Docker daemon or Docker engine

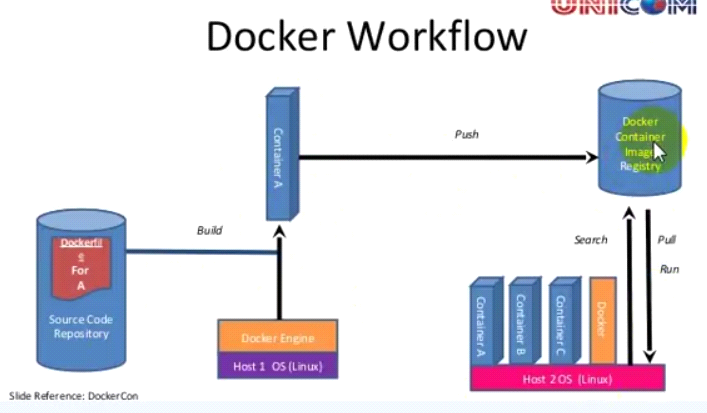
Only 1 host os…No Guest OS

Docker will create image. Image –OS file identifier/ template for environment.

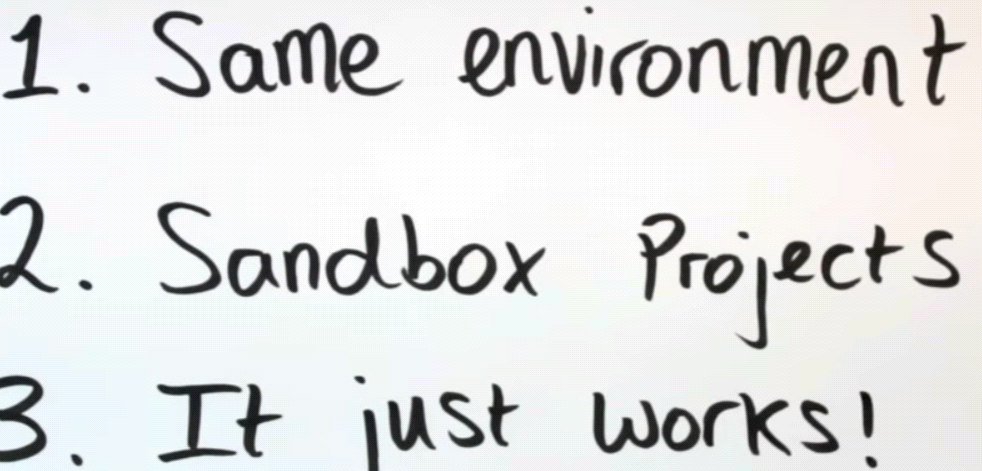
We can create many container from image.

* In VMware, for each VM we need one OS
* Cost very low

Pull-> Download

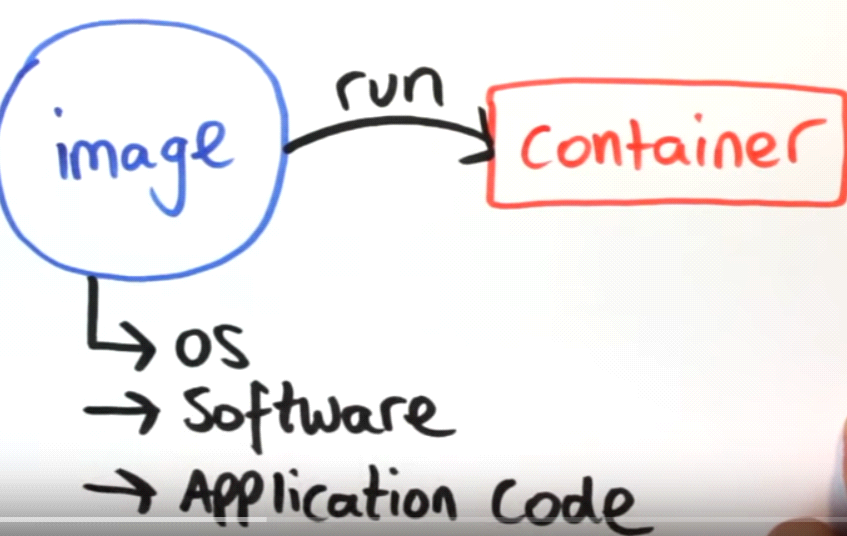


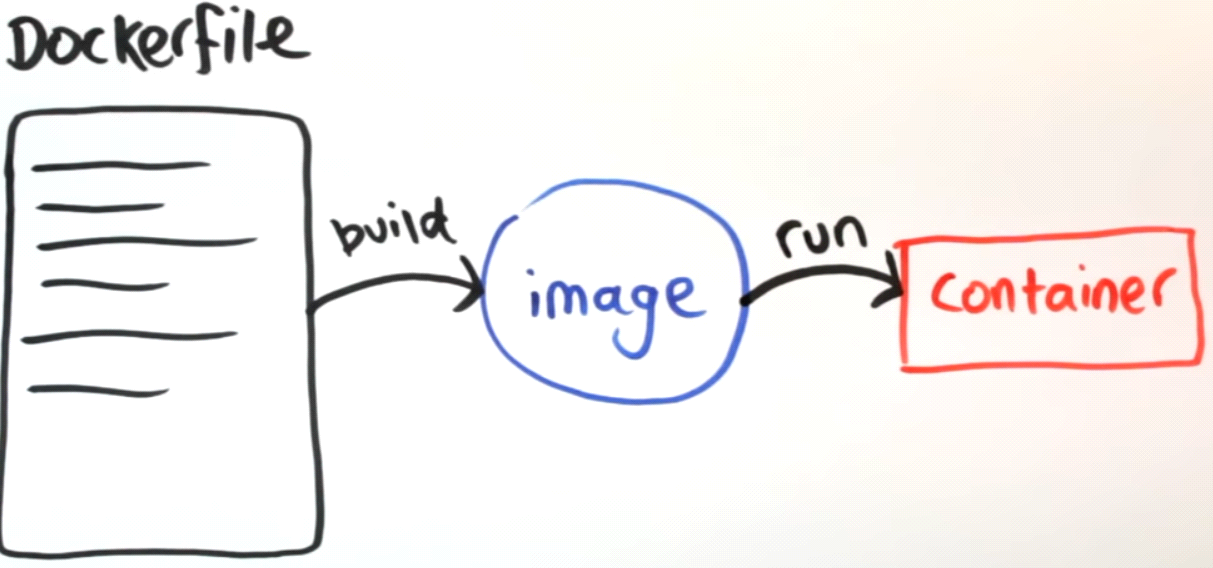
  
**Learn Docker in 12 Minutes 🐳**



Docker uses host machine kernel.

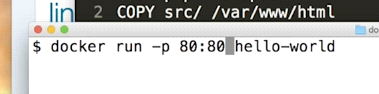
All linux distributions centos,Ubuntu..built on same kernel





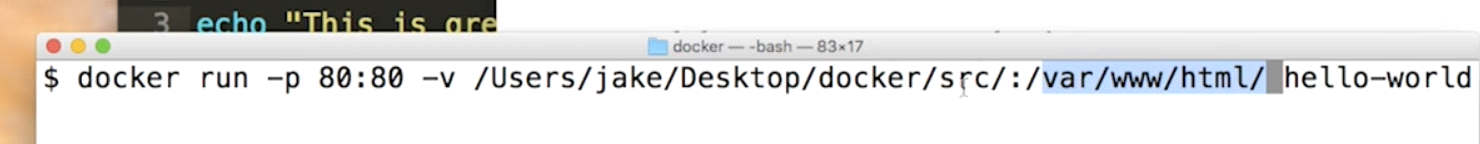


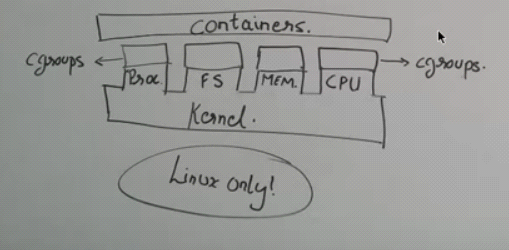




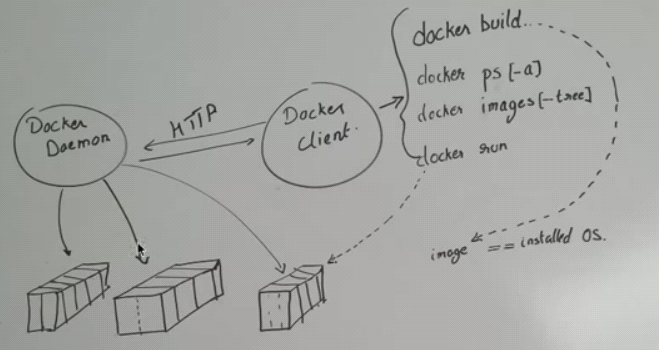
We need to build again..If we change content.

Sol: Share folder

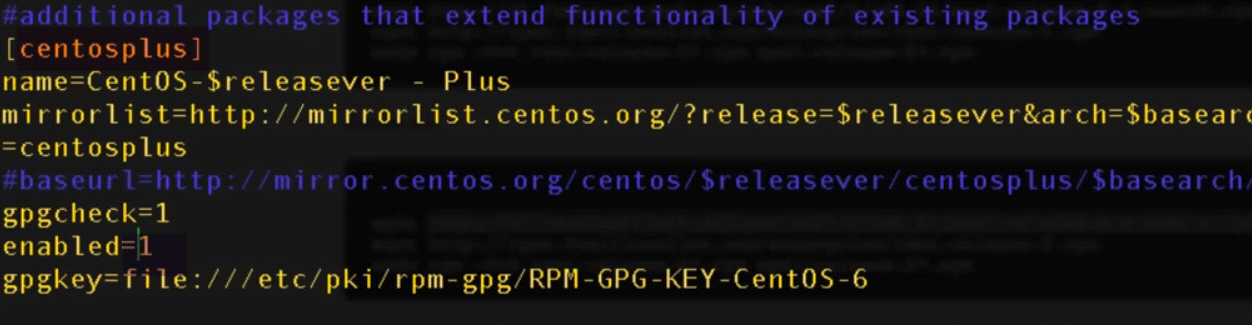




Cgroup: containergroup



Create Custom image



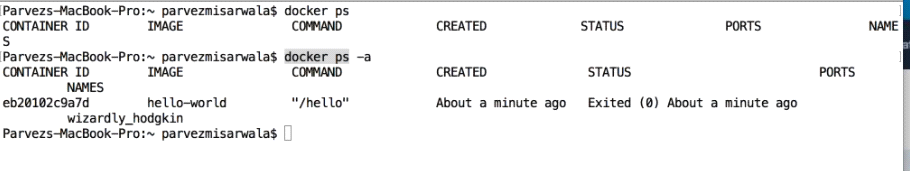
Add EPEL repo

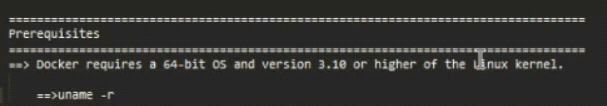
Enabled=1 in centosplus-> CentOSBase.repo

Service docker start

Docker ps

Docker images





 Verify Requirements

For standard installation Docker required 64 bit operating system having Kernel >= 3.10 version. Older versions of Kernel have some missing requirements to run all features of Docker.

# uname -r

3.19.0-49-generic

|  |
| --- |
| * sudo yum update   sudo vim /etc/yum.repos.d/docker.repo  [dockerrepo]  name=Docker Repository  baseurl=https://yum.dockerproject.org/repo/main/centos/$releasever/  enabled=1  gpgcheck=1  gpgkey=https://yum.dockerproject.org/gpg  sudo yum install docker-engine  sudo systemctl enable docker.service  sudo systemctl start docker.service  sudo docker run hello-world  <https://tecadmin.net/install-and-manage-docker-on-centos/>    2.)  sudo yum update  curl -fsSL <https://get.docker.com/> | sh  systemctl enable docker.service  systemctl start docker.service  sudo docker run hello-world |

CentOS-6

Add the EPEL Repository

Docker is part of Extra Packages for Enterprise Linux (EPEL), which is a community repository of non-standard packages for the RHEL distribution. First, we’ll install the EPEL repository:

rpm -iUvh <http://dl.fedoraproject.org/pub/epel/6/x86_64/epel-release-6-8.noarch.rpm>

yum update -y

install Docker by installing the docker-io package:

yum -y install docker-io

service docker start

chkconfig docker on

sudo docker run hello-world

sudo docker images

Issues:

After installation:

1.)udo /etc/init.d/docker status

docker dead but pid file exists

**Sol:** yum update -y device-mapper-libs

<http://stackoverflow.com/questions/27216473/docker-1-3-fails-to-start-on-rhel6-5>

2.) service docker status

**docker dead but subsys locked**

[root@localhost ~]# service docker status

docker dead but subsys locked

[root@localhost ~]# **service docker restart**

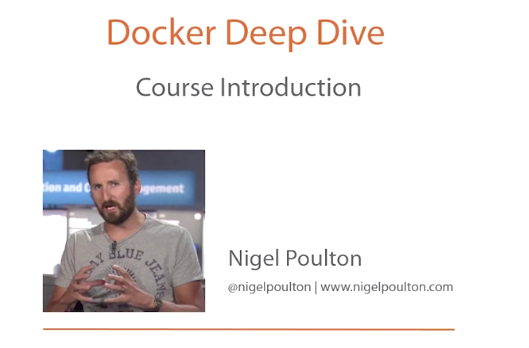
Stopping docker: [FAILED]

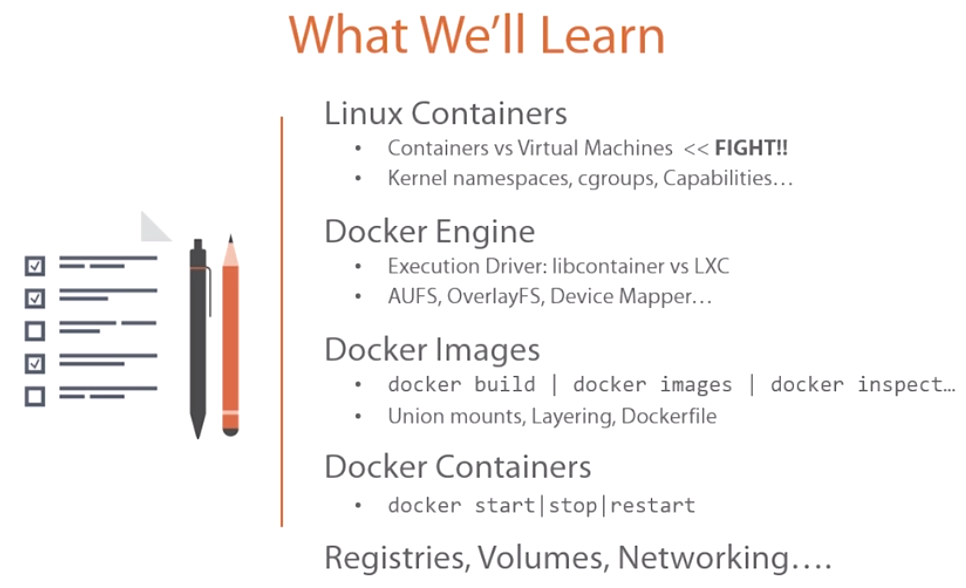
Starting docker: [ OK ]

[root@localhost ~]# service docker status

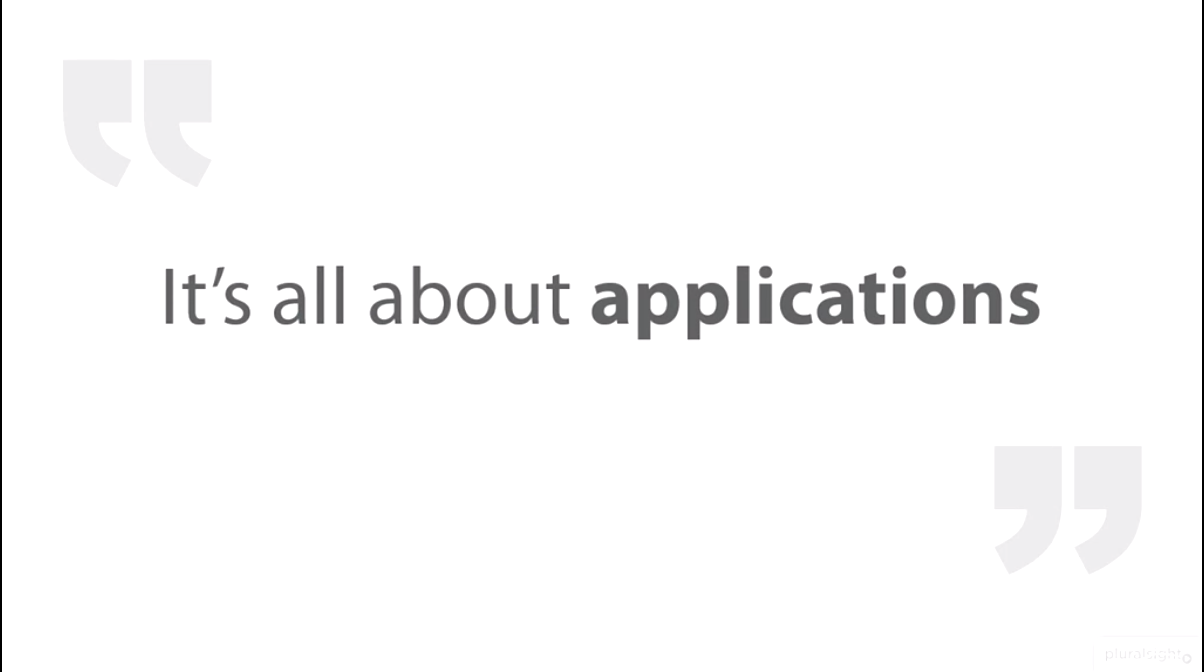
docker (pid 2241) is running...

**Docker Deep Dive**





**I.1.)** 

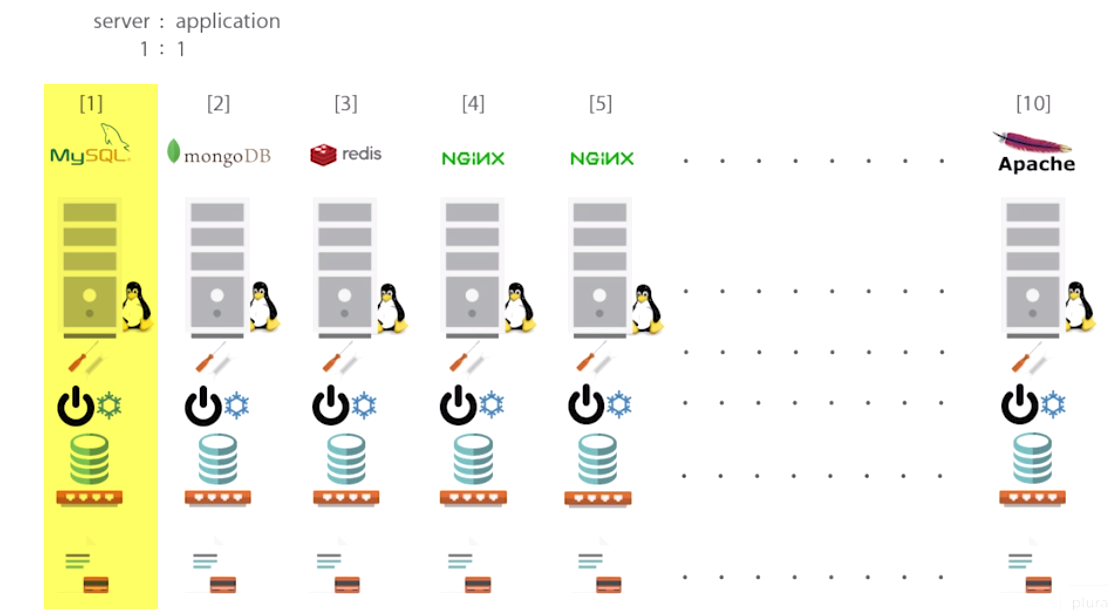


1. **devops\_-The Rise of the Virtual Machine**

**10 applications**🡪 **10 servers, 10 POs->10 OS-> 10 OS licenses-< 10 tools -> redundant coolers-> 10 networking cables, switchboards**

**10 weeks to setup.**

**PO-> make order->rack server->build os-> app install**

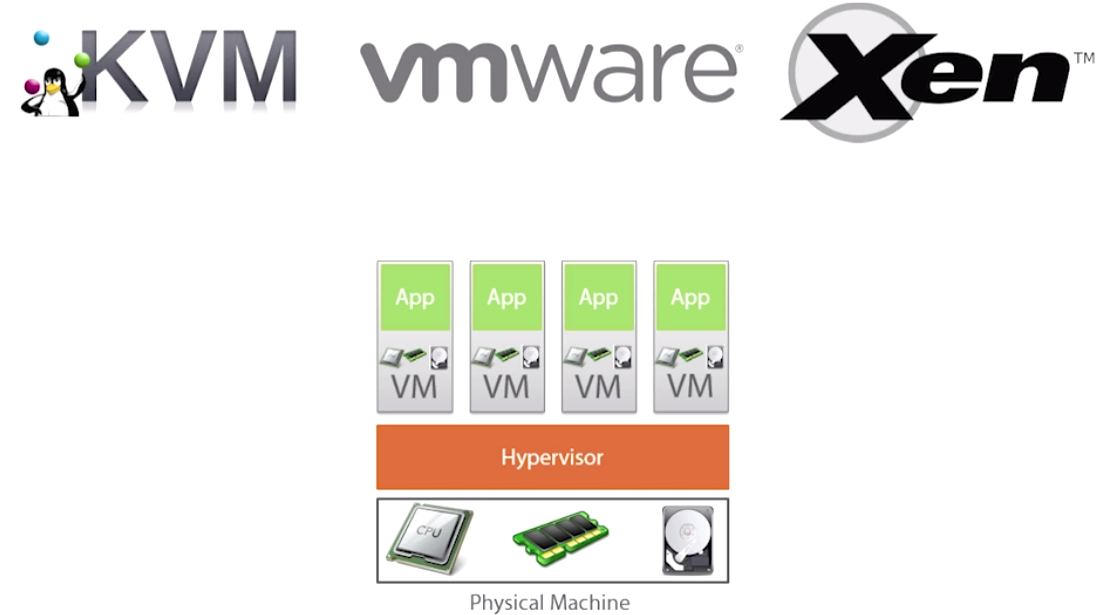


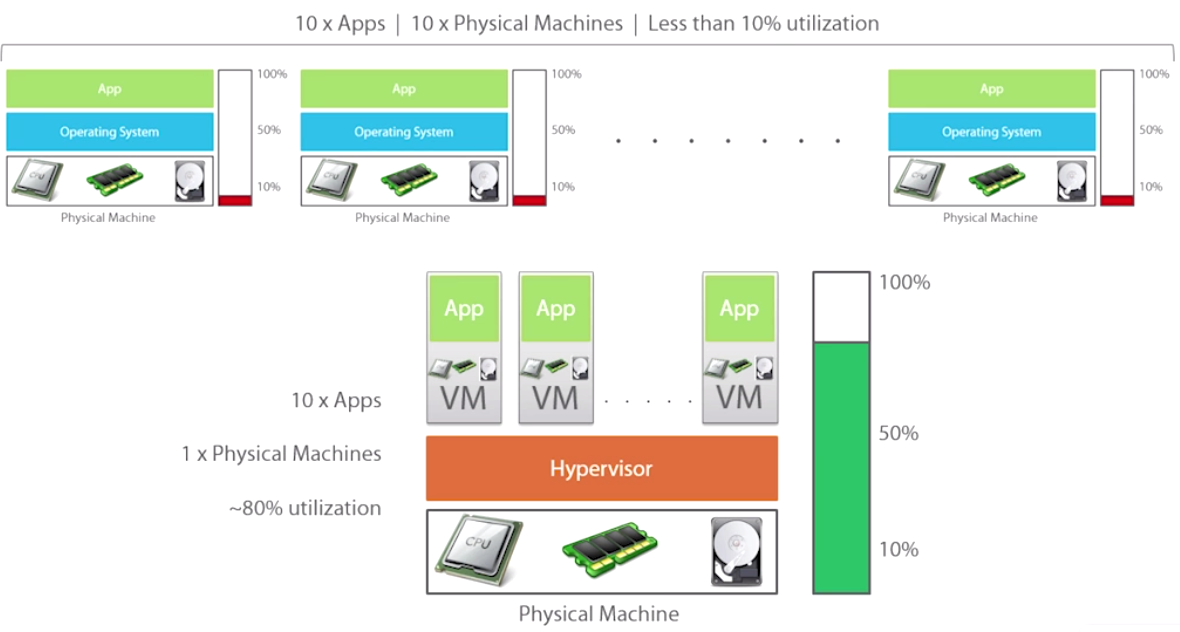
**Massive water of resources,…Actullay we use only 10% of resourses**

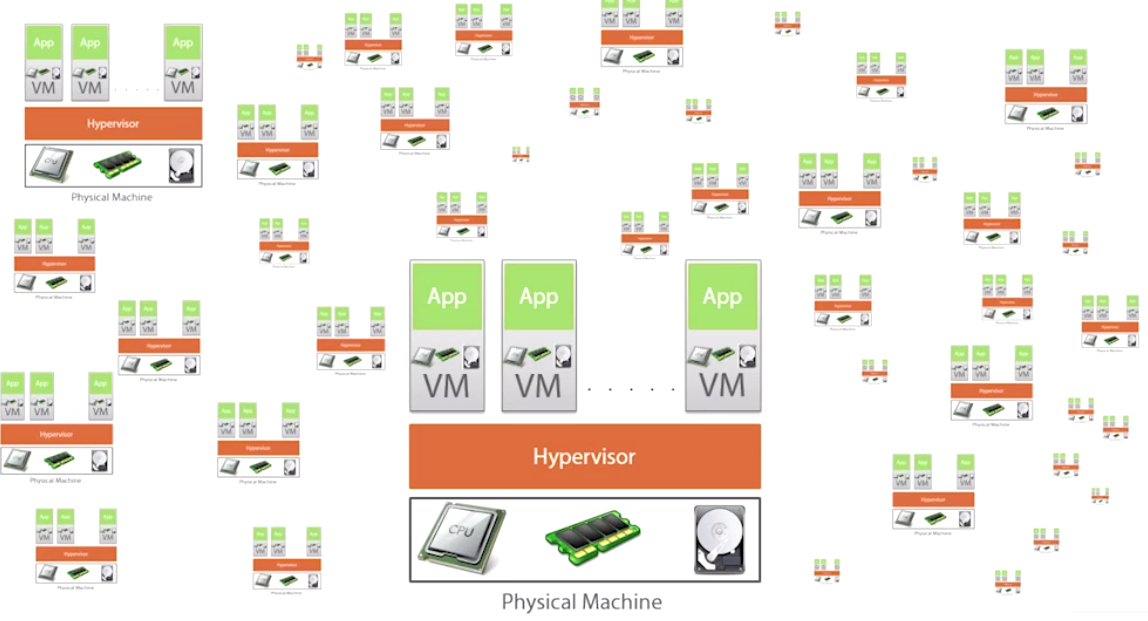
**Cost ton in CAPEX-APEX Cost**



1. **Devops\_02\_02-The Ugly Virtual Machine Virtualization**

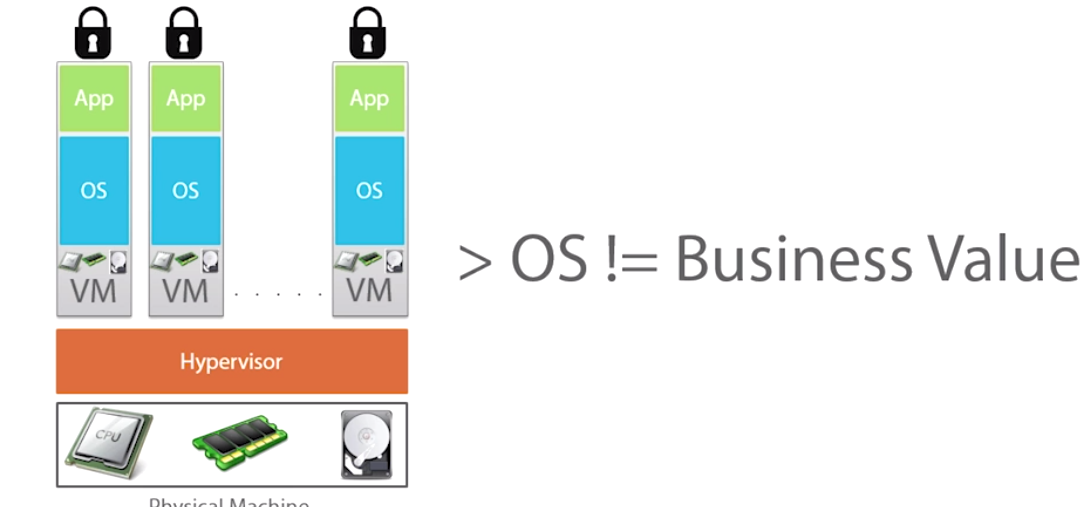






1. **OS only exists to facilitate application.**

**Spin up application without need of OS??**



**We still need 10 Guest OS for 10 applications**



**Every OS has bunch of overheads🡪consumes RAM, HDD, CPU**

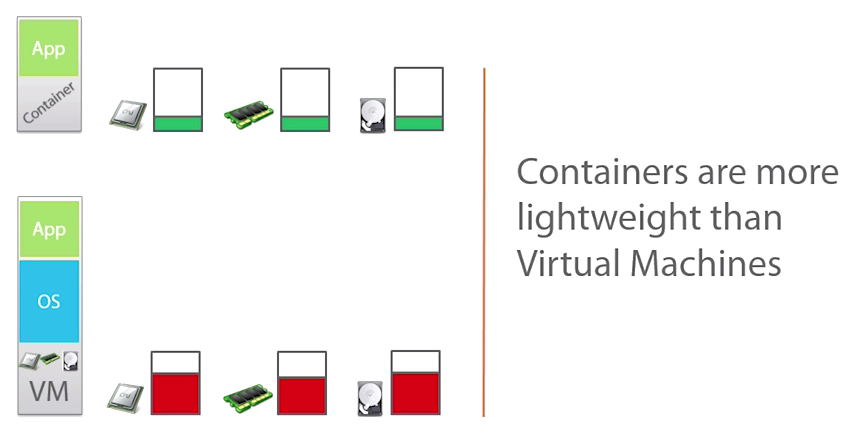
**Each OS has individual licensing**

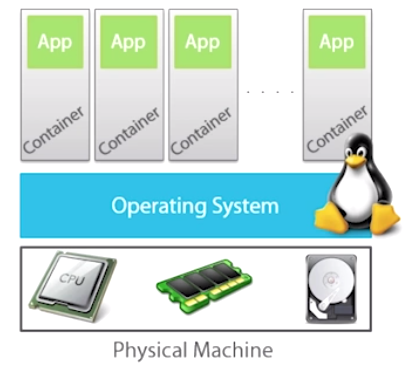
**OS is waste of resources.**

**No matter how big company is… Think of only its applications..**

**- Secure, isolated, minimal OS services..**

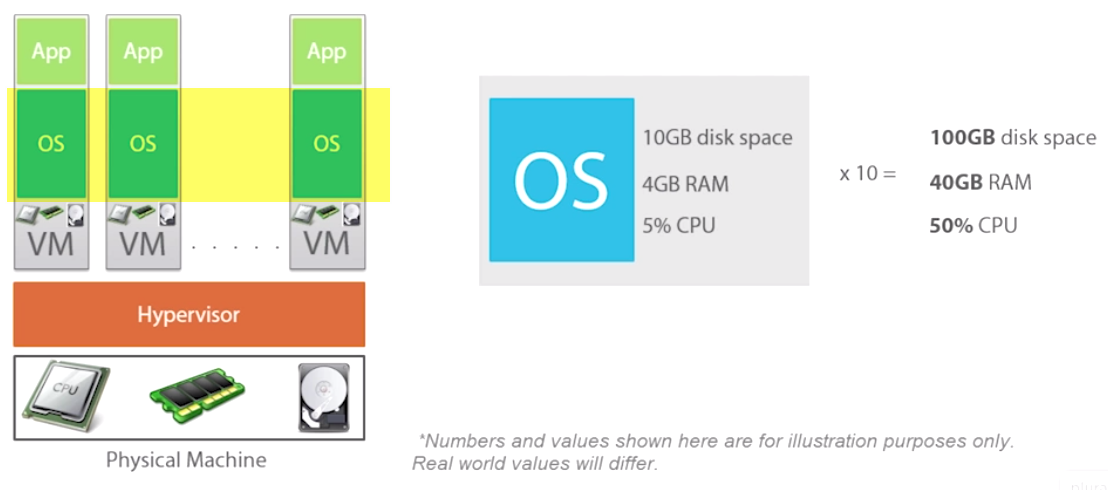
**3.) Devops\_ Containers**





**OS installed on hardware/bare metal**

**Linux kernel owns and manages hardware below it.**

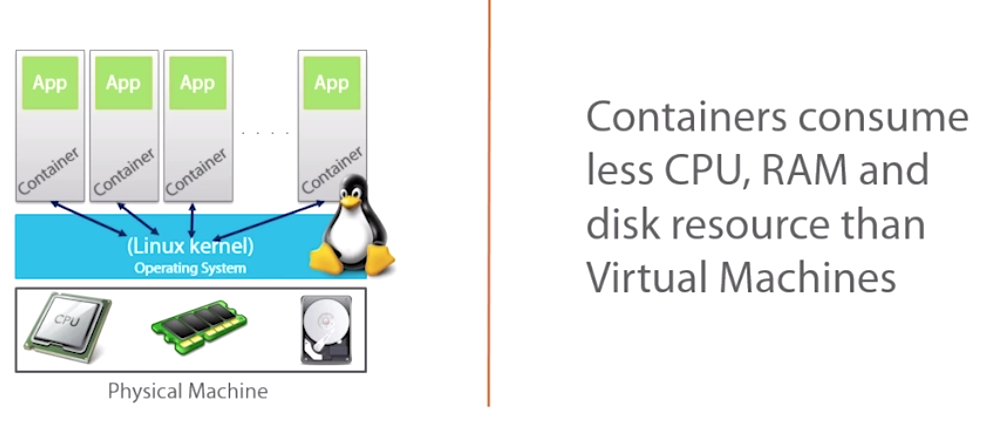


**10 windows licenses**

**Each Container here is a super-light weight OS. (NOTHING like full blown OS, as in VM).**

**Every container shares single common linux kernel, down in the host.**

**So single instance of kernal on machine.**

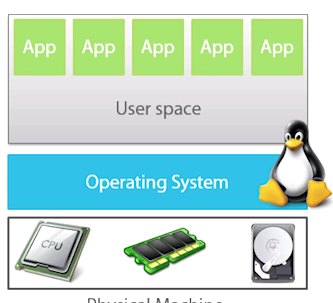


**Also –Portable**

**- faster**

**4) devops\_ 02\_04-Containers Under the Hood**

**Why can’t we install 5 apps in OS ??**

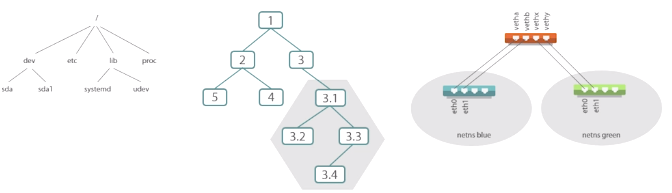


**Lets say 2 apps..Both share common library file. Each need different version of file on same system???**

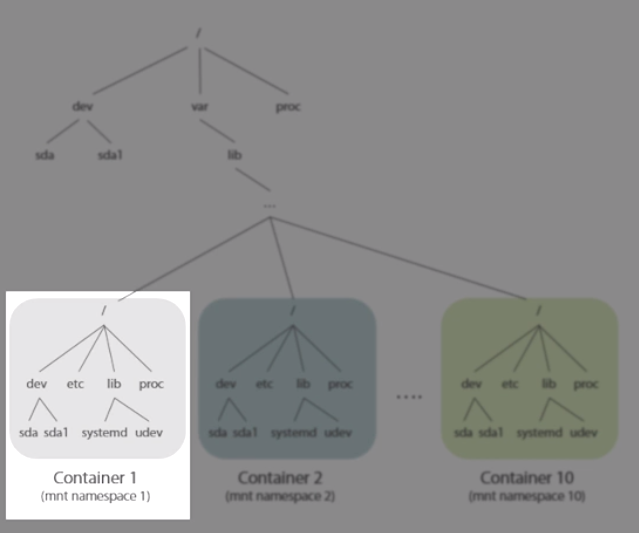
**-Container is isolated instance of user Space**

**- A system with 10 containers have 10 isolated and independent user space….So we can safely install 10 apps on system.**

**We need isolated instances like root FS, process tress, networking stacks**



**EX: isolated view of root FS=> 1 container can have its own FS**



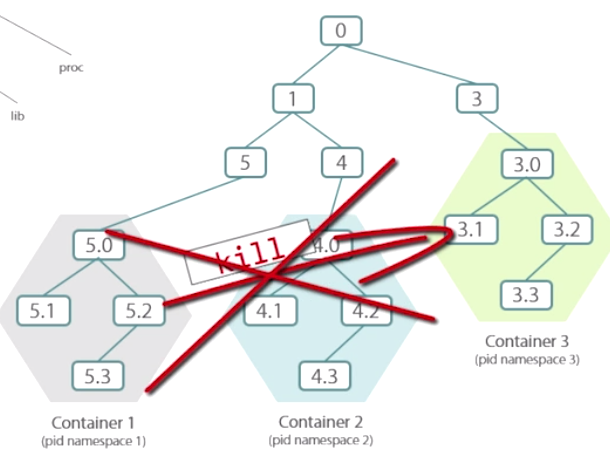
**10 containers=> 10 independent isolated views of root FS**

**An APP running in container-1 can add, modify, remove files(libraries or any) anywhere within its view of FS without impacting other container.**

**Clear: Each container have its view of root FS.So can mess around without impacting other containers**

**ii.) 10 containers->10 independent isolated process trees.**

**Process in 1 container cannot send signal to other. It’s isolated**



**iii.)** Each container get’s its own network stack. IP address, Port range,it’s own routing table

HOW??????????????

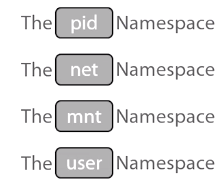
How we get this isolation??????????????????

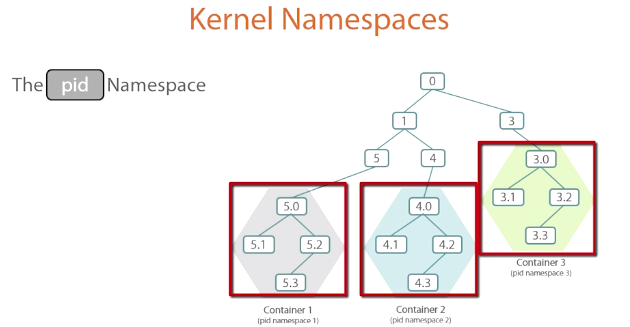
This kind of isolation is achieved by feature called

1.)**Kernel namespaces.**

**2.) CGroups (Control groups)**

**It does- PId, networking, FS(mnt), user namespaces..**





**Partition of namespace like pid/process tree🡪 Isolated partion to 1 container**

**-> Another isolated partition to other container**

**Fig: Single process tree from view of host.**

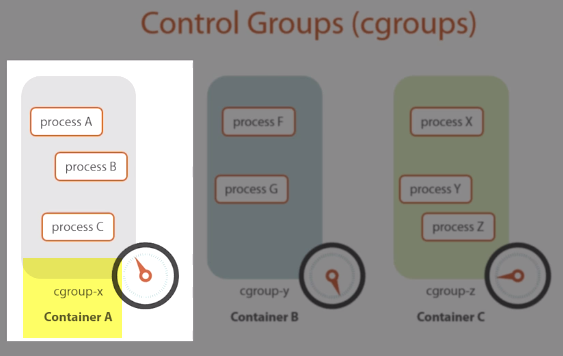
**Each of these Partitions here get assigned to specific container, and**

**Each of these containers are like WALLS.**

**ii.) cgroup(Control groups)**

**Recourses group together and apply limits.**

**In case of container, we map container to cgroups (1-1 mapping) and we set limits on CPU,RAM,…container have access to**



**Cgroups are really flexible..**

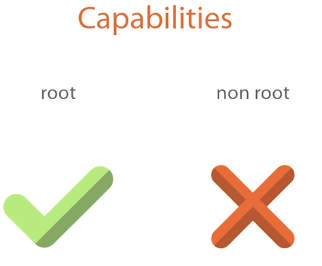
**Need to increase CPU,MEM (resourses)???...Simple. Just tweak cgroup limits**

**- more flexible than VCPU,VCPU on hypervisor**

**Specially -If we plan multiple container on system, like we do…especially when containers have hostile(**opposed or unfriendly)  **workloads, well we need container system that supports cgroups.**

**iii.) Capabilities**

* **What privileges user or process gets..**
* **Security**



**Capabilities take privileges of root user and break them into small privileges**

