Networking (S9)

Switching Routing

- Switching:
  - Switch connects host with interfaces in a network
    - *ip link*
- Routing:
  - Connect networks together
  - Router is also a device in the network
  - Gateway used as path to outside network
    - *route* to display routing table
    - *ip route add <ip> via <ip>*
    - *ip route add default via <ip>*
  - ip forwarding with /proc/sys/net/ipv4/ip_forward
    - /etc/sysctl.conf for persistent forwarding

DNS (Dynamic Name Service)

- Add known hosts to /etc/hosts with ip and hostname (old way)
  - *Hostname* to find out the systems hostname
  - Works for commands like *ping, ssh, curl, …*
- DNS Server resolves the names in modern day infrastructure
  - Add dns server to /etc/resolv.conf
  - Add public nameservers like 8.8.8.8 for internet servers
  - *nslookup <dns-name>* to see server address or *dig*
- Domain Names:
  - Separated with dots to group websites under subdomains
  - *search* entry in /etc/resolv.conf to look up for internal domain names

Network Namespaces

- processes in container have the same namespace
- host has own interfaces, routing tables, arp tables and can have its onw network inside
  - *ip netns add* to add namespace
  - link two namespaces with:
    - *ip link add <veth name1> veth peer name < veth name2>*
    - *ip link set <namespace> netns <namespace1/2>*
    - ip –n <namespace1/2> addr add <ip-addr1/2> dev <veth name1/2>
    - ip –n <namespace1/2> link set <veth name1/2> up
  - to create virtual network between multiple networks, create a virtual switch
    - *ip link add <virtual network name> type bridge*
    - *ip link set dev < virtual network name > up*
    - *ip link add <veth name1> veth peer name < bridge network name1>*
    - *ip link set ip link set <veth name> netns <namespace>*
    - ip link set <bridge network name1> master <virtual network name>
    - ip –n <namespace1/2> addr add <ip-addr1/2> dev <veth name1/2>
    - ip –n <namespace1/2> link set <veth name1/2> up

Docker Networking

- container can be run with network option --network
  - none: no network connection to the outside
  - host: container is connected to host network
  - bridge: internal network is created, where containers are connected to

CNI (Container Network Interface)

- set of standartds defining how programs should be developed to solve container networking
- already has a set of plugins available (BRIDGE, VLAN, IPVLAN, …)
- also third party plugins weaveworks, flannel, etc.
- docker has its own standards

Cluster Networking

- kubernetes applications/parts listen to different ports (see documentation)

Pod Networking

- kubernetes doesn't have built in solution
  - every pod can reach every pod in same node with a unique ip address
  - every pod should be able to communicate with every pod on other nodes without NAT
- many solutions available like wavenet and flannel
- you can also build your own solution with some shell scripting and virtual networks

CNI in Kubernetes

- see *cni with ps –aux | grep kubelet*   *ls /opt/cni/bin* and */etc/cni/net.d/10-bridge.conf*

CNI weave works

- uses agents on every node for address resolution and ip address assignment
- can be deployed as service/deamon or as pod
  - kubectl apply –f <weave_url with version>
- kubectl get pods –n kube-system

IP Address Management – Weave

- Manual: host-local plugin for ip address management
  - can be configured in net-script.conf
- weave creates range from 10.32.0.1 to 10.47.255.254 on default

Service Networking

- clusterIP only reachable inside cluster
- exposes port to outside
- kube-proxy creates rules for iptables, ipvs, userspace
  - cat /var/log/kube-proxy.log

DNS in Kubernetes

- deploys built in dns server by default
- url for services: web-service.apps.svc.cluster.local

CoreDNS in Kubernetes

- kubernetes deploys dns server as kube-dns
  - core dns after 12.1
  - deployed as pod's in cluster which runs coredns executable
  - service is also deployed
  - */etc/coredns/Corefile*
  - *Kubectl get configmap*

Ingress

- Layer 7 Load-Balancer built-in kubernetes cluster
- Deploys solution(ingress controller) and specifies rules for configuration(ingress resources)
- Ingress controller:
  - Possible solutions: GCE, nginx, …
  - Deployed as deployment with definition file, needs command to start
  - Pass in configmap object as easy configuration
  - POD_NAME and POD_Namespace as env variables
  - Also needs a NodePort service
  - Also a service account with roles, and rolebindings
- Ingress Resources:
  - Set of rules and configurations for the ingress controller
    - Forwarding, routing, …
  - Created with yaml file as ingress object
  - Kubectl get/describe ingress
  - Configure rules in definition file under spec > rules > http > paths
  - Default backend for wrong routes
  - Use host field to specifie the url