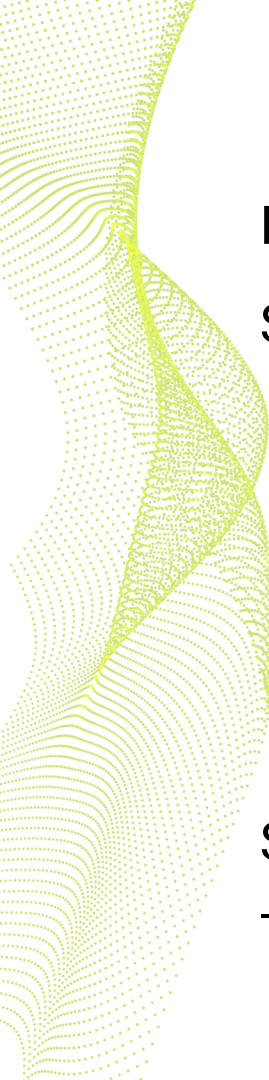




The evolution of cloud orchestration systems from ephemeral to persistent storage

CNCF webinar
2020-10-07



Boyan Krosnov - CPO of StorPool

StorPool is

- very fast and very reliable scale-out block storage system
- software-defined
- API controlled, DevOps, New IT, integrations
- CNCF member; StorPool CSI

StorPool is integrated with a number of cloud orchestration systems
- OpenStack, CloudStack, OpenNebula, OnApp, **Kubernetes**

Agenda

- Cattle vs Pets
- Separation of code and state
- Historical review
 - AWS - EBS
 - OpenStack - Cinder
 - Kubernetes - CSI
- Conclusions

Cattle vs Pets

Pets



Servers or server pairs that are treated as indispensable or unique systems that can never be down. Typically they are manually built, managed, and “hand fed”. Examples include mainframes, solitary servers, HA loadbalancers/firewalls (active/active or active/passive), database systems designed as master/slave (active/passive), and so on.

Cattle vs Pets

Cattle

Arrays of more than two servers, that are built using automated tools, and are designed for failure, where no one, two, or even three servers are irreplaceable. Typically, during failure events no human intervention is required as the array exhibits attributes of “routing around failures” by restarting failed servers or replicating data through strategies like triple replication or erasure coding. Examples include web server arrays, multi-master datastores such as Cassandra clusters, multiple racks of gear put together in clusters, and just about anything that is load-balanced and multi-master.

Separation of code and state

- Traditional IT -> Pets
- Modern IT & DevOps -> mostly Cattle
- even Cattle need persistent storage
- there are still many Pets around - physical-to-VM, VM-to-container

Separation of code and state

In traditional IT

- x86 servers
 - "compute"
 - multi-tier application
 - local disks in servers are just for the OS and software
- SAN (Block storage) or NAS (Filer) stores data
 - High availability
 - Shared system - multiple apps on one system
 - the same data accessed by multiple servers

Separation of code and state

Code:

- package application as a set of containers - reduce/remove all external dependencies. No deb/rpm/npm/pip hell.
- packaging for whole complex applications - multiple containers - e.g. with Helm Charts

State:

- Started as "someone else's problem"
- State in database
- Shared filesystem
- State in object store / S3

History - Amazon EBS

- August 2006: EC2 (virtual machines)
- August 2008: EBS (block storage service, virtual disks for virtual machines) - 2 years after EC2
- features added over time: boot from EBS, snapshot, clone, resize

History - OpenStack Cinder

- October 2010: first release
- some persistent volumes added into main project
- September 2012: Cinder service - block storage plugins
- Add features over time - booting, migration, encryption, resize
- The same pattern as AWS - 2 years from compute service to a modular persistent storage service.

History - Kubernetes CSI

- June 2014: first public release
- supporting stateful applications is an explicit project goal since 2016, StatefulSets
- some persistent volume support added into main project, FlexVolumes
- June 2018: 1.10 introduces CSI
- features added over time - raw block, cloning, snapshots, resize

Conclusions

- Cloud (orchestration) systems start with a romantic view of a pure world where every component of every application is stateless Cattle
- Reality sets in 2-4 years later
 - Even Cattle often need persistence
 - There are many Pets still alive and well
- You need a good solution for persistent storage in your cloud



Thank you
info@storpool.com

