#### PERCONA

MongoDB in the Cloud: How to Reduce Infrastructure Costs Using Open-Source Components and Kubernetes - Percona

Michał Nosek

Senior Enterprise Architect @ Percona





© Copyright 2023 Percona® LLC. All rights reserved



### I. Why costs are getting out of control?

### II. K8s and Operators 5 min crash course

### III. Practical approaches to cost reduction

#### IV. Summary





# I. Why costs are getting out of control?

Cloud and region	MongoDB Atlas	laaS pay-as-you-go	laaS reserved (3y)							
M30 (3-node RS, 2vCPU, 8GB RAM, 40GB disk)										
AWS us-east1										
Azure centralus										
GCP us-east1										
I	M200 (4x3-node RS, 64vCP	U, 256GB RAM, 2TB storage	)							
AWS us-east1										
Azure centralus										
GCP us-east1										



Cloud and region	MongoDB Atlas	laaS pay-as-you-go	laaS reserved (3y)							
M30 (3-node RS, 2vCPU, 8GB RAM, 40GB disk)										
AWS us-east1	\$4,730	\$2,666	\$1,204							
Azure centralus	\$5,256	\$2,450	\$1,049							
GCP us-east1	\$3,854	\$2,796	\$1,393							
Ν	M200 (4x3-node RS, 64vCP	U, 256GB RAM, 2TB storage	)							
AWS us-east1	\$555,822	\$355,086	\$159,674							
Azure centralus	\$626,515	\$366,672	\$162,575							
GCP us-east1	\$525,755	\$379,376	\$198,303							

Cloud and region	MongoDB Atlas	laaS pay-as-you-go	laaS reserved (3y)
	M30 (3-node RS, 2vC	PU, 8GB RAM, 40GB disk)	
AWS us-east1	177%	\$2,666	\$1,204
Azure centralus	214%	\$2,450	\$1,049
GCP us-east1	137%	\$2,796	\$1,393
	M200 (4x3-node RS, 64v0	CPU, 256GB RAM, 2TB stora	ge)
AWS us-east1	156%	\$355,086	\$159,674
Azure centralus	171%	\$366,672	\$162,575
GCP us-east1	138%	\$379,376	\$198,303



Cloud and region	MongoDB Atlas	laaS pay-as-you-go	laaS reserved (3y)	
	M30 (3-node RS, 2v0	PU, 8GB RAM, 40GB disk)	'	
AWS us-east1	392%	\$2,666	\$1,204	
Azure centralus	501%	\$2,450 \$1		
GCP us-east1	276%	\$2,796	\$1,393	
	M200 (4x3-node RS, 64v	CPU, 256GB RAM, 2TB stora	ge)	
AWS us-east1	349%	\$355,086	\$159,674	
Azure centralus	386%	\$366,672	\$162,575	
GCP us-east1	265%	\$379,376	\$198,303	



# Ease of use (and scale) trap

- Cloud resources are virtually unlimited
- Pay-as-you go
- Less careful planning
- Building 20-shard 5-node each cluster?

Easy!



#### What's the best tool to resolve MongoDB performance issues quickly?





# Resolving problems with a credit card

- Over Indexing
- Suboptimal shard keys
  - Unbalanced shards
  - Queries spanning all shards
- No data archival





# (Couple of) sizes fit all

- Any upgrade costs a lot (double!)
- Only 10 production-grade tiers
  - Some with low-cpu options
  - $\circ$  ~ Some with NVMe options

м50	8vCPUs	32	\$2.16/h					
<b>↓+97%</b>								
M60	16vCPUs	64	\$4.27/h					
<b>₽+85%</b>								
M80	32vCPUs	128	\$7.90/h					

MongoDB Atlas, 3-node RS, AWS eu-north-1



# **Cloud DBaaS topologies flexibility**

- Lack of arbiter nodes
- No way to differentiate nodes
- Can't run single node



# Data transfer costs



- Hidden
- Hard to estimate
- Hard to verify
- Inter AZ -> Cross AZ -> Cross Region -> Internet
- AWS: \$0.01 \$0.09
- Azure: \$0.01 \$0.18
- GCP: \$0.01 \$0.23



# Backups

#### AWS

- Snapshots: \$0.05GB/month
- S3: \$0.023GB/month

#### MongoDB Atlas on AWS

- Snapshots: \$0.19GB/month
- PITR: \$1.55 \$0.40 (!)



# Suboptimal configuration

- Overprovisioning config replica set nodes
- Too many shards for the dataset
- Lack of tuning and sticking to defaults
- Multiple mongod services on a single node with a default cache options



# Suboptimal configuration





# "Enterprise-grade" features needs

- LDAP Authentication/Authorization
- Data at rest encryption
- Own encryption key management
- Audit log
- Advanced backups
- Support?
- Automation for scale



# Cloud DBaaS lock-in







# II. K8s and Operators 5 min crash course

# What is Kubernetes?



- Worker nodes
- Pods
- StatefulSets
- ReplicaSets
- PersistentVolumes
- PersitentVolumeClaims
- Secrets



# MongoDB on K8s



- 1. Underlying Kubernetes
- 2. StatefulSet for Primaries/Replicas/Mongo Config
- 3. High Availability and failover
- 4. DR
- 5. Storage (PVC/Hostpath)
- 6. Endpoints
- 7. Configuration
- 8. Monitoring
- 9. Backups
- 10. Upgrades
- 11. Encryption
- 12. ....



# What is Kubernetes Operator?





# What is Kubernetes Operator?

- Operator controls database and k8s primitives
- Day-1 simplified to one step
- Day-2 operations automated





# What is Kubernetes Operator?

- 1. Deploy easily: replica sets, shards, (mongo/d/s/c)
- 2. Topology management (arbiters, node affinity, scaling)
- 3. Monitoring integration
- 4. Network exposure and load balancing
- 5. Backups management with Percona Backup for MongoDB
- 6. Self-healing
- 7. Upgrade automation (minor, manual major)
- 8. Configuration adjustments



#### PERCONA

**Kubernetes Operators** 











VMware Tanzu



# How to use it?

```
apiVersion: psmdb.percona.com/v1
kind: PerconaServerMongoDB
metadata:
  name: percona-live-cluster
spec:
  crVersion: 1.15.0
  image: percona/percona-server-mongodb:6.0.4-3
  secrets:
    users: minimal-cluster
  replsets:
  - name: shard1
    size: 3
    resources:
      limits:
        cpu: "4"
        memory: "8G"
      requests:
        cpu: "4"
        memory: "8G"
    volumeSpec:
      persistentVolumeClaim:
        resources:
          requests:
            storage: 30Gi
```

sharding: enabled: true configsvrReplSet: size: 3 resources: limits: cpu: "2" memory: "4G" requests: cpu: "2" memory: "4G" volumeSpec: persistentVolumeClaim: resources: requests: storage: 3Gi mongos: size: 1





# III. Practical approaches to cost reduction

## Use open source

- 1. Advanced backups
- 2. LDAP

3.

- Data-at-rest encryption
- 4. KMIP integration
- 5. Auditing
- 6. Monitoring



**PERCONA** Distribution for MongoDB

Features



#### Enterprise Advanced/Atlas





# K8s resources are cost-efficient

- Managed K8s is ~\$70/month at any major cloud provider
- Utilize "raw" resources at "raw" resource prices
- Embrace discounts:
  - Reserved prices (41% 1 year, 62% 3-year)
  - Spot instances



# K8s resources are cost-efficient

#### Spot instances



Source: https://us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#SpotInstances:



# Embrace the variety of available cloud compute resources

- a. Create node groups with instance types that fit your workload
- b. Utilize multiple node groups
- c. Choose CPU/memory ratio wisely



# Embrace the variety of available cloud compute resources

eks.amazonaws.com/capacityType=ON\_DEMAND eks.amazonaws.com/nodegroup=critical node.kubernetes.io/instance-type=r5.2xlarge topology.kubernetes.io/region=eu-west-1 topology.kubernetes.io/zone=eu-west-1b



# Assign resources granularly

### 4 CPU -> 8 CPU -> 16 CPU -> 32 CPU 16GB -> 32GB -> 64GB -> 128 GB



© Copyright 2023 Percona® LLC. All rights reserved

# Assign resources granularly

### 4 CPU -> 8 CPU -> 16 CPU -> 32 CPU 16GB -> 32GB -> 64GB -> 128 GB

# Im (0.001 CPU) and 1 (CPU) k, M, G, T...

# 12,5 CPU and 19GB RAM



© Copyright 2023 Percona® LLC. All rights reserved

# Assign resources granularly

```
replsets:
- name: shard1
  size: 3
  resources:
    limits:
      cpu: "12.5"
      memory: "19G"
    requests:
      cpu: "12.5"
      memory: "19G"
```

- 1. Each shard
  - a. Data nodes
  - b. Hidden nodes
  - c. Arbiters
- 2. Config RS
- 3. Mongos
- 4. Backup pod



# Play container tetris



#### 48 cores to schedule



# Play container tetris



#### 48 cores to schedule, 60 cores to pay for



# Play container tetris



#### 48 cores to schedule, 48 cores to pay for



# Autoscaling



- 2. HPA/KEDA
- 3. Cluster Autoscaler

	k8s node 1 - r5.4xlarge	k8s node 2 - r5.4xlarge	
Kubernetes	Node capacity DB container 1 underutilized	Node capacity DB container 2 underutilized	2 nodes - r5.4xlarge 2 containers, underutilized



xtx	AZ-1	AZ-2	AZ-3	
	Primary	Secondary	Secondary	

spec:

allowUnsafeConfigurations: false

```
replsets:
    - name: shard1
    size: 3
    affinity:
        antiAffinityTopologyKey:
        "kubernetes.io/zone"
```

arbiter: enabled: false size: 1



stx	AZ-1	AZ-2	AZ-3	
	Primary	Secondary	Arbiter	

spec:

allowUnsafeConfigurations: false

```
replsets:
    - name: shard1
    size: 3
    affinity:
        antiAffinityTopologyKey:
        "kubernetes.io/zone"
```

arbiter: enabled: true size: 1



x	AZ-1	AZ-2	AZ-3	
92	Primary			
	Secondary			
	Secondary			

spec:

allowUnsafeConfigurations: false

```
replsets:
   - name: shard1
   size: 3
   affinity:
     antiAffinityTopologyKey:
     "kubernetes.io/host"
```

arbiter: enabled: false size: 1



	AZ-1	AZ-2	AZ-3	
×				
	Primary			

spec:

allowUnsafeConfigurations: true

```
replsets:
   - name: shard1
   size: 1
   affinity:
     antiAffinityTopologyKey:
     "kubernetes.io/host"
```

arbiter: enabled: false size: 1



# Use multiple storage classes

kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
 name: bronze

parameters:
 type: qp2

fsType: xfs

kind: StorageClass apiVersion: storage.k8s.io/v1 metadata: name: silver parameters: type: gp3 iops: 6000 fsType: xfs

kind: StorageClass apiVersion: storage.k8s.io/v1 metadata: name: gold parameters: type: io2 iopsPerGB: 40 allowAutoIOPSPerGBIncrease: "true" fsType: xfs



# Use multiple storage classes

```
replsets:
- name: shard1
size: 3
volumeSpec:
    persistentVolumeClaim:
    storageClassName: gold
    resources:
    requests:
    storage: 300G
```



# Local NVMe storage is cheap!

#### Local storage

- i3.8xlarge
- 32 vCPU
- 244GB RAM
- 4x1900GB NVMe SSD included
- 1,650k Read + 720k Write IOPS

#### \$1,822/month

#### EBS storage

- r5.8xlarge
- 32 vCPU
- 256GB RAM
- EBS 4x600GB
- 64k IOPS each

#### \$15,915/month



# Local NVMe storage is cheap!

- 1. It's ephemeral. You lose the data when the node dies
- 2. You can't extend its size
- 3. It depends strictly on the instance type
- 4. Pods won't be automatically rescheduled

**Recommendations:** 

- 1. Always use HA mode (at least 3 nodes)
- 2. OpenEBS for simplified management
- 3. Set Write Concern properly





# Stop solving issues with credit card

- Indexes:
  - Check usage stats
  - Consider partial indexes
- Sharding key:
  - Plan it
  - Adjust it
- Data archival
  - Capped collection
  - TTL indexes





# Understanding is a key

- Understand the workload
- Review outstanding queries
- Look for underutilization, too

```
helm repo add percona
https://percona.github.io/percona-helm-charts/
helm install pmm \
--set secret.create=false \
--set secret.name=pmm-secret \
percona/pmm
```

pmm:

```
enabled: true
image: percona/pmm-client:2.37.0
serverHost: monitoring-service
```



# Understanding is a key

- Understand the workload
- Review outstanding queries
- Look for underutilization, too

۲.	and Query Analytics / Pivily Quer	liayiloo Li ~o	O Lust		
			🗎 Hon	ne 🔠 Query Analy	tics
		PMM Query Analytics 🗸			
	Filters Show Selected Reset All			🕒 Copy Link	O Add column
9		# Query v Search by Q Load		Query Count \$	Query Time
	Environment	TOTAL ESLER, BERGEREL BARRARE BARRARE	10.69 load	3.59k QPS	2.97
	Prod 96.01%	1 update sbtest1 set k=k? where id=?	2.68 load	263.74 QPS	10.15
v.	n/a 3.63%	2 update sbtest1 set c=? where id=?	2.21 load	258.02 QPS	8.57
Ŧ	Dev 0.19%	3 INSERT INTO "sbtest1" ( 'id' , 'k' , 'c' , 'pad' ) VALUES () O	1.65 load	119.41 QPS	13.81
5	Staging 0.17%	4 SELECT 'EVENT_NAME', 'COUNT_STAR', 'SUM_TIMER ()	1.46 load	1.20 QPS	1.22
Π	Cluster Show all (7)	5 insert into sbtest1 (id, k, c, pad) values(?+) ①	1.18 load	409.10 QPS	2.88
UU	MySQLCluster1 _65.47%	6 delete from sbtest1 where id=?  O Units the deduction of a location of the set of the	0.66 load	513.02 QPS	1.28
ê	pxc-80-cluster 29.43%	7 SHOW BINARY LOGS	0.38 load	0.20 QPS	1.89
		8 select c from sbtest1 where id=? الله المعادية العادية العادية المعادية الم	0.09 load	1.05k QPS	82.34
	ps-80-gr c0 0.49%	9 commit  © blisse the block has the block h	0.08 load	74.37 QPS	1.12
	PXCCluster1 dlg 0.19%	ل الله الما الله الما الله الما الله الله	0.02 load	51.04 QPS	489.03
	Replication Set Show all (7)	11 CREATE TABLE 'sbtest1' ( ID INTEGER NOT NULL AUTO ()	0.02 load	4.13 QPS	5.83
	MySQLReplSe 🕼 65.47%	1 2 3 4 5 ···· 10 > 25 / page → 1-25 of 244 items			
	pxc-80-cluster glg 29.43%				
	n/a 4.11%				



# Understanding is a key

- Understand the workload
- Review outstanding queries
- Look for underutilization, too

	器 Insight /	Home Dashb	oard ☆ ≪								0 L	est 12 hours 🗸 🤇	ରି 🗘 15m -	ê
Q,														
☆	1					17								
88		U	ed Memory Anor	naly v i	<ul> <li>Last 15 minutes</li> </ul>				High Mer	nory Used			() Last 15 minu	stes
٨														
2						ps-80-source	mariadh-105	ns-80.	replica	ps-80-ar-'	2	ns-80-ar-2	ps-80-gr-1	
0		No used me	mory change	in last 7 days	3	94.0%	85.8%	85.2%	84.8%			84.5%	84.2%	
জ্ঞ														
6														
o0o	1		ow CPU Anomali	es		1			Low CPI	J Servers			④ Last 15 minu	utes
4	nongo-config- 1 18.3%	mongo-rs1-1 16.4%	<sup>mongo-config-</sup> 2 16.1%	ps-80-replica 14.9%	mongo-rs1-3 14.8%	mongo-config-1 19.5%	mongo-con 16.8%	nfig-2	mong 16	o-rs1-1 .4%	Е	ongo-rs1-3 15.3%	mongo-rs1-2 14.8%	
	mongo-rs1-2 14.5%	mongo-config- 3 13.6%	mongors2-3 13.1%	mariadb-105 12.0%	mongo-rs2-2 11.9%	mongo-config-3 13.6%	mongo-rs 13.4%	2-3	ps-80- 12	replica .2%		ariadb-105 12.1%	mongo-rs2-2 11.6%	
۲	> COMMAND	CENTER (15 p	anels)		Concernance and									
(?)	> Service Su	mmary (1 panel												





# IV. Summary

# Why costs are getting out of control?

- 1. Convenience fee
- 2. Licenses/extra features
- 3. Suboptimal configuration/usage



# What can you do to reduce costs?

- 1. Use open source K8s Operator + PSMDB
- 2. Utilise cheap resources efficiently
- 3. Eliminate common mistakes in configuration and

usage



# What's the potential?



# What's the potential?

# It depends.



© Copyright 2023 Percona® LLC. All rights reserved

# Key takeaways

- 1. Cloud DBaaS such as MongoDB Atlas comes with a huge premium fee In some cases 5x cloud resources cost.
- 1. Hidden costs and suboptimal MongoDB configuration make the situation worse Cloud and DBaaS fees are difficult to predict and understand.
- 1. Kubernetes and Percona MongoDB Operator can handle any MongoDB workload The solution has been proven in production for many years.
- 1. Running MongoDB on K8s has a significant cost saving potential

It makes it possible to utilize cheap "raw" cloud resources without giving up convenience and automation



# Want to know more about Percona Kubernetes Operators?



# PERCONA

Databases run better with Percona