DigitalOcean

How we Operate Ceph at Scale

Cephalocon 2022, Virtual Event

Matt Vandermeulen, Storage Systems

Contents

- Who are we?
- Ceph use at DO
- Automation
- Cluster Operations
- Reflection
- Q&A

A brief history

What is DigitalOcean?

- Founded in 2012
- Core concept: Simplicity
- SSD backed VM was very attractive in 2012
- Four years later we introduce Volumes (block)
- Many more products since, including Spaces (object) and several SaaS offerings such as DBaaS, DOKS (k8s), Serverless Functions, and managed hosting with Cloudways
- More than 12 datacenters in 8 regions
- IPO in March 2021

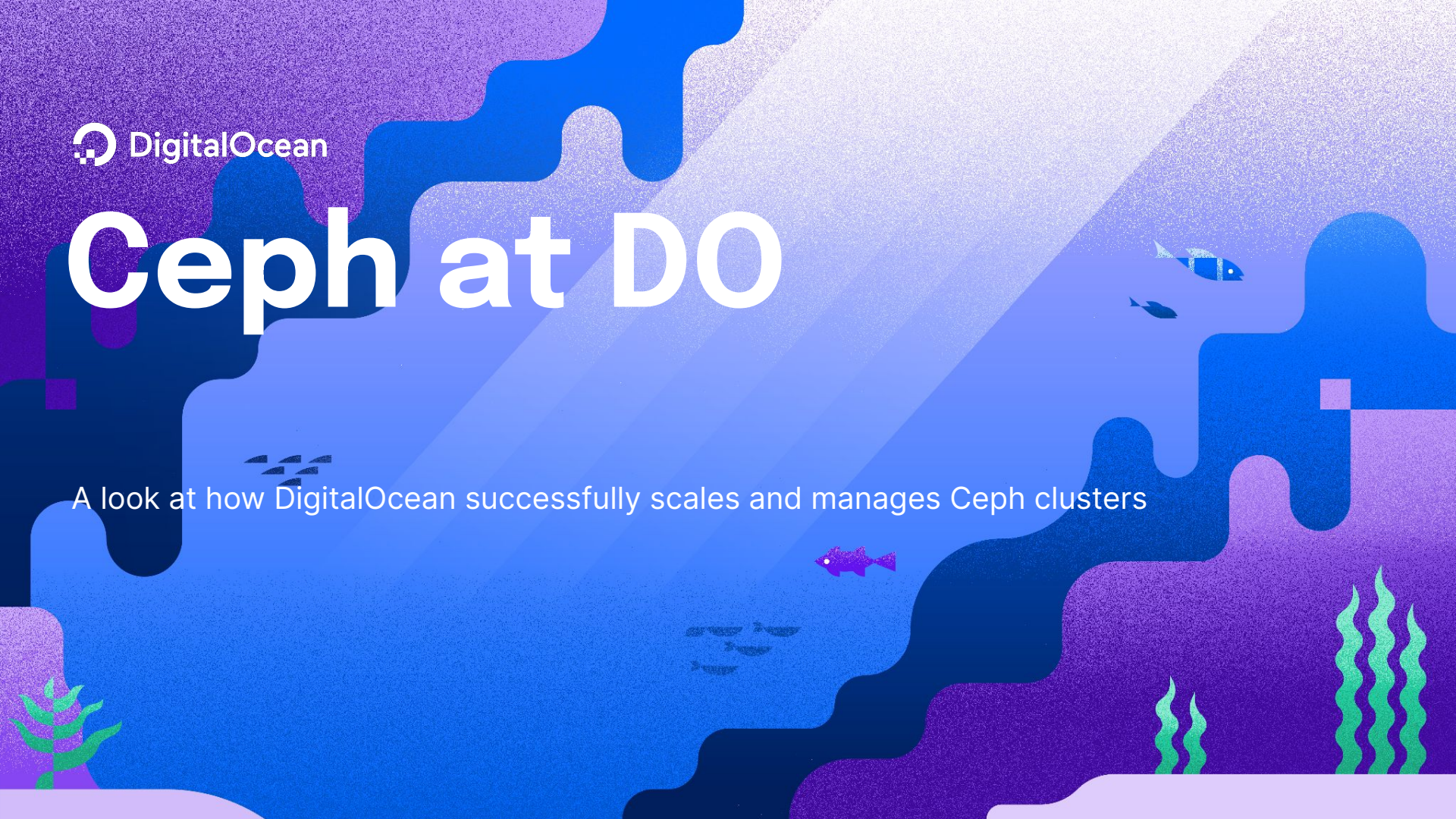
2012 Droplet: Introduction of the SSD-backed VM

2016 Volumes: Ceph backed detachable droplet storage

2017 Spaces: Ceph backed object storage

The Storage Systems Team focuses on supporting the operations and development of the data persistence layer of storage products.

Storage Systems
Mission Statement


DigitalOcean
Ceph at DO
A look at how DigitalOcean successfully scales and manages Ceph clusters

Ceph Usage

- Block storage (Volumes)
- Object storage (Spaces)
- Other teams consume Volumes and Spaces for many products

Automation: Node Introduction

- **Other supporting teams order hardware that we've qualified**
  - We stress test all equipment before deploying
- **The datacenter team racks all the equipment**
  - They also populate the source of truth for the equipment
- **The network team configures ToRs and OOB**
  - Trickier for Spaces which is public facing
- **The infrastructure teams take nodes through provisioning**
  - Large workflow that ensures firmwares are up to date
  - Nodes end up with a base OS, and are handed off to us
- **Storage Systems fires automation to put these nodes to work**
  - Another large workflow that Ceph-ifies a group of nodes
  - When workflow completes, we have a production ready cluster!

Automation: Tooling

Chef runs on a staggered cron throughout the fleet ensuring packages are installed as expected, and other things we don't manage, like the kernel, are up to date.
We heavily rely on Ansible/AWX to do many maintenance operations that are long running, cluster deployments, augments, etc.
There are many bash scripts still written and used as necessary. Don't let perfect be the enemy of progress.

01 Chef for general configuration management

02 Ansible via AWX for just about everything Ceph

03 Hacky bash scripts or other one-off use case specific tools

Automation: Ansible Use Cases

- Net-new Ceph cluster deployment
- Existing Ceph cluster augment: Nodes and disks
- Node reboots: On demand, and when required (i.e. kernel updates)
- Maintaining node-local and centralized Ceph configuration
- Ceph upgrades, and other long-running operations
- OSD restarts
- Node/Cluster decommission
- Lots of utilities/goodies

Automation: Snowflakes

- **Automation thrives on consistency**
- **Snowflakes, being unique, are not consistent**
- **Snowflakes include**
  - CPU, RAM, Disk models
  - Network configs
  - Centralized Ceph configuration
  - That one script you forgot was running…
- **Melt your snowflakes!**

Operations: Deployment

- Run our giant `ceph-deploy` playbook
- Wait
- :woo: or :try-not-to-cry:

Operations: Deployment

- **Simple stuff first: Ensure `chef` is up to date**
- **Create the initial `monmap`, deploy monitors, establish quorum**
  - This requires some integration with our secrets solution
- **OSDs are created ahead of time, then deployed in parallel**
  - CRUSH is populated first, then all OSDs deployed in parallel
- **Deploy and start OSD containers**
- **Verify cluster is healthy and bored**
- **Future: Zero-intervention deployment**

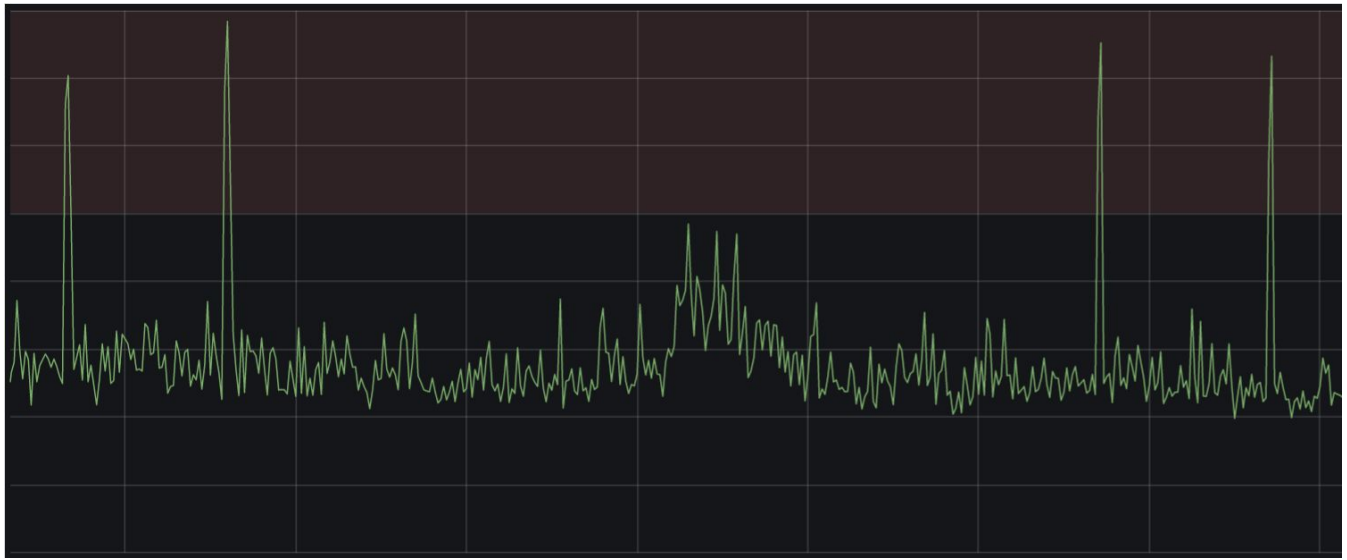
Operations: Augment

- **Block and Object are slightly different**
  - Deploying the OSDs is the same
  - Introducing them to the cluster is different
- **Block slowly upweights the OSDs over time**
  - Makes use of `archimedes`
  - Mostly to mitigate peering latency
- **Object cancels all remapped backfill and slowly undoes them**
  - First set `nobackfill` and `norebalance` OSD flags
  - Makes use of `pgremapper`
  - Mostly to maximize concurrency and minimize degradation
- **It's possible both could make use of the normal upmap balancer**

Operations: Maintenance

- Cluster augments
- OSD restarts, failures, flaps
- Node reboots, failures
- Peering storms

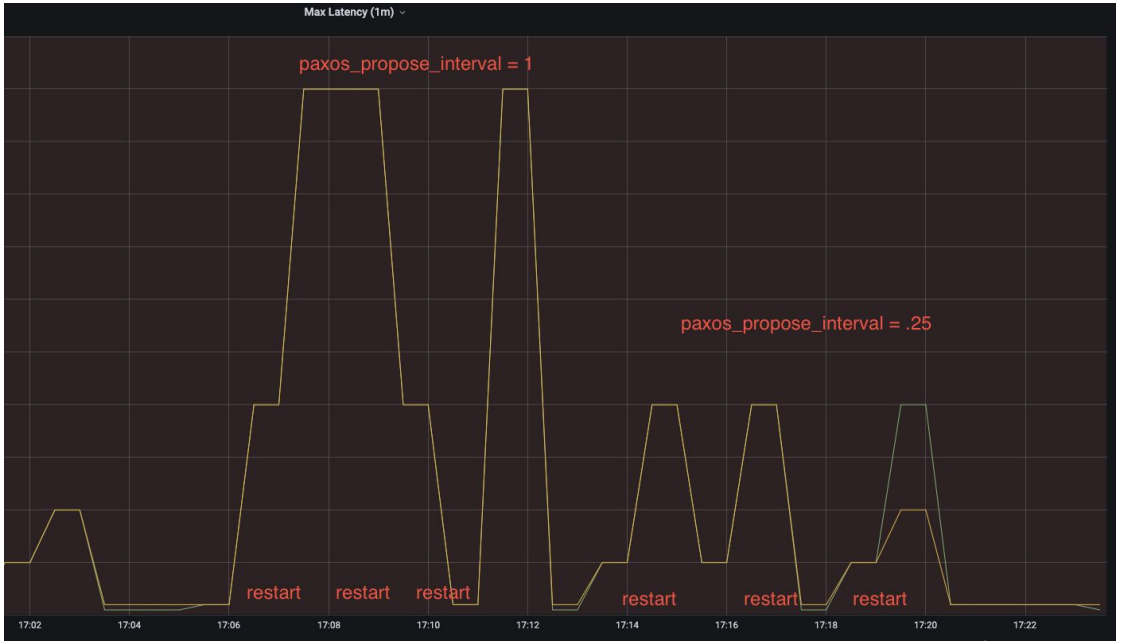
Operations: Peering Latency



Operations: Peering Latency

- In a mailing list message, Sage suggested reducing `paxos_propose_interval` from its 2s default
- We observed that this made peering latency (slightly) worse
- What if we don't let peering begin as the OSD process starts?
- Set `noup`, bring OSDs up, wait for `status = preboot`, unset `noup`
- Reduces CPU contention when starting a host full of OSDs at once
- Reduces `osdmap` updates, too!
- Now reducing `paxos_propose_interval` seems to help!

Operations: Peering Latency



Operations: Index Pain

- Our Spaces clusters have billions (and billions) of objects
- RGW index layer does not like buckets with millions of objects
- Prior to dynamic resharding and async recovery meant real pain
- Compaction in `rocksdb` could cause outages
- We were spilling over into L5 because our index layer is huge
- RGW defaults are not tailored to handle this scale

Operations: Index Squish?

- `rocksdb`, by default, won't compact a file until the level is full
- This means no upper bound on tombstone lifetime
- We're left with tombstone buildup, and slow iterations
- Explore `rocksdb` options looking for silver bullets
- A lot of effort was spent in this work
- Nautilus brought a newer `rocksdb` with more options

Operations: Index Squish!

- **We discovered the `ttl` option in `rocksdb`**
- **This option forces compaction on data that reaches a given age**
- **The first `ttl` compaction run can be stressful on the OSD**
  - We did not experience any issues, though it took hours
  - Disable GC+LC while this runs if necessary
- **We now have consistently higher load on our index nodes**
  - Absolutely worth the trade off
- **We were able to discard *tons* of junk data**
  - We have an upper bound on tombstone lifetime!
- **We disabled periodic compactions in favour of `ttl` compactions**
- **This has been a silver bullet… so far**

Operations: Observability

- ceph_exporter vs prometheus module
- storexporter
- marigraph
- Alert on actionable things
- Make use of inhibition as appropriate


DigitalOcean
Reflection
If we did it all again

Reflection

- Treat block and object clusters similarly
- Use a single source of truth… for any truth
- Finding that tradeoff between automation and services is tricky
- Melt your snowflakes as soon as possible!


DigitalOcean
Thank You!
Hiring plug https://www.digitalocean.com/careers
Q&A?