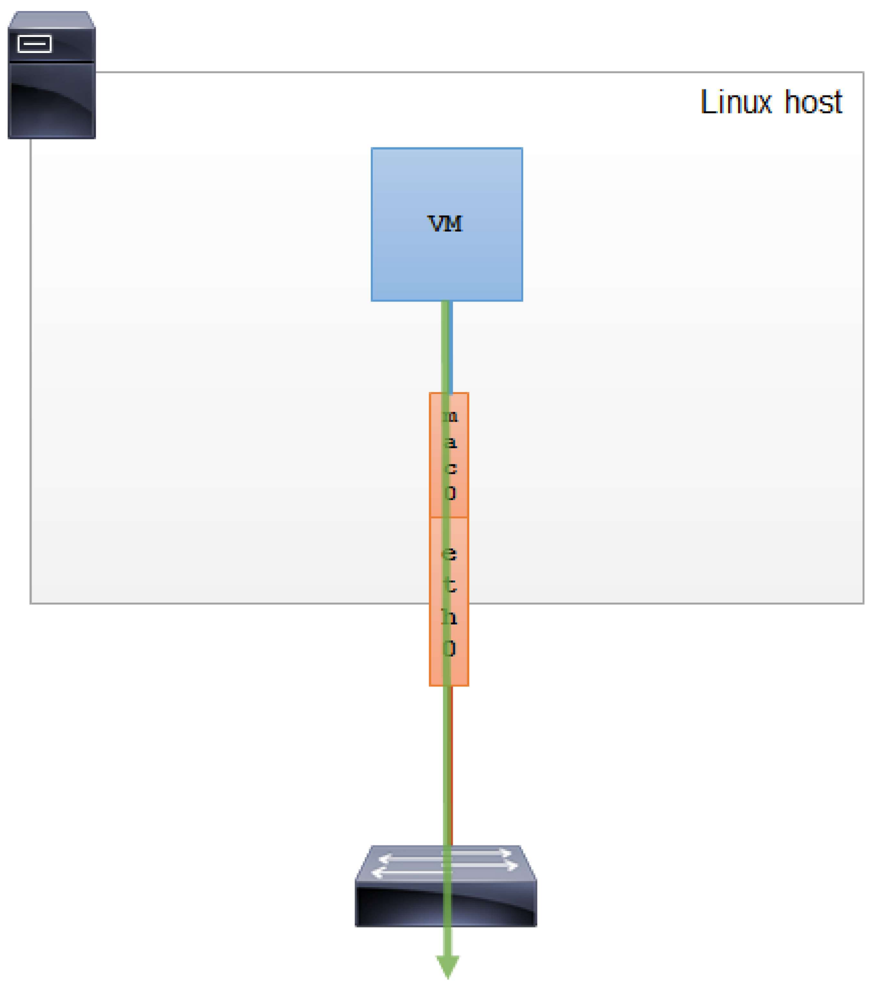
Macvlan Bridge

Macvlan connects all sub-interfaces on a parent interface with a simple bridge. Frames from one interface to another one get delivered directly and are not sent out. Broadcast frames get flooded to all other bridge ports and to the external interface, but when they come back from a VEP switch, they are discarded. Since all macvlan sub-interface MAC addresses are known, macvlan bridge mode does not require MAC learning and does not need STP. Bridge mode provides fastest communication between the VMs, but has a “flaw” you should be aware of – if parent interface state goes down, so do all macvlan sub-interfaces. VMs will not be able to communicate with each other when physical interfaces gets disconnected.



Macvlan Passthru

Allows a single VM to be connected directly to the physical interface. The advantage of this mode is that VM is then able to change MAC address and other interface parameters.



Macvlan vs Bridge

> The macvlan is a trivial bridge that doesn't need to do learning as it knows every mac address it can receive, so it doesn't need to implement learning or stp. Which makes it simple stupid and and fast.
>
> - Eric W. Biederman (http://article.gmane.org/gmane.linux.network/190237)

Use Macvlan:

- When you only need to provide egress connection to the physical network to your VMs or containers.
- Because it uses less host CPU and provides slightly better throughput (https://www.google.si/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=0ahUKEwiSiaXg69bLAhVDdCwKHY9KA_cQFggdMAA&url=http%3A%2F%2Fevents.linuxfoundation.org%2Fsites%2Fevents%2Ffiles%2Fslides%2FLinuxConJapan2014_makita_0.pdf&usg=AFQjCNFl2atyFB5aALHH5cKA8XkFqGQ1DQ&sig2=Nc5q1QsTrG9g8_YYRDrEog).

Use Bridge:

- When you need to connect VMs or containers on the same host.
- For complex topologies with multiple bridges and hybrid environments (hosts in the same Layer2 domain both on the same host and outside the host).
- You need to apply advanced flood control, FDB manipulation, etc.

Next

- Configure Docker macvlan network (http://hicu.be/docker-networking-macvlan-bridge-mode-configuration)
- Configure multiple Docker macvlan networks on 802.1Q trunk VLANs (http://hicu.be/docker-networking-macvlan-vlan-configuration)

Share this:

(https://hicu.be/bridge-vs-macvlan?share=facebook&nb=1) (https://hicu.be/bridge-vs-macvlan?share=linkedin&nb=1) (https://hicu.be/bridge-vs-macvlan?share=twitter&nb=1)

Tags: 802.1Qbg (https://hicu.be/tag/802-1qbg) brctl (https://hicu.be/tag/brctl) bridge (https://hicu.be/tag/bridge) bridge-utils (https://hicu.be/tag/bridge-utils) bridging (https://hicu.be/tag/bridging) container (https://hicu.be/tag/container) docker (https://hicu.be/tag/docker) ethernet (https://hicu.be/tag/ethernet) Ethernet interface (https://hicu.be/tag/ethernet-interface) frame (https://hicu.be/tag/frame) host (https://hicu.be/tag/host) interface (https://hicu.be/tag/interface) ip (https://hicu.be/tag/ip) l2 (https://hicu.be/tag/l2) layer 2 (https://hicu.be/tag/layer-2) linux (https://hicu.be/tag/linux) mac (https://hicu.be/tag/mac) MAC address (https://hicu.be/tag/mac-address) MAC addresses (https://hicu.be/tag/mac-addresses) macvlan (https://hicu.be/tag/macvlan) passthru (https://hicu.be/tag/passthru) physical (https://hicu.be/tag/physical) physical interface (https://hicu.be/tag/physical-interface) physical interfaces (https://hicu.be/tag/physical-interfaces) physical network (https://hicu.be/tag/physical-network) private (https://hicu.be/tag/private) spanning tree protocol (https://hicu.be/tag/spanning-tree-protocol) stp (https://hicu.be/tag/stp) sub-interface (https://hicu.be/tag/sub-interface) switch (https://hicu.be/tag/switch) switching (https://hicu.be/tag/switching) vepa (https://hicu.be/tag/vepa) virtual (https://hicu.be/tag/virtual) virtualization (https://hicu.be/tag/virtualization) vm (https://hicu.be/tag/vm)

Macvlan vs Ipvlan → (https://hicu.be/macvlan-vs-ipvlan)

9 comments for “Bridge vs Macvlan”

Pingback: Macvlan and IPvlan | Sreenivas Makam's Blog (https://sreeninet.wordpress.com/2016/05/29/macvlan-and-ipvlan/)

Pingback: Guide to Linux LXD containers - MYWEBTECH (https://www.mywebtech.blog/guide-linux-lxd-containers/)

Pingback: Guide to Linux LXD containers on Ubuntu 16.10 - MYWEBTECH (https://www.mywebtech.blog/guide-linux-lxd-containers-ubuntu/)

Pingback: Linux Containers LXD - MYWEBTECH (https://www.mywebtech.blog/linux-containers-lxd/)

Pingback: Why use Linux Containers LXD vs KVM? - MYWEBTECH (https://www.mywebtech.blog/linux-containers-lxd-vs-kvm/)

Pingback: linux 网络虚拟化： macvlan【zoues.com】 | zoues (http://www.zoues.com/2017/02/14/linux-%e7%bd%91%e7%bb%9c%e8%99%9a%e6%8b%9f%e5%8c%96%ef%bc%9a-macvlan%e3%80%90zoues-com%e3%80%91/)

David
August 12, 2017 at 00:54
great article!!!

Pingback: Configurazione VGA passthrough con libvirt e qemu-kvm | Un po' di tutto (https://fcel2008.wordpress.com/2017/10/04/configurazione-vga-passthrough-con-libvirt-e-qemu-kvm/)

Stewart Adam
November 30, 2017 at 01:44
I couldn’t find a contact address to write to you directly, so I’ll just leave this hear – I can’t underscore enough how excellent this blog is. I’ve been looking all over for details about macvlan and the implementation details and you really cut to the chase with the posts on your blog here. Just wanted to say thanks!

Comments are closed.