

" A journey between **infrastructure** and **data** "

VERON

ARNAUD



**Postgres LXC & Docker
Containerized Deployment
with OpenSVC**

Agenda

Use Case

Technologies

- LXC
- Docker
- OpenSVC

Demo / Screenshot

Use Case

Postgres Hosting infrastructure (~350 instances ~10 nodes)

Confidential data accessed by thousands people

Production Ready, DRP compliant with 3rd site arbitration

High End storage arrays over stretched san

Synchronous data replication

Containerized deployment (VM-Like with ssh requirement)

Manageability, Reliability, Automated

Technology

LXC



LXC – Linux Containers

- OS Lightweight virtualization technology
- Do not emulate hardware, share OS kernel with hypervisor
- Dedicated os instance, package db, network stack
- Rely on cgroups to allow resources capping (cpu, mem, io, network, ...)
- Use namespaces for pid, uts for hostname, ...
- Can be compared to Solaris Zones or BSD Jails
- Since 2008

```
0:00 | \ \ nginx: worker process
57:15 | \ /usr/bin/python3 /usr/share/opensvc/lib/osvcd.py
0:23 [lxc monitor] /var/lib/lxc/avnprdinfra03
0:11 | \ /sbin/init
0:14 | \ \ /sbin/udev -d
0:00 | \ \ \ /sbin/udev -d
0:00 | \ \ \ /sbin/udev -d
0:00 | \ /sbin/portreserve
1:56 | \ /sbin/rsyslogd -i /var/run/syslogd.pid -c 5
1:14 | \ storapid start -name storapid
0:00 | \ storwatchd start -name storwatchd
1:05 | \ /usr/sbin/sshd
0:00 | \ \ sshd: root [priv]
0:00 | \ \ \ _sshd; root [net]
0:00 | \ xinetd -stayalive -pidfile /var/run/xinetd.pid
0:00 | \ /usr/sbin/saslauthd -m /var/run/saslauthd -a pam -n 2
0:00 | \ \ /usr/sbin/saslauthd -m /var/run/saslauthd -a pam -n 2
1:14 | \ sendmail: accepting connections
0:00 | \ sendmail: Queue runner@01:00:00 for /var/spool/clientmqueue
1:42 | \ /usr/sbin/httpd
0:00 | \ \ /usr/sbin/httpd
0:00 | \ \ \ /usr/sbin/httpd
0:00 | \ \ \ /usr/sbin/httpd
0:00 | \ \ \ /usr/sbin/httpd
0:23 | \ cron
226:32 | \ /usr/bin/python /usr/share/opensvc/lib/osvcd.py
0:23 [lxc monitor] /var/lib/lxc/avnprdinfra04
0:12 | \ /sbin/init
0:14 | \ \ /sbin/udev -d
0:00 | \ \ \ /sbin/udev -d
0:00 | \ \ \ /sbin/udev -d
0:00 | \ /sbin/portreserve
2:04 | \ /sbin/rsyslogd -i /var/run/syslogd.pid -c 5
1:02 | \ storapid start -name storapid
0:00 | \ storwatchd start -name storwatchd
1:12 | \ /usr/sbin/sshd
0:00 | \ \ xinetd -stayalive -pidfile /var/run/xinetd.pid
0:00 | \ /usr/sbin/saslauthd -m /var/run/saslauthd -a pam -n 2
0:00 | \ \ /usr/sbin/saslauthd -m /var/run/saslauthd -a pam -n 2
```

Technology

Docker



Docker

- Standardized packaging for software and dependencies
- Apps isolation from each other
- Share same OS kernel
- Use namespace and cgroups
- No guest OS layer to maintain/patch
- Not tied to virtualization solution, but only to docker stack
- Fast, Portable, less deps
- Since 2014 (v1.0)

```
23900 ?      Ssl  23:44      \_ /usr/bin/dockerd -R unix://
23924 ?      Ssl  260:49     \_ /usr/bin/containerd
90196 ?      Sl   1:55       \_ \_ containerd-shim -namespace moby -workdir /var/lib/con
tainerd/io.containerd.runtime.v1.linux/moby/0bdd936ac3bc449db7046e5fedc96d228ef7c3137a77655a
69da7105e13aef6b -address /run/containerd/containerd.sock -containerd-binary /usr/bin/contai
nerd -runtime-root /var/run/docker/runtime-runc
90599 ?      Ssl   0:10      | | \_ /pause
1266 ?      Sl   1:54      | \_ containerd-shim -namespace moby -workdir /var/lib/con
tainerd/io.containerd.runtime.v1.linux/moby/1a19ed1140365e960d4e71a847f6b15915314780e561cc22
83d07f89a23b8691 -address /run/containerd/containerd.sock -containerd-binary /usr/bin/contai
nerd -runtime-root /var/run/docker/runtime-runc
1381 ?      Ssl  37:49      | | \_ mysqld
1727 ?      Sl   1:56      | \_ containerd-shim -namespace moby -workdir /var/lib/con
tainerd/io.containerd.runtime.v1.linux/moby/c851964af55802993c49bec3d46ad5c949f2ea3eb7ae2de2
62f983c768aa0cbe -address /run/containerd/containerd.sock -containerd-binary /usr/bin/contai
nerd -runtime-root /var/run/docker/runtime-runc
1783 ?      Ss   1:57      | \_ apache2 -DFOREGROUND
3831 ?      S    0:37      | \_ apache2 -DFOREGROUND
```

Technology

OpenSVC



OpenSVC Introduction

OpenSVC is an open source product for managing traditional and containerized applications across multiple hosts



OpenSVC

- Linux, Unix, Windows
- Python Language
- Daemon + Command Set
- Heterogeneous deployment (bare-metal, VM, container, cloud, ...)
- Multi datacenters, Inter Continental Cluster
- Service oriented
- Centralized web portal
- Since 2009

OpenSVC – From traditional...

Service Topologies

standalone (quick, easy, systemd alternative)
cluster failover, flex, scaler (manual or automated)

Resources Types

ip, disk, vg, lv, pool, fs, app, sync, task, container

Resources Actions

start/stop/status/restart
scope by family, by tag, up to, down to

Services actions

start/stop/status/restart (easy to delegate)
cli service selector with patterns

Inventory

Orchestrate

Audit

Monitor

Alert

OpenSVC – ... to Orchestration

Asynchronous target-state driven actions

actions/targets are posted to a daemon listener
and then orchestrated by the cluster daemon monitors

Production ready

quorum arbitration
resilient cluster comms: unicast, multicast, disk, third site relay

DevOps oriented

Templating & (Un)Provisioning features
Rest API
delegate service management and deployment to third party
infrastructure through socket (Unix, TLS)

Provision

Unprovision

Automate

Delegate

Autoscale

OpenSVC – ... to Orchestration

Containers integration

Orchestrate legacy & modern techs :

zones, jails, lxc, lxd, kvm, docker, podman

Enterprise grade Docker registry management

Cluster backend networks (CNI)

Embedded DNS Backend management

L4 & L7 cluster ingress gateways

Meta Service

describe complex application stacks

schedule dependencies between services and subservices

HTTP/2 TLS Socket

Remote management

Provision

Unprovision

Automate

Delegate

Autoscale

OpenSVC as SysOps ↔ DevOps bridge

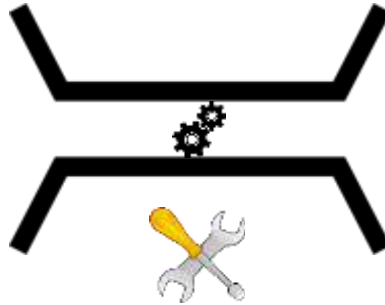
SysOps

Features :

- Deployment procedure
- Infrastructure “as code”

Services types :

- High availability without application collaboration
- Long lifecycle app
- Legacy / Traditional



Clustering Provisioning
Orchestration Automation

DevOps

Features :

- Orchestrated deployment
- CI/CD

Services types:

- Native app resiliency
- Dynamic scaling
- Short lifecycle (testing)
- Modern/Stars

OpenSVC – Services Topologies



Active/Passive



Flex



Flex



Flex

Active/Active



Scaler



Scaler



Scaler



Scaler



Scaler

Active/Active with
autoscaling (+ -)



Cluster

DEMO

Use case => Template => Deployment (LXC, Docker)

<https://github.com/opensvc/dataops2019>

Data**ops**

binlogic

THANK YOU

JUNE 20-21

2019

Follow us **@BINLOGIC**