Portworx and DCOS

Portworx Storage on DCOS using
AWS CloudFormation and EBS block devices

Background

Background

Consumer facing services like Siri and Twitter run at un-imaginable scale and deployments often involve thousands of compute nodes.

A battle-proven, enterprise grade solution to orchestration at this scale is Apache Mesos. It pools compute nodes and makes them available to pluggable frameworks to use when deploying workloads.

portworx

Background

These past few years have seen a huge adoption in containers and in particular Docker containers.

Marathon is a Mesos framework that will schedule Docker containers onto your cluster of compute nodes. This frees up developers to concentrate on their `Dockerfile` and Marathon will think about what compute node to run the container(s).

dcos is a packaged version of Mesos and Marathon (along with other tools) built by a company called Mesosphere. It has a GUI and CLI to make the operation of a Mesos cluster running Marathon easier.

State{ less, full }

State{ less, full }

This stack will work great if your entire workload is stateless (perhaps a Twitter bot or a Google Maps api client). Quite often though, there is a process or two that will require persistent storage (perhaps a Postgres or Redis server).

Following the 12-factor manifesto we could run our stateful services as pet's not cattle and install them on single-purpose static servers.

portworx

State{ less, full }

This seems like a shame, we have the full power of an industrial grade container scheduler but still have to manually operate some of our stack.

What if we had a tool that treated our heterogeneous compute cluster as a heterogeneous storage cluster too?

portworx

portworx

Portworx

Portworx storage offers a container aware storage fabric that will run on a commodity cluster of compute nodes.

This lets us schedule a stateful workload using Marathon and not worry if it ends up on node A, B or C - Portworx storage will provision a volume **before** the container starts (because of the low-level Docker volume plugin).

Portworx

Because Portworx storage offers replication - we automatically have high-availability for our Postgres, Redis, MySQL or otherwise stateful container (if the container lands on another node - Portworx storage will ensure the data is there).

We can also take snapshots of existing volumes and then run other workloads against the snapshot volume. For example, we could easily run a test-suite against a snapshot of production data only a few seconds old.

Compute

AND

Storage

Compute AND Storage

Mesos plays the role of the kernel in our cluster and Marathon that of the init system. This allows us to treat a cluster of many nodes as one large computer.

Adding Portworx storage to this cluster means we now have a unified storage layer. It knows where containers are and what volumes they need. It will get the data volume in place before the container starts and constantly replicate data to other nodes without the container needing to know - a truly container aware storage fabric!

Unify your entire stack and deploy stateful alongside stateless processes to the same cluster using the same orchestration framework.

Let's get to it!

Workshop

In this workshop we will:

- configure the aws cli
- create a dcos cluster using AWS CloudFormation
- setup nodes and attach block devices
- deploy etcd and marathon-lb using dcos
- deploy px-dev using dcos
- explore our cluster using pxctl

Workshop (cont...)

In this workshop we will:

- deploy a stateful app
- demonstrate HA by doing failover on the app
- snapshot a volume
- deploy a test workload against the snapshot volume

Workshop (cont...)

You can follow along with the workshop yourself:

https//github.com/binocarlos/px-posts/dcos

Portworx and DCOS

Visit the Portworx website to find out more!