Software Platforms

LM in Computer Engineering

Massimo Maresca

From Development and Operations to Devops

- Software Development and Computing/Networking Operation Management used to live in distinct environment
- The evolution of Software Development technology from
  - Service Oriented Architecture (Information System Integration) to
  - Microservices (implementation of Information systems by means of a large number of simple services)

  leads to the integration between Software Development and Computer/Networking Operation Management
- Software Development and Operation Management converge to the "devops" discipline.

From Multitasking to Virtualization and Containers

- In the beginning: Single task computers
- 1st evolution: Real Time: Background vs Foreground
- 2nd evolution: Multiprogramming (Concurrency)
  - The process concept (Memory Allocation)
  - Process management (Process Control Block)
  - Process Status (Waiting, Ready, Running)
  - Process Scheduling
  - Operating System Services (Centralized I/O)
  - Time sharing vs. Real Time
- 3rd evolution: Multithreading
  - From processes to threads

From Multitasking to Virtualization and Containers

- 4^{th} evolution: Platforms - Service Execution Environments
  - Application Containers (e.g., servlet, service)
  - Standard interfaces and formats
  - Shared Container services
  - Application/Service Deployment and Management
- 5^{th} evolution: Virtual Machines
  - Replication of OS
  - Segregation of operation spaces
  - Perfect protection
- 6^{th} evolution: Containers
  - Linux Namespace technology
  - Docker

Linux namespaces

linux ha inventato la possibilita di spazi virtuali segregati in modo da avere sistemi separati, appaiono come una serie di piccole macchine.

prima avevamo una lista con tutti i processi operanti ora abbiamo varie liste ognuna per sottospazi

- The principle behind namespaces:
  - to create segregated Virtual Spaces in the Operating System,
  - to take advantage of Linux Multitasking
  - Only CPU sharing under OS control
  - No resource sharing (private process space, network space, filesystem space)
- From Multitasking, to Service Execution Environments, to VMs to namespaces
  - A process running in a segregated namespace appears as a sort of Linux Virtual Machine in that it is an autonomous Linux entity
  - It is not **under** the OS whereas it is **over** the OS
  - It leverages all the Linux features and capabilities

Experiment #1 : Direct Connection between namespaces

- Creation and configuration of two distinct network namespaces
- Connection between the two namespace through a direct virtual connection

creo due network namespace, che sembreranno due macchine diverse che noi connetteremo tra di loro

Experiment #1 : Direct Connection

ip netns del net1 &> /dev/null ip netns del net2 &> /dev/null	Clear Everything	
#		
ip netns add net1 ip netns add net2	Creating two namespaces: net1 and net2	
ip netns exec net1 ifconfig lo up ip netns exec net2 ifconfig lo up	Configuring local interfaces	
#		
ip link add veth1 type veth peer name veth2	Creating a link	creo un link tra veth1 e veth2 i links sono di layer 2
#		
ip link set veth1 netns net1 ip link set veth2 netns net2	Associating links to namespaces	associo i due veth ai corrispettivi network
ip netns exec net1 ifconfig veth1 10.0.15.1/24 up /24=8 bit per l'adress ip netns exec net2 ifconfig veth2 10.0.15.2/24 up	Associating IP Addresses to links and setting them up	configuro i link
#		
ip netns exec net1 ping 10.0.15.2	Testing connection through ping	

TEST COMMAND

./Direct_connection.sh

Experiment # 2: Bridged Connection between namespaces over a Virtual Bridge

- Creation and configuration of two distinct network namespaces
- Creation of a virtual bridge
- Connection between the two namespace through a virtual bridge

Experiment # 2: Bridged Connection over a Virtual Bridge

```
ip netns exec net1 ip link del veth1 >& /dev/null
ip netns exec net2 ip link del veth2 >& /dev/null
ip netns del net1 >& /dev/null
ip netns del net2 >& /dev/null
ip link del veth1-mybridge >& /dev/null
ip link de veth2-mybridge >& /dev/null
#
ip link del mybridge >& /dev/null
#
ip netns add net1
ip netns add net2
ip netns exec net1 ifconfig lo up
ip netns exec net2 ifconfig lo up
```

TEST COMMAND

./Bridged Connection.sh

```
ip link add mybridge type bridge
ip link set dev mybridge up
ip link add veth1 type veth peer name veth1-mybridge
ip link add veth2 type veth peer name veth2-mybridge
ip link set veth1-mybridge master mybridge
ip link set veth2-mybridge master mybridge
#
ip link set veth1 netns net1
ip link set veth2 netns net2
ip netns exec net1 ifconfig veth1 10.0.15.1/24 up
ip netns exec net2 ifconfig veth2 10.0.15.2/24 up
#
ifconfig veth1-mybridge up
ifconfig veth2-mybridge up
#
ip netns exec net1 ping 10.0.15.2
```

adesso creo un bridge

accendo il bridge

creo due link

adesso i link non arrivano direttamente alle macchine ma al bridge e poi pensa lui a consegnare

Experiment # 3: From Network Namespace to Internet

- Creation and configuration of a network namespace
- Connection between the network namespace and the “big Internet” through the “namespace host”, which in our case is the “guest” in the “host”

voglio accedere all'esterno (pingare un server esterno via nat) all'interno di un network namespace

Experiment # 3: From Network Namespace to Internet

nns.sh script

```
# Reset iptables
  iptables -P INPUT ACCEPT
  iptables -P FORWARD ACCEPT
  iptables -P OUTPUT ACCEPT
  iptables -F
  iptables -X
# Remove Network namespace ns1 and link v-eth1
  ip netns del ns1 &>/dev/null
  ip link del v-eth1 &>/dev/null
# Add namespace ns1
  ip netns add ns1
# Create v-eth1 and v-peer1
  ip link add v-eth1 type veth peer name v-peer1
# Move v-peer1 to ns
  ip link set v-peer1 netns ns1
# set v-eth1
  ip addr add 10.200.1.1/24 dev v-eth1
  ip link set v-eth1 up
# Set v-peer1 in the ns
  ip netns exec ns1 ip addr add 10.200.1.2/24 dev v-peer1
  ip netns exec ns1 ip link set v-peer1 up
# Set loopback interface in the ns
  ip netns exec ns1 ip link set lo up
# Add default route in the ns
  ip netns exec ns1 ip route add default via 10.200.1.1
# Set host routing tables
  iptables   -t nat
                   -A POSTROUTING
                    -s 10.200.1.0/24
                   -j MASQUERADE
# Enable routing in the host
sysctl -w net.ipv4.ip_forward=1
# External Ping
  ip netns exec ns1 ping 130.251.1.4
```

quest'indirizzo per accedere all'internet

TEST COMMAND:

```
./nns.sh
```

Experiment # 4: Network Namespace from Host and “Container”

- Program based namespace creation
- Connection to the “big Internet” through the namespace host

myc1.c – Example of clone ()

```
int main(int argc, char *argv[]) {
   char *stack;                   /* Start of stack buffer */
   char *stackTop;                /* End of stack buffer */
   pid_t pid;
   if (argc < 2) {
      fprintf(stderr, "Usage: %s <string>\n", argv[0]);
      exit(0);
   }
   stack = malloc(STACK_SIZE);
   stackTop = stack + STACK_SIZE;
   pid = clone(childFunc, stackTop, CLONE_NEWNET | SIGCHLD, (void *) argv[1]);
   printf("Child Pid: %d\n", pid);
   sleep(2);
   waitpid(pid, 0, 0);
   printf("Closing main\n");
}
```

```
int childFunc(void *arg) {
        printf("Start of child: arg = %s \n", (char *) arg);
        system((char *) arg);
        sleep(1);
        printf("End of child \n");
}
```

Experiment # 4: Network Namespace from Host and “Container”

Container Creation: myc1

gcc myc1.c –o myc1

./myc1

Network Namespace Configuration in Container (nnsc.sh)

ip link add v-peer1 type veth peer name v-eth1 netns 1

ip addr add 10.200.1.2/24 dev v-peer1

ip link set v-peer1 up

ip link set lo up

ip route add default via 10.200.1.1

Network Namespace Configuration in Host (nnsh.sh)

ip addr add 10.200.1.1/24 dev v-eth1

ip link set v-eth1 up

Set host routing tables

iptables -t nat
-A POSTROUTING
-s 10.200.1.0/24
-j MASQUERADE

Enable routing in the host

sysctl -w net.ipv4.ip_forward=1

Experiment

ip link del v-peer1 >& /dev/null
ip link del v-eth1 >& /dev/null
Create Container (./myc1 bash)
Configure Container (./nnsc.sh in Container shell)

Configure Host (./nnsh.sh in Host shell)
Ifconfig in Container (ifconfig in Container shell)
Ifconfig in host (ifconfig)
Ping from container (ping 130.251.1.4 from myc1 bash)

Experiment # 5: Joining a Network Namespace

- The setns() call

Experiment # 5: setns()

setns-pid.c

```
strcpy (spid, argv[1]);
strcat(nsname, "/proc/");
strcat(nsname, spid);
strcat(nsname, "/ns/net");
fd = open(nsname, O_RDONLY);
if (setns(fd, CLONE_NEWNET)==-1) {
    fprintf (stderr, "Error in setns\n");
    exit(-1);
}
system(argv[2]);
```

setns-name.c

```
strcpy (network_namespace_name, argv[1]);
strcat(nsname, "/var/run/netns/");
strcat(nsname, network_namespace_name);
fd = open(nsname, O_RDONLY);
if (setns(fd, CLONE_NEWNET)==-1) {
    fprintf (stderr, "Error in setns\n");
    exit(-1);
}
system(argv[2]);
```

Experiments

1: Activate container (./myc1 bash)
- From host shell: setns-pid <PID> bash
  - ifconfig
- From container shell: ifconfig lo up
- From host shell: ifconfig

2: Activate container (./myc1 bash)
- Associate nns name to ns process. From host shell:
  - ln -s /proc/<PID>/ns/net /var/run/netns/<nns-name>
- From host shell: setns-name <nns-name> bash
  - ifconfig
- From container shell: ifconfig lo up
- From host shell: ifconfig

Experiment # 6: Simple Container creation

- Wrapping up everything
- Creating a container

Experiment # 6: Container Activation

myc-cmd bash

Then see ifconfig to test ns

```
int childFunc(void *arg){
	printf("Start of child: arg = %s \n", (char *) arg);
	system("ip link add v-peer1 type veth peer name v-eth1 netns 1");
	system("ip addr add 10.200.1.2/24 dev v-peer1");
	system("ip link set v-peer1 up");
	system("ip link set lo up");
	system("ip route add default via 10.200.1.1");
	system((char *) arg);
	sleep(2);
	printf("End of child \n");
}
```

```
int main(int argc, char *argv[]) {
	char *stack;			/* Start of stack buffer */
	char *stackTop;			/* End of stack buffer */
	pid_t pid;
	stack = malloc(STACK_SIZE);
	stackTop = stack + STACK_SIZE;

	pid = clone(	childFunc, stackTop,
			CLONE_NEWNET | SIGCHLD,
			(void *) argv[1]);
	printf("Child Pid: %d\n", pid);
	sleep(1);
	system("ip addr add 10.200.1.1/24 dev v-eth1");
	system("ip link set v-eth1 up");
	system("iptables	-t nat
				-A POSTROUTING
				-s 10.200.1.0/24
				-j MASQUERADE");
	system("sysctl -q -w net.ipv4.ip_forward=1");
	waitpid(pid, 0, 0);
	printf("Closing main\n");
}
```

Experiment # 7: Simple Container creation

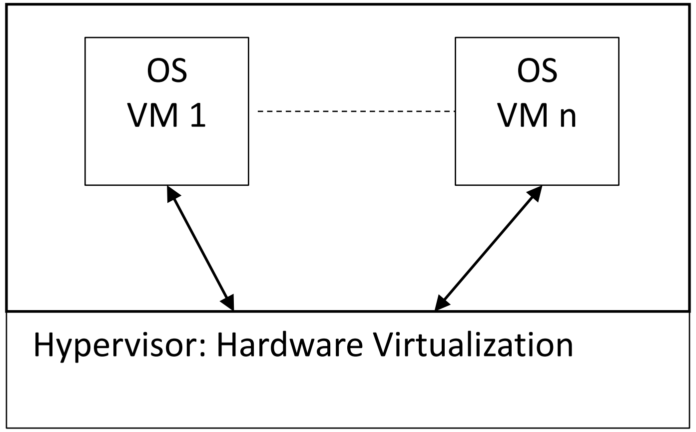
- Creating a container and running Tomcat inside the container

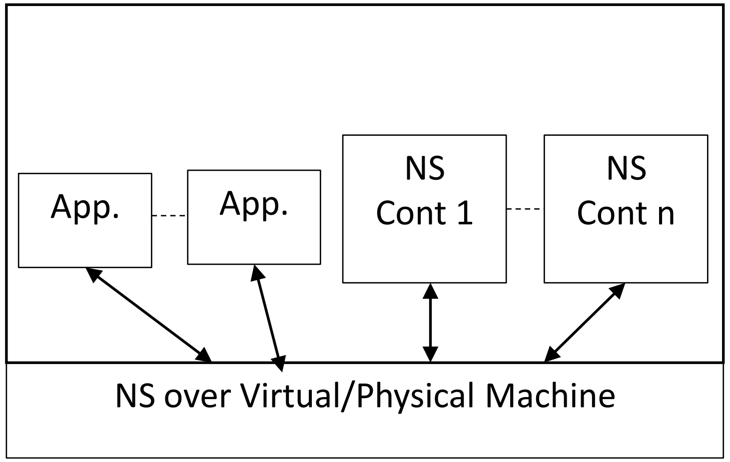
Experiment # 7 – A container for Tomcat

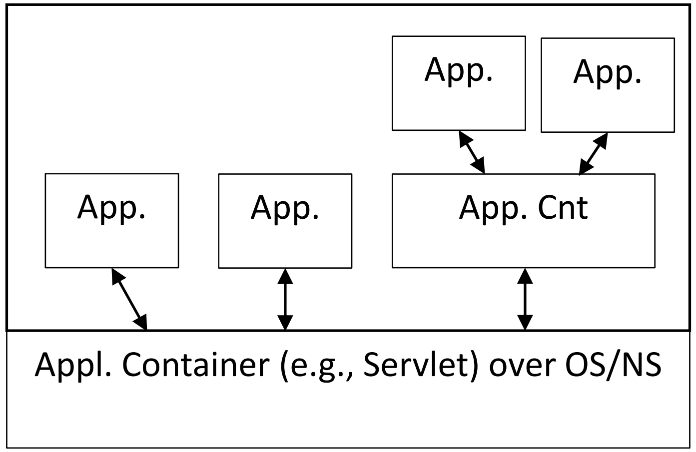
Experiments:

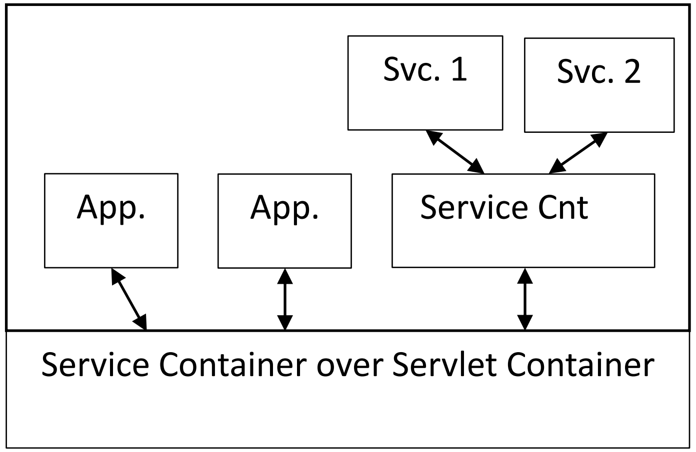
1. Run myc-cmd bash
   Then see ifconfig to test ns
2. Run myc-cmd <tomcat bin path>/startup.sh
   On host: ps au | grep tomcat
   curl 10.200.1.2:8080
3. From local Ubuntu browser open 10.200.1.2:8080
4. Then
   a. add Port Forwarding rule to Windows Vbox configuration:
      Host port 9095 <-> Guest port 9090
      VBoxManage controlvm “VM Name" natpf1 “<rule name>,tcp,,9095,,9090"
   b. add the following iptables rules in Ubuntu Guest:
      iptables -P INPUT ACCEPT
      iptables -P FORWARD ACCEPT
      iptables -P OUTPUT ACCEPT
      iptables -t nat -A PREROUTING -p tcp --dport 9090 -i enp0s3 -j DNAT --to 10.200.1.2:8080
   c. use Windows host browser to access 10.200.1.2:8080 through localhost:9095

Virtualization, Segregation: Platform Summary









Provisioning (i.e., releasing information systems)

- As a program running in OS (License based charging: Investment costs)
  - In a VM
  - In a Container
  - As a War
- Cloud based provisioning (Per-use based charging: Operation Costs)
  - Infrastructure as a service
  - Platform as a Service
  - Storage as a Service
  - Software as a Service (Web based access)