DevOps Crash Course

by Peter Yanush

softserve

Коротко про віртуалізацію

Основні системи віртуалізації на базі гіпервізора:

1. VmWare
2. Virtual Box
3. Qemu
4. Hyper-V

Віртуалізація на базі цих систем вимагає роботу через гипервізор, а це оверхед. Віртуалізація вимагає образ з ОС а це робить їх дуже великими. Ну і розгортання таких систем займає не мало часую.

Віртуальна машина вимагає віртуалізацію заліза(ОЗУ,Проц,Відео, т.д.)

softserve

Коротко про віртуалізацію

Основні системи віртуалізації на базі ядра:

1. **OpenVZ**
2. **Systemd-nspawn**
3. **LXC**
4. **Docker**

Віртуалізація на базі цих систем використовує ядро гостьової системи, відповідно до цього ми маємо обмеження в списку розгортаємих ОС.

Виходячи з того що ми використовуємо ядро гостьової ОС ізоляція не дуже надійна так як вона залежить від вразловостей ядра.

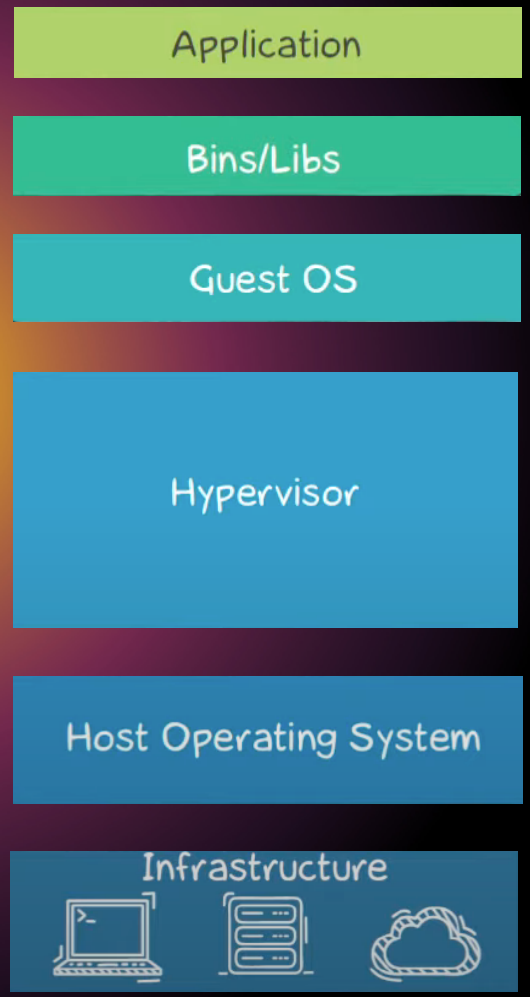
softserve

Різниця між ВМ та Контейнером

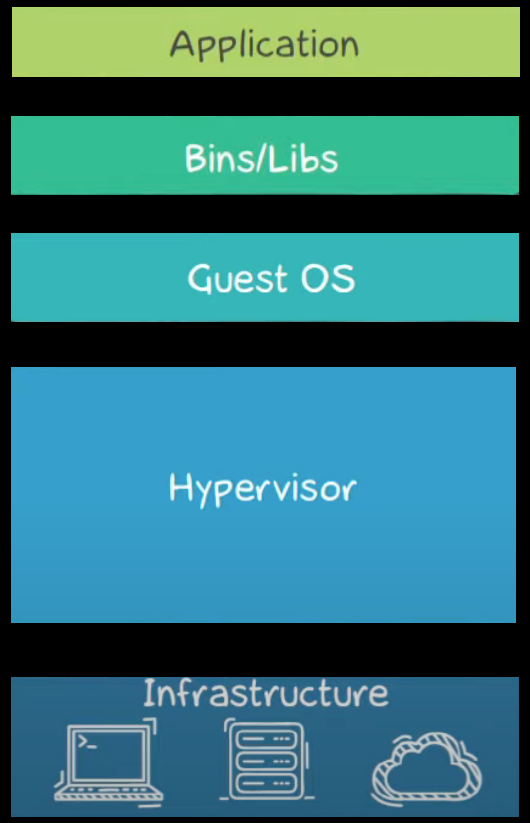
1. **Віртуальна машина віртуалізує все необхідне залізо для гостьової ОС**
2. **Контейнер використовує ядро гостьової ОС**
3. **В ВМ може працювати майже будь яка ОС**
4. **В контейнері здебільшого Linux від недавно Windows 2016 та 10**
5. **Віртуальна машина краще для ізоляції**
6. **Контейнер не дуже підходить для ізоляції має багато вразливостей.**

softserve

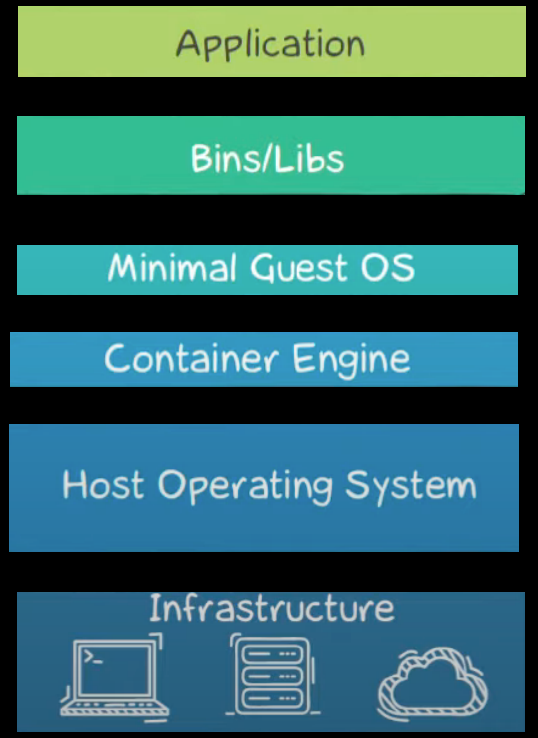
Hosted Hypervisor Virtual Machine



Bare Metal Hypervisor Virtual Machine



Container



softserve

Що використовується для контейнириризації

Namespaces

1. PID
2. Networking
3. Mount
4. User

Control Groups

1. Memory
2. CPU
3. Block I/O
4. Network

А ще:

1. Capabilities(користувач root може повне capabilities і може все змінювати, до прикладу користувач sftp більш обмежені права змінювати)
2. Copy-on-Write(це система яка нам дозволяє працювати з образами докера і відповідно більш кфкутивно їх використовувати)
3. І т.д.

softserve

Move from pets to cattle

З Docker вибудувалась нова філософія побудови проекту:

1. Один процес – один контейнер
2. Всі залежності в контейнері
3. Чим менше образ - тим краще
4. Інстанси стають ефімерні
5. Docker рулить)))

Ефімерні інстанси міняють підхід до всього вже не важливо доступність сервера а важливо доступність сервісу який розгорнутий в цій системі. Відповідно це змінює підхід до моніторингу та збору логів та й загалом до побудови інфраструктури.

softserve

Docker

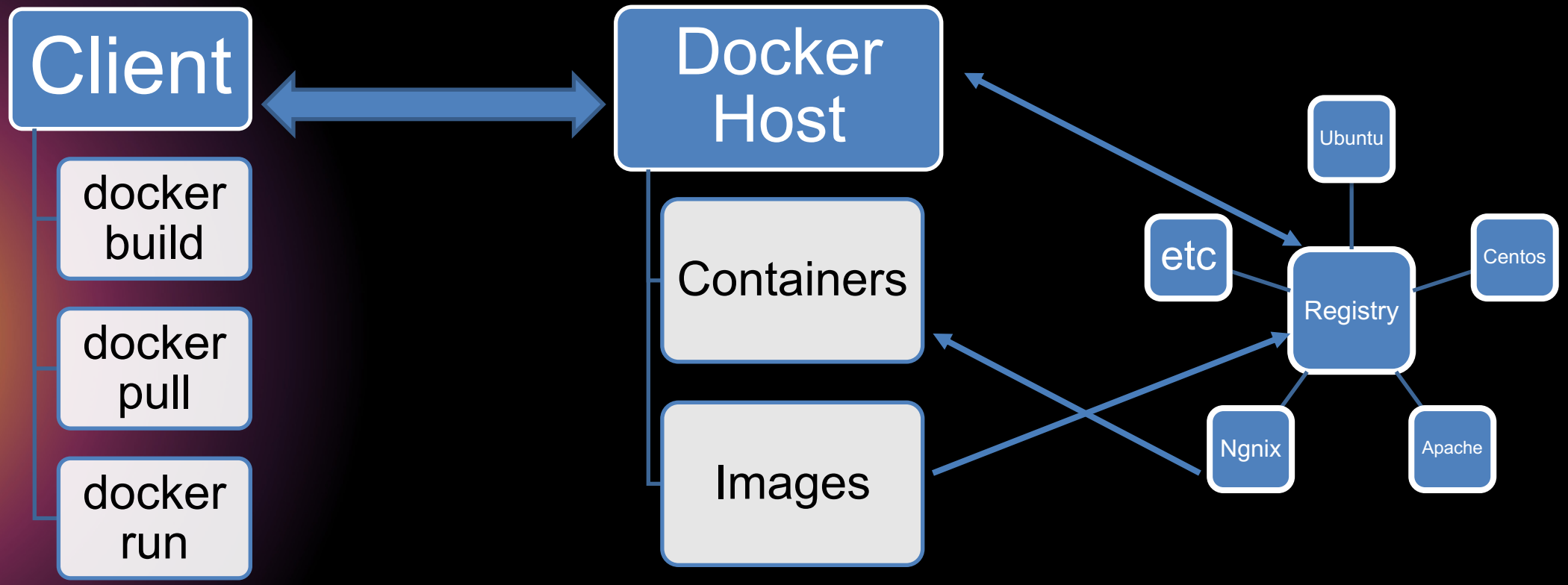
1. Змінив філософію
2. Стандартизував упаковку сервісу(програми)
3. Вирішує питання залежностей
4. Гарантує повторюваність
5. Мінімальний оверхед

softserve

Docker з чого складаєтьця його інфраструктура

- Docker Daemon
- Docker CLI
- Dockerfile
- Image
- Container
- Docker registry

Docker



softserve

Docker Daemon

1. Серверна частина
2. Працює на хост-машині
3. Скачує образи і запускає з них контейнери
4. Створює мережу між контейнерами
5. Збирає логи з контейнерів
6. Створює нові образи

Docker CLI

1. Консольна утиліта для роботи з докер-демоном
2. Може працювати не тільки локально але й по мережі

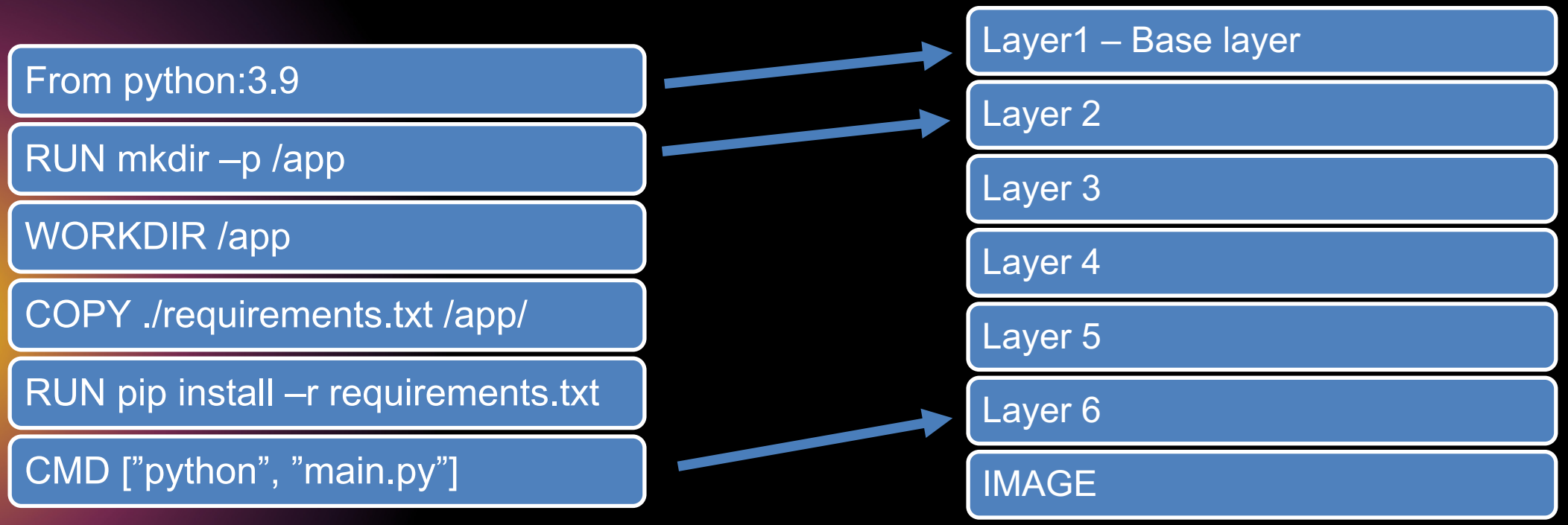
softserve

Docker CLI

1. docker ps – можем подивитись ті контейнери які запущені на хості
2. docker images - можем подивитись образи які скачані локально і доступні зараз
3. docker search <name> - пошук образу в регістрі
4. docker pull <name> - скачати образу з регістрі на машину
5. docker build </path/to/dir> - зібрати образ
6. docker run <name> -запустити контейнер
7. docker rm <name> - видалити контейнер
8. docker logs <name> - логи контейнера
9. docker start/stop/restart <name> - робота з контейнером

softserve

Dockerfile



softserve

Docker Image

1. Пакування нашого контейнера
2. З них у нас запускається контейнери
3. Зберігаються в докер-регістрі або проектних регістрах
4. Мають hash(sha-256), ім’я і tag
5. Мають структуру із ’слоїв’
6. Створюються по інструкції з Dockerfile

softserve

Docker Container

1. Запускається з образа
2. Ізольований
3. Повинен вміщувати в собі все що потрібно для роботи вашої програми
4. 1 процес – 1 контейнер(Best practice)

softserve

Docker practice

1. systemctl status docker
2. docker search hello-world
3. Створюємо imwork.py → print (“I’m work CONGRATS!”) → docker build –t ImWork .[Enter] → Вуаля → Dockerfile
   → FROM python:3.8 (hub.docker.com→search python)

```
RUN mkdir /opt/app
WORKDIR /opt/app

COPY imwork.py  /opt/app

CMD [“python”, “imwork.py”]
```

docker images→docker run→docker ps –a →

create imeverytimework.py → print (str(datetime.datetime.now())) \ time.sleep (2) →

docker build –t ImEWork.[Enter] → docker run –name ImEWork –rm -d

ImEWork→docker stop ImEWork

docker rm $(docker ps –a -q)

docker rmi $(docker images -q)

softserve

softserve