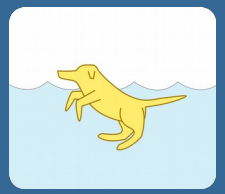
Can you hear me now?

Networking for containers



Jay Coles
DOGER.IO
2016

anchor

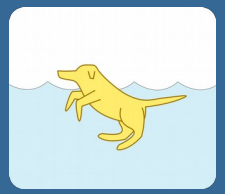
@container_doge
code@pocketnix.org

Who am I

- Face behind doger.io
- Been playing with containers in mainline since 2010
- Worked with openVZ before this
- Original implementation was a bet that I could write more user friendly tools than openVZ

Who is this for

- Will mainly be focusing on what you need to implement your own LXC or docker implementation
- People interested in networking
- Not going to have time to dive deep, just enough to get you pointed in right direction

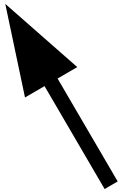


What we will be covering

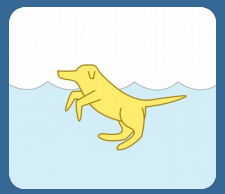
- Bridges
- VLANS
- Dummy interfaces
- Bonded interfaces
- MACVLAN
- MACVLAN/tap
- VETH
- VXLAN
- SDN

What we will be covering

- Bridges
- VLANS
- Dummy interfaces
- Bonded interfaces
- MACVLAN
- MACVLAN/tap
- VETH
- VXLAN
- SDN



Assumed knowledge, only briefly touched on

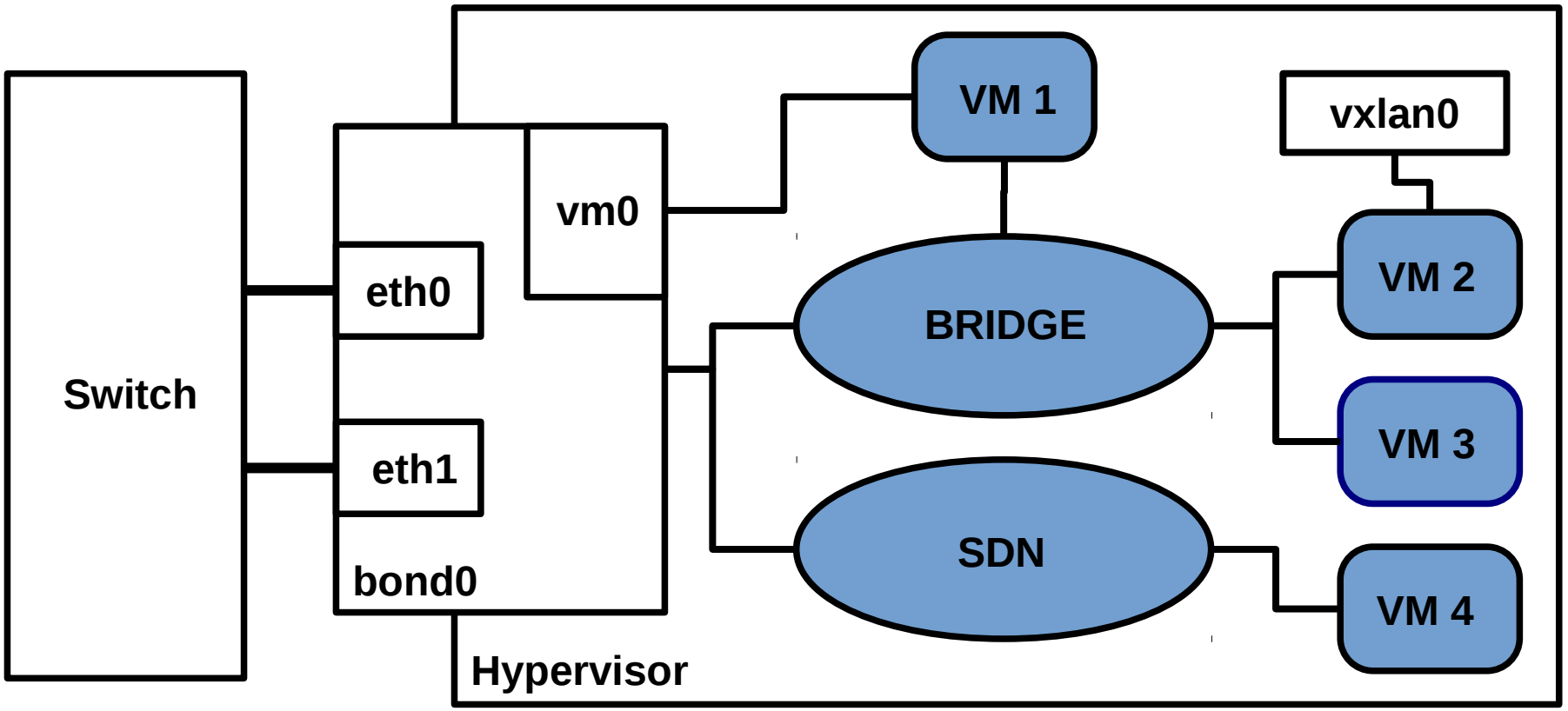


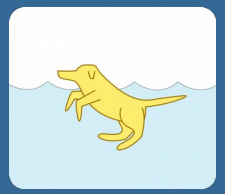
Jay Coles
DOGER.IO
2016

anchor

@container_doge
code@pocketnix.org

How it goes together





Jay Coles
DOGER.IO
2016

@container_doge
code@pocketnix.org

Bonding

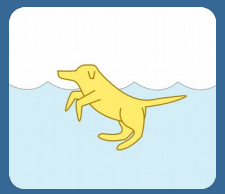
- Join 2 or more links into one logical link
- Speed increases (with caveats)
- High Availability (ability to lose one or more links and continue working)
- Typically use LACP (802.11ad) with switches

VLANs

- Split a switch or group of Physical switches into smaller Logical switches
- Useful for segregating clients (not security in itself but can be used to provide security)
- Comes in 'tagged' and 'untagged' form
- VLAN can spawn multiple switches with 'trunking'

Dummy Interfaces

- Mainly used for Linux routers
- Provide 'Service' addresses
- Integrates well with OSPF and BGP
- Ensure a service remains available even if the link is down
- Intent clearer than adding IP addresses to loopback interface, Better visibility.



Jay Coles
DOGER.IO
2016

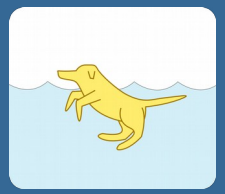
@container_doge
code@pocketnix.org

Bridge

- Emulate a L2 Switch in software (the box you plug the cables into)
- Basis of most Virtualisation and container setups (libvirt, docker)
- Virtual networks plug into this virtual switch
- Can be connected to a real network

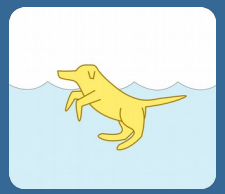
VETH

- The cable, Packets go in one end, come out the other
- When mixed with bridges, packets may be processed multiple times
- 'link detection' detects if other end is up or down (use to confirm if a container is 'ready')
- Simple networking: One end in a container, one end in a bridge



MACVLAN

- The other cables, takes an interface and splits it up
- Comes in 2 versions (macvlan/tap)
- High performance (possible to reduce the amount of times packets go via network stack)
- May fall back to software processing if many created on same interface (~10 depends on hardware)



Jay Coles
DOGER.IO
2016

anchor

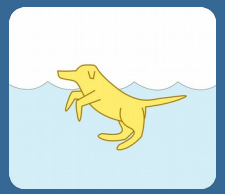
@container_doge
code@pocketnix.org

Overlay Networks

- Lets you have a network layout that looks different from what it is implemented on
- Easier time with migration of resources
- Lower friction for provisioning networks compared to vlans

VXLAN

- A cable cut in half that still works somehow
- Joins multiple machines together as if they where directly connected (Layer 2 network)
- Unlike normal VLANs can go over the internet
- Unlike GRE tunnel can connect more than 2 machines together
- 24 bits for vlan id compared to normal vlans 12 bits

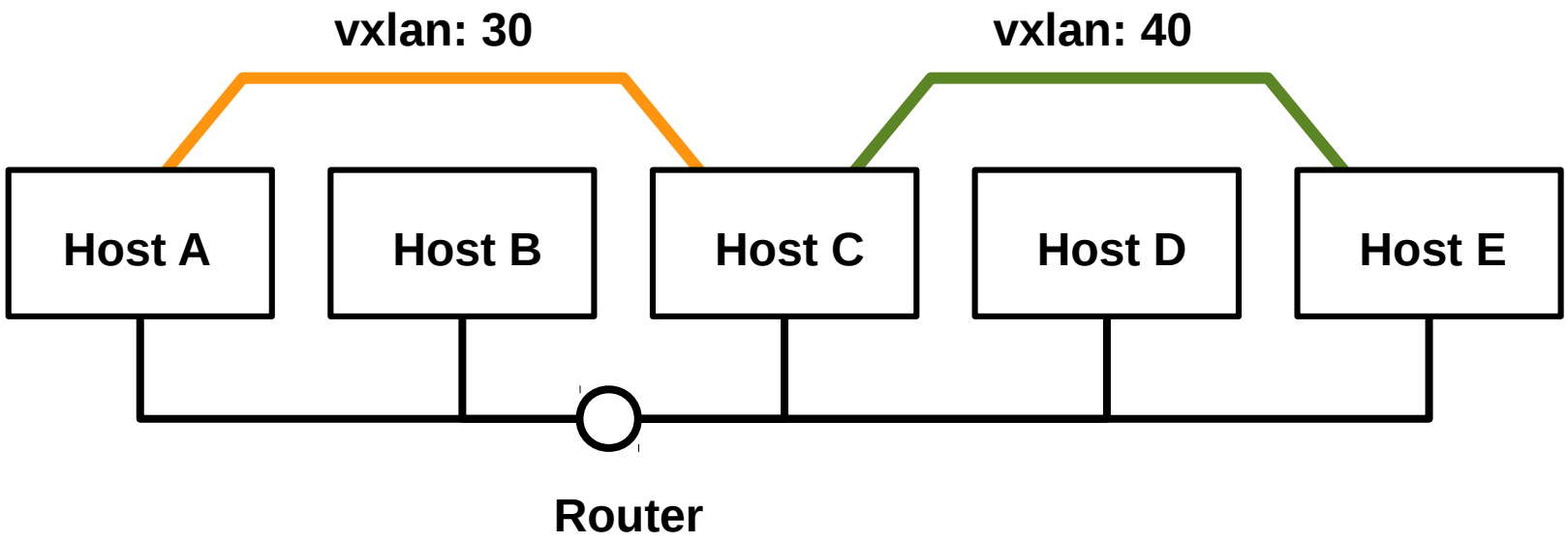


Jay Coles
DOGER.IO
2016

anchor

@container_doge
code@pocketnix.org

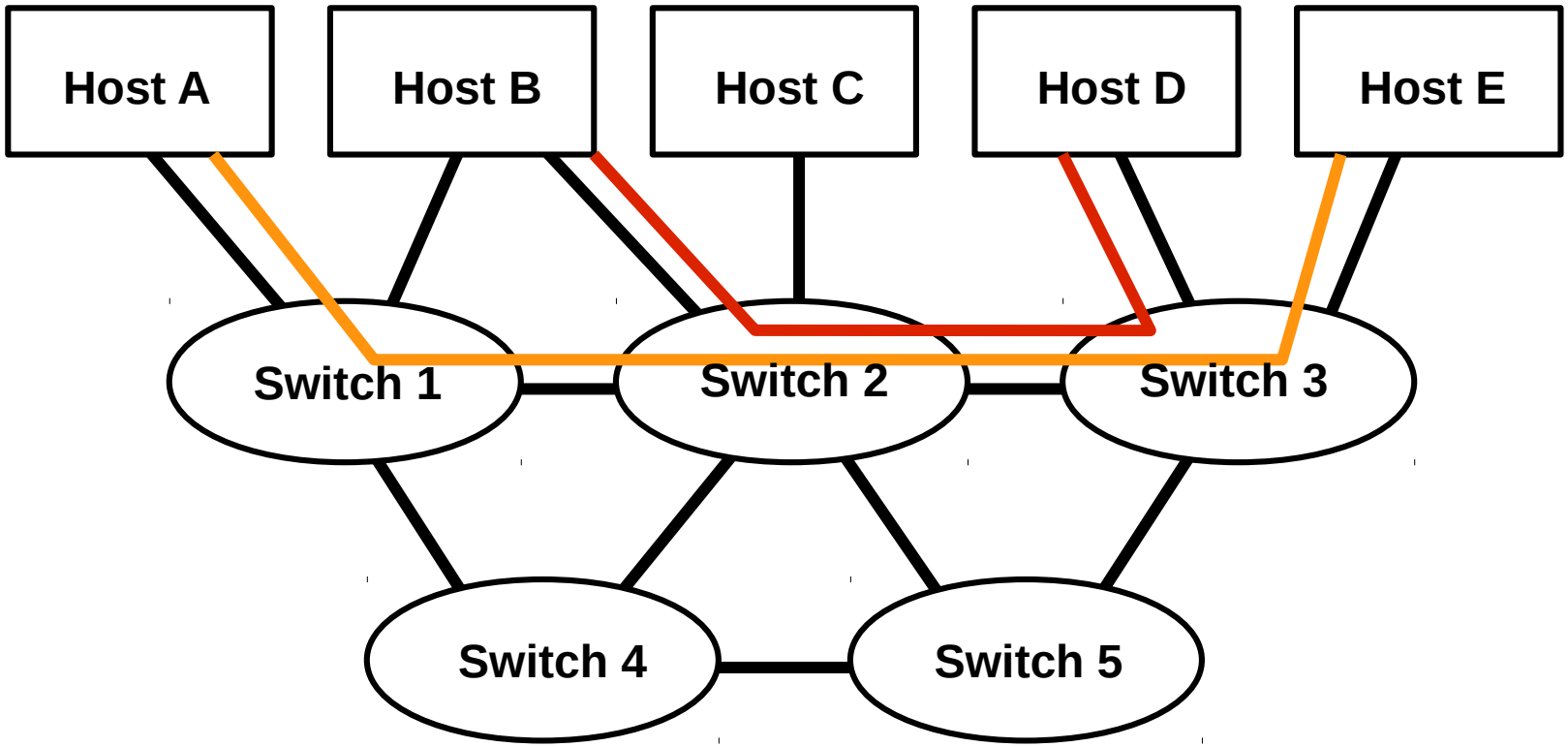
How it goes together

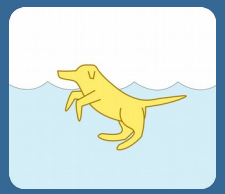


SDN

- Software processes controlling the flow of packets through your network
- Can switch traffic around congested links
- Can provide isolation/VLAN like functionality
- If you can think of it, you can likely do it

How it goes together



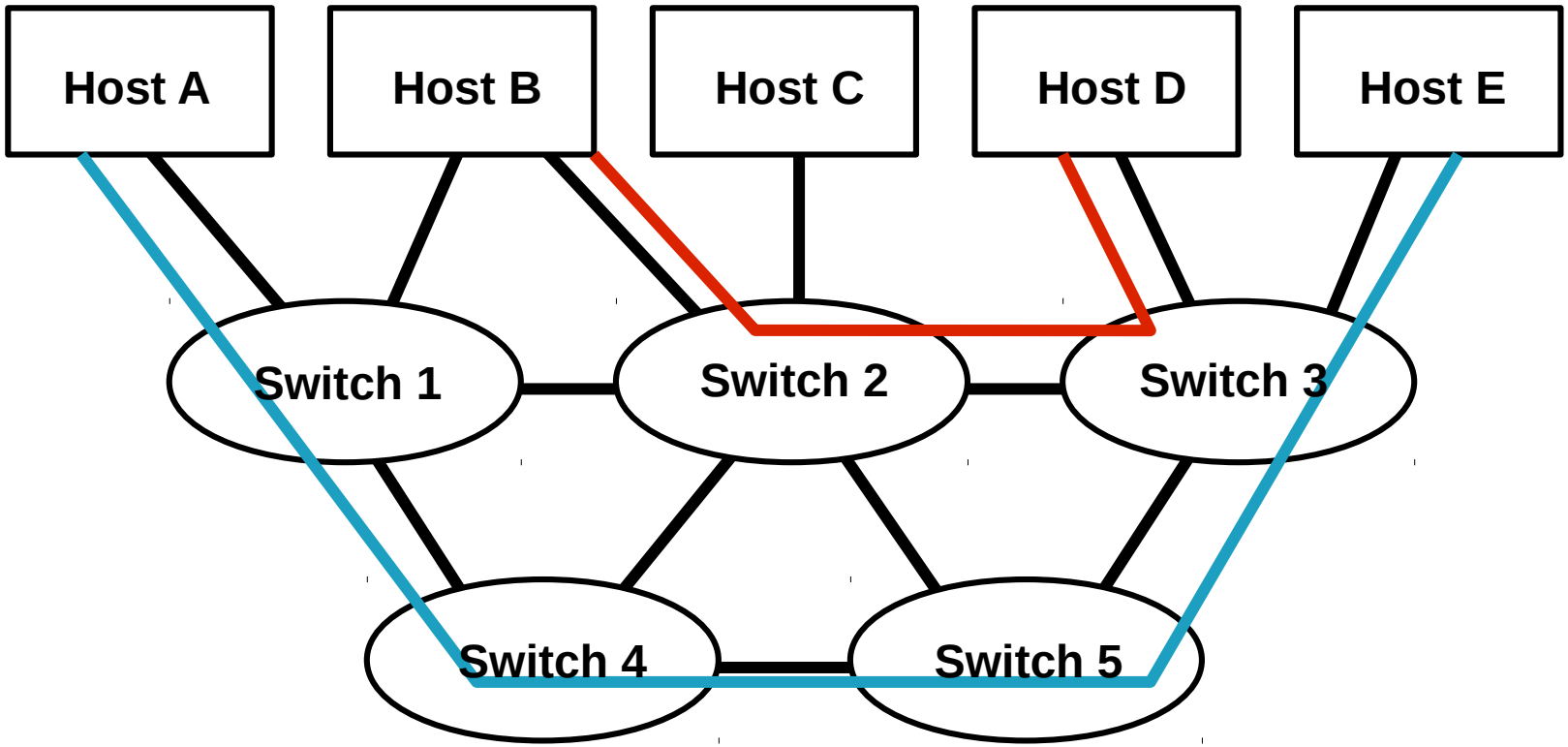


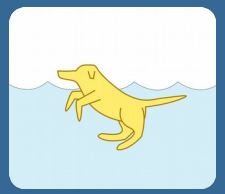
Jay Coles
DOGER.IO
2016

anchor

@container_doge
code@pocketnix.org

How it goes together



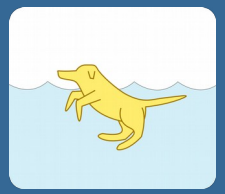


Jay Coles
DOGER.IO
2016

@container_doge
code@pocketnix.org

OpenFlow

- Specification to talk to SDN Switches/Machines
- Open protocol, easy to obtain specs
- Allows mixing and matching of Hardware devices and Software Control Planes
- Widely implemented
- Start looking the other way if either half does not implement it (vendor lock-in)



Jay Coles
DOGER.IO
2016

anchor

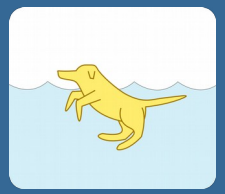
@container_doge
code@pocketnix.org

OpenVSwitch

- Shows up as a bridge device
- Has a userspace daemon to control packet flow
- Daemon installs rules in kernel openvswitch table for device on demand
- Kernel routes directly if it has a rule
- Can speak openflow, coordinate multiple machines

Roll your own?

- Will work fine on 'perfect' network (ie localhost)
- Worst case is normally no traffic rather than degraded performance
- Odd hard to diagnose issues in production (TCP in TCP, mtu issues)
- Easier to let someone else do the hard work and support it (linux kernel)



Questions? T-Shirts?

(yes we have T-Shirts now)

More info at www.pocketnix.org and doger.io

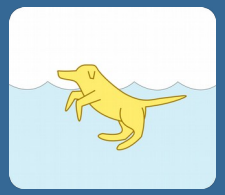
Bonus Points

- Mess with your coworkers
- Binary in interface names
- UTF-8 Extended chars
- Hardcoded color interfaces!

Bonus Points

```
root@tsurugi:/home/dablitz# ip li add '(ﾉ°_°)ﾉ' type dummy
root@tsurugi:/home/dablitz# ip li sh '(ﾉ°_°)ﾉ'
64: (ﾉ°_°)ﾉ: <BROADCAST,NOARP> mtu 1500 qdisc noop state DOWN mode DEFAULT group default
    link/ether 32:6a:f2:05:42:3d brd ff:ff:ff:ff:ff:ff
root@tsurugi:/home/dablitz#
```

```
root@tsurugi:/home/dablitz# echo `tput setaf 14`⚓`tput setaf 7`
⚓
root@tsurugi:/home/dablitz# IFACE="`tput setaf 14`⚓`tput setaf 7`"
root@tsurugi:/home/dablitz# ip li add "$IFACE" type dummy
root@tsurugi:/home/dablitz# ip li sh "$IFACE"
65: ⚓: <BROADCAST,NOARP> mtu 1500 qdisc noop state DOWN mode DEFAULT group default
    link/ether 86:c1:b6:e0:53:61 brd ff:ff:ff:ff:ff:ff
root@tsurugi:/home/dablitz#
```



Jay Coles
DOGER.IO
2016

anchor

@container_doge
code@pocketnix.org