

# DEVOPS: MISE EN PLACE DE CLOUD-NATIVE STORAGE

Pablo Mercado

**Responsible:** Marcel Graf

# SOMMAIRE

1. Introduction
2. Comparatif
3. Proof of Concept
4. Analyse
5. Conclusion

# INTRODUCTION

1. Containerisation
2. Stockage
3. Stockage dans un cluster Kubernetes

# INTRODUCTION - CONTAINERISATION

*Pourquoi passer de la VM au container?*

- prix exorbitant d'une licence VMware
- overhead dû à la virtualisation du hardware

**Note:** [initiative](#) sur la souveraineté numérique en cours d'élaboration

# INTRODUCTION - STOCKAGE

# INTRODUCTION - STOCKAGE

Jens Axboe?

# INTRODUCTION - STOCKAGE

Jens Axboe?

Twitter

- bloc layer maintainer du Linux kernel
- outil de benchmark de stockage: FIO
- FIO est open source et régulièrement mis à jour

# INTRODUCTION - STOCKAGE

Jens Axboe?

Twitter

- bloc layer maintainer du Linux kernel
- outil de benchmark de stockage: FIO
- FIO est open source et régulièrement mis à jour

FIO pour benchmark du CAS (Container attached storage)?



# INTRODUCTION - STOCKAGE DANS UN CLUSTER KUBERNETES

# INTRODUCTION - STOCKAGE DANS UN CLUSTER KUBERNETES

- Kubernetes ne fournit pas de stockage persistant par défaut
- CSI permet une implémentation de PersistentVolume
- le PersistentVolume est monté dans le Filesystem du container

CSI: Container Storage Interface

# INTRODUCTION - STOCKAGE DANS UN CLUSTER KUBERNETES

- Kubernetes ne fournit pas de stockage persistant par défaut
- CSI permet une implémentation de PersistentVolume
- le PersistentVolume est monté dans le Filesystem du container

CSI: Container Storage Interface

Quelle implémentation choisir?

# COMPARATIF

1. Longhorn (le StorageClass par défaut du cluster de l'IICT)
2. EBS (cloud provider: AWS)
3. Autres solutions de stockage



## COMPARATIF - LONGHORN







- installation facile depuis Rancher
- offre un système de backup
- open source



## COMPARATIF - EBS

- AWS est le cloud provider le plus populaire
- offre un système de backup
- propriétaire

# COMPARATIF - AUTRES SOLUTIONS DE STOCKAGE (1/2)

|  |   |  |
|--|---|--|
| <br><b>ChubaoFS</b><br>ChubaoFS<br>Cloud Native Computing Foundation (CNCF)<br>★ 2,315<br>Funding: \$3M       | <br><b>LONGHORN</b><br>Longhorn<br>Cloud Native Computing Foundation (CNCF)<br>★ 3,066<br>Funding: \$3M | <br><b>OpenEBS</b><br>OpenEBS<br>Cloud Native Computing Foundation (CNCF)<br>★ 7,074<br>Funding: \$3M |
| <br><b>Piraeus</b><br>Piraeus Datastore<br>Cloud Native Computing Foundation (CNCF)<br>★ 265<br>Funding: \$3M | <br><b>ROOK</b><br>Rook<br>Cloud Native Computing Foundation (CNCF)<br>★ 8,954<br>Funding: \$3M         | <br><b>Vineyard</b><br>Vineyard<br>Cloud Native Computing Foundation (CNCF)<br>★ 499<br>Funding: \$3M |

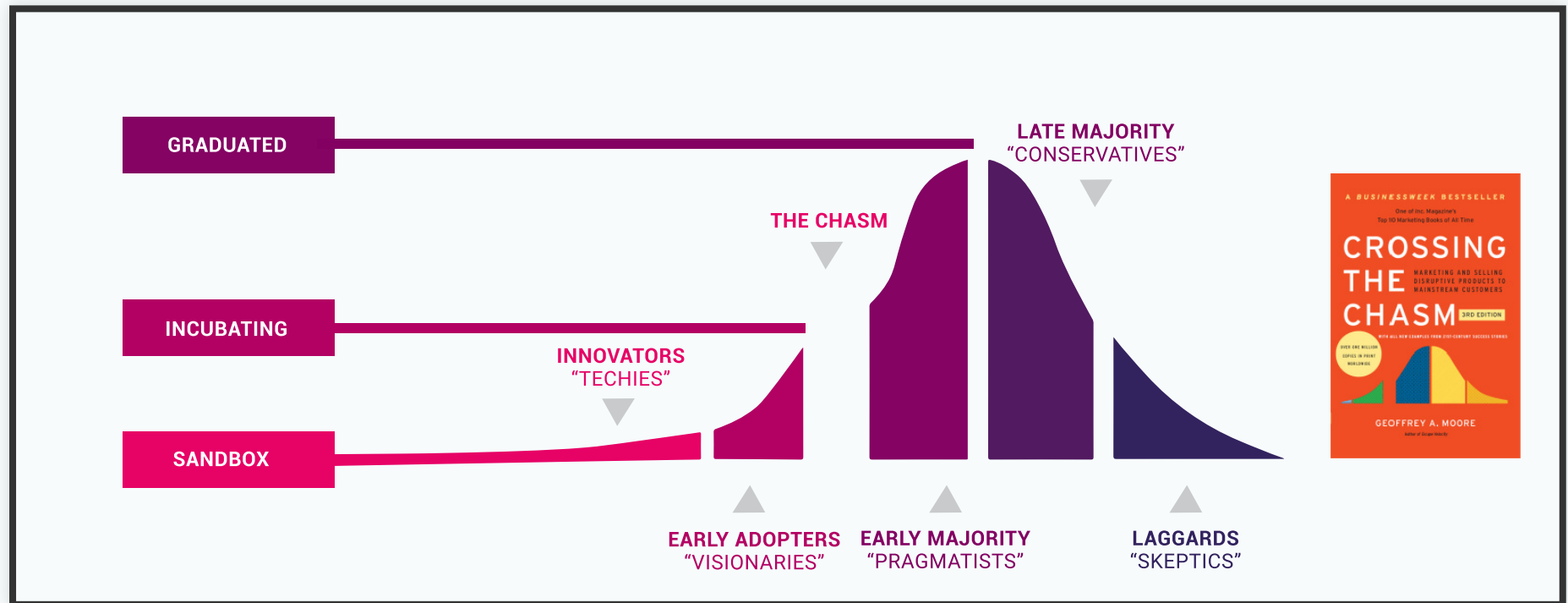
**Source:** [CNCF](#) (Date de consultation: 6 septembre 2021)

**Note:** [iCoSys](#) (Fribourg) utilise Piraeus Datastore

# COMPARATIF - AUTRES SOLUTIONS DE STOCKAGE (2/2)

D'après le rapport technique d'Architecting-IT (méthodologie suivant dbench)

- OpenEBS a de mauvaises performances malgré sa popularité (github ★)
- Rook+Ceph est plus performant que Longhorn
- StorageOS n'est pas un projet supporté par la CNCF





# PROOF OF CONCEPT

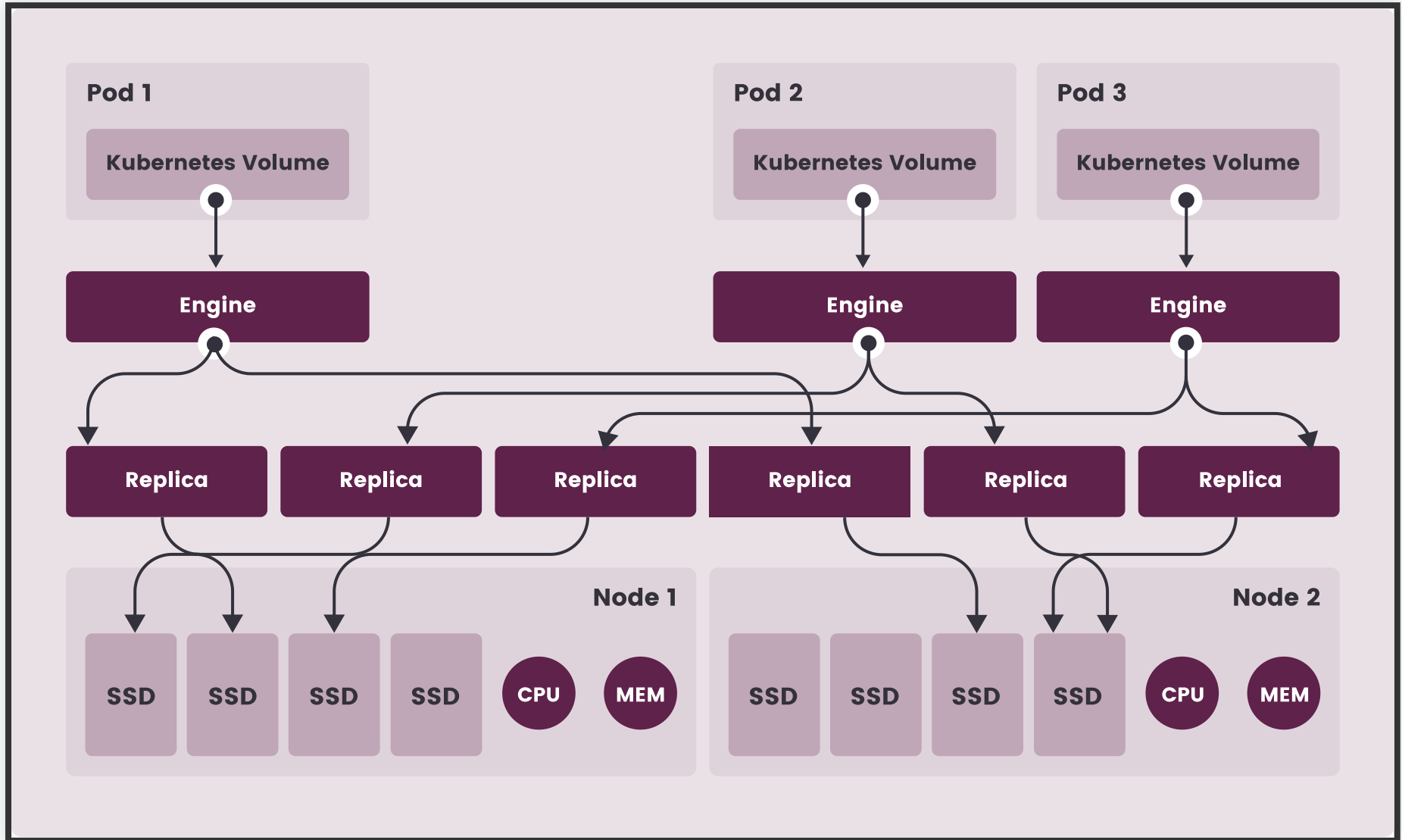
1. Benchmark de SSD
2. Fonctionnement de Longhorn
3. Description des jobs FIO
4. Configuration cluster EKS
5. Exemple de manifest

# POC - BENCHMARK DE SSD

SSD sous test: *SAMSUNG MZVLB1T0HBLR-000H1*

| Environnement       | 1                         | 2                   |
|---------------------|---------------------------|---------------------|
| OS                  | Pop!_OS (basé sur Ubuntu) | Windows 10          |
| Outil de benchmark  | FIO                       | Samsung Magician    |
| open source (outil) | ✓                         | ✗                   |
| Random read         | <b>954'488 IOPS</b>       | 226'806 IOPS        |
| Random write        | 88'031 IOPS               | <b>196'533 IOPS</b> |

# POC - FONCTIONNEMENT DE LONGHORN





# POC - DESCRIPTION DES JOBS FIO (1/3)

Quel profil de workload?

# POC - DESCRIPTION DES JOBS FIO (1/3)

Quel profil de workload?

- lecture/écriture séquentielle
- lecture/écriture aléatoire
- profil hybride

# POC - DESCRIPTION DES JOBS FIO (1/3)

Quel profil de workload?

- lecture/écriture séquentielle
- lecture/écriture aléatoire
- profil hybride

Quelles métriques?

# POC - DESCRIPTION DES JOBS FIO (1/3)

Quel profil de workload?

- lecture/écriture séquentielle
- lecture/écriture aléatoire
- profil hybride

Quelles métriques?

- IOPS
- bande passante
- latence



# POC - DESCRIPTION DES JOBS FIO (1/3)

Quel profil de workload?

- lecture/écriture séquentielle
- lecture/écriture aléatoire
- profil hybride

Quelles métriques?

- IOPS
- bande passante
- latence

Pourquoi l'output en JSON?

# POC - DESCRIPTION DES JOBS FIO (1/3)

Quel profil de workload?

- lecture/écriture séquentielle
- lecture/écriture aléatoire
- profil hybride

Quelles métriques?

- IOPS
- bande passante
- latence

Pourquoi l'output en JSON?

- parsing
- données exhaustives

## POC - DESCRIPTION DES JOBS FIO (2/3)

```
1
2 [global]
3 name=read_iops
4 readwrite=randread
5
6 randrepeat=0
7 verify=0
8
9 ioengine=libaio
10 direct=1
11 gtod_reduce=1
12 filename=/data/fiotest
13 bs=4K
14 iodepth=16
15 fdatsync=0
16 size=250G
17 time_based
18 ramp_time=10s
19 runtime=30s
20
21 [architecting-it-test1-read]
```

# POC - DESCRIPTION DES JOBS FIO (3/3)

Différence de méthode d'output:

```
1
2 [global]
3 name=seq_read
4 readwrite=read
5
6 randrepeat=0
7 verify=0
8
9 ioengine=libaio
10 direct=1
11 gtod_reduce=1
12 filename=/data/fiotest
13 bs=1M
14 iodepth=16
15 fdatasync=0
16 size=250G
17 time_based
18 ramp_time=10s
19 runtime=30s
20 group_reporting=1
21
22 numjobs=4
23 offset_increment=500M
24
25 [architecting-it-test7-sequential-read]
```

# POC - CONFIGURATION CLUSTER EKS

Mesure étalon, éviter *EBS optimized*:

- Sélection d'éléments "general purpose"
  - volume EBS: gp2
  - instance EC2: t3.small
- même version de Kubernetes que le cluster IICT

# POC - EXEMPLE DE MANIFEST

```
1
2 kind: StorageClass
3 apiVersion: storage.k8s.io/v1
4 metadata:
5   name: longhorn-test
6   namespace: mercado
7 provisioner: driver.longhorn.io
8 allowVolumeExpansion: true
9 parameters:
10   numberOfReplicas: "3"
11   staleReplicaTimeout: "2880" # 48 hours in minutes
12   fromBackup: ""
```

# POC - EXEMPLE DE MANIFEST

```
1
2 # inspired from: https://raw.githubusercontent.com/mahaupt/fiobench/master/fiobench.yaml
3 kind: PersistentVolumeClaim
4 apiVersion: v1
5 metadata:
6   name: fiobench-longhorn-pvc
7   namespace: mercado
8   labels:
9     bench: image
10    deployment: "04"
11    subject: three-replica
12 spec:
13   storageClassName: longhorn-test
14   accessModes:
15     - ReadWriteOnce
16   volumeMode: Filesystem
17   resources:
18     requests:
19       storage: 1000Gi
```

# POC - EXEMPLE DE MANIFEST

```
1
2 apiVersion: apps/v1
3 kind: Deployment
4 metadata: ...
5 spec:
6   replicas: 1
7   selector:
8     matchLabels: ...
9   template:
10     metadata:
11       labels: ...
12     spec:
13       containers:
14         - name: fiobench
15           image: ghcr.io/pabloheigvd/tb-fiobench:latest
16           command: [ "/bin/sh", "run-all-jobs.sh" ]
17           volumeMounts:
18             - name: test-volume
19               mountPath: /data
20       volumes:
21         - name: test-volume
22           persistentVolumeClaim:
23             claimName: fiobench-longhorn-pvc
```

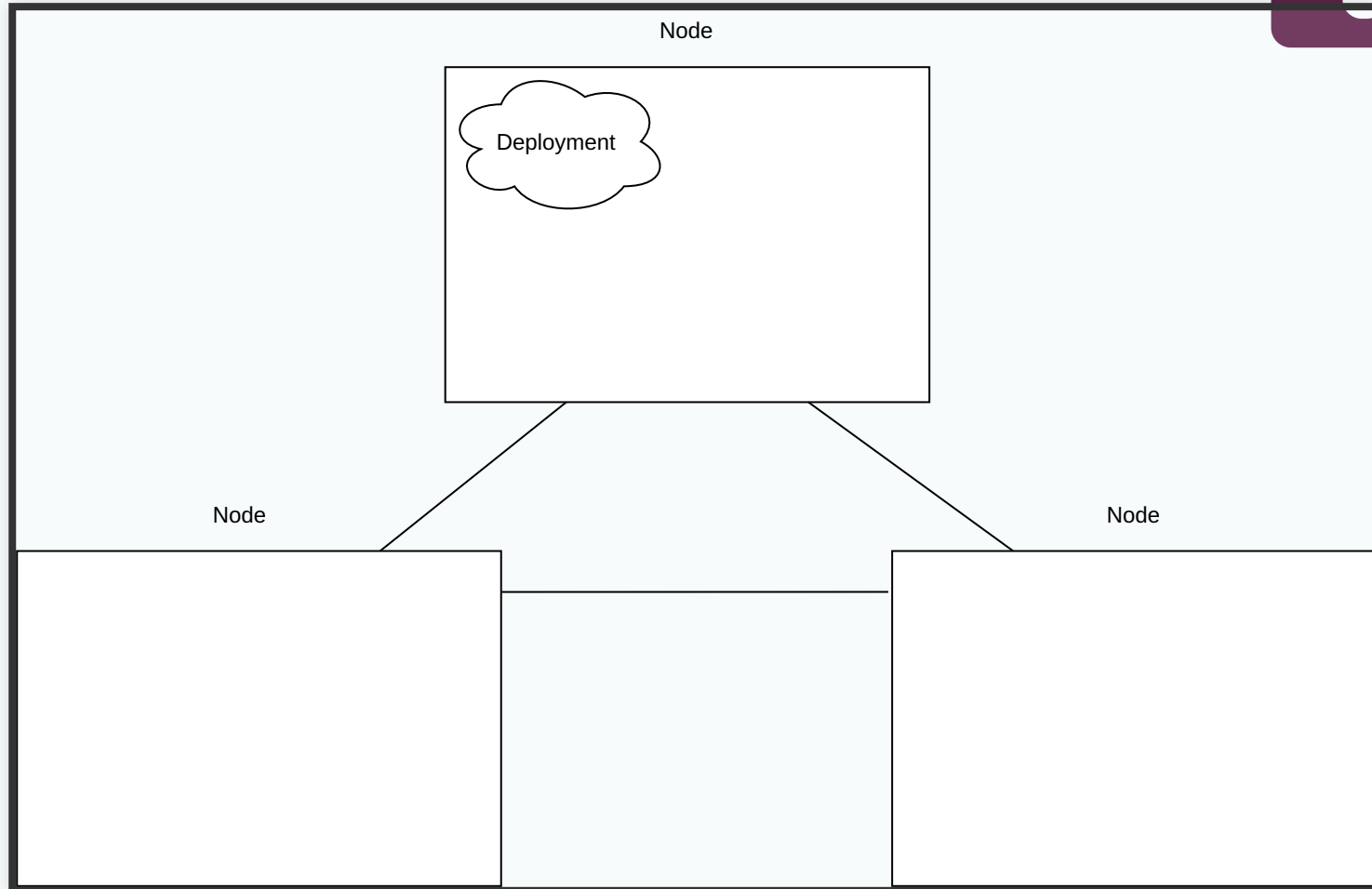
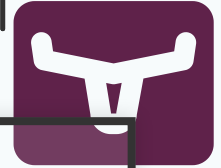


# EXAMPLE DE BENCHMARKING

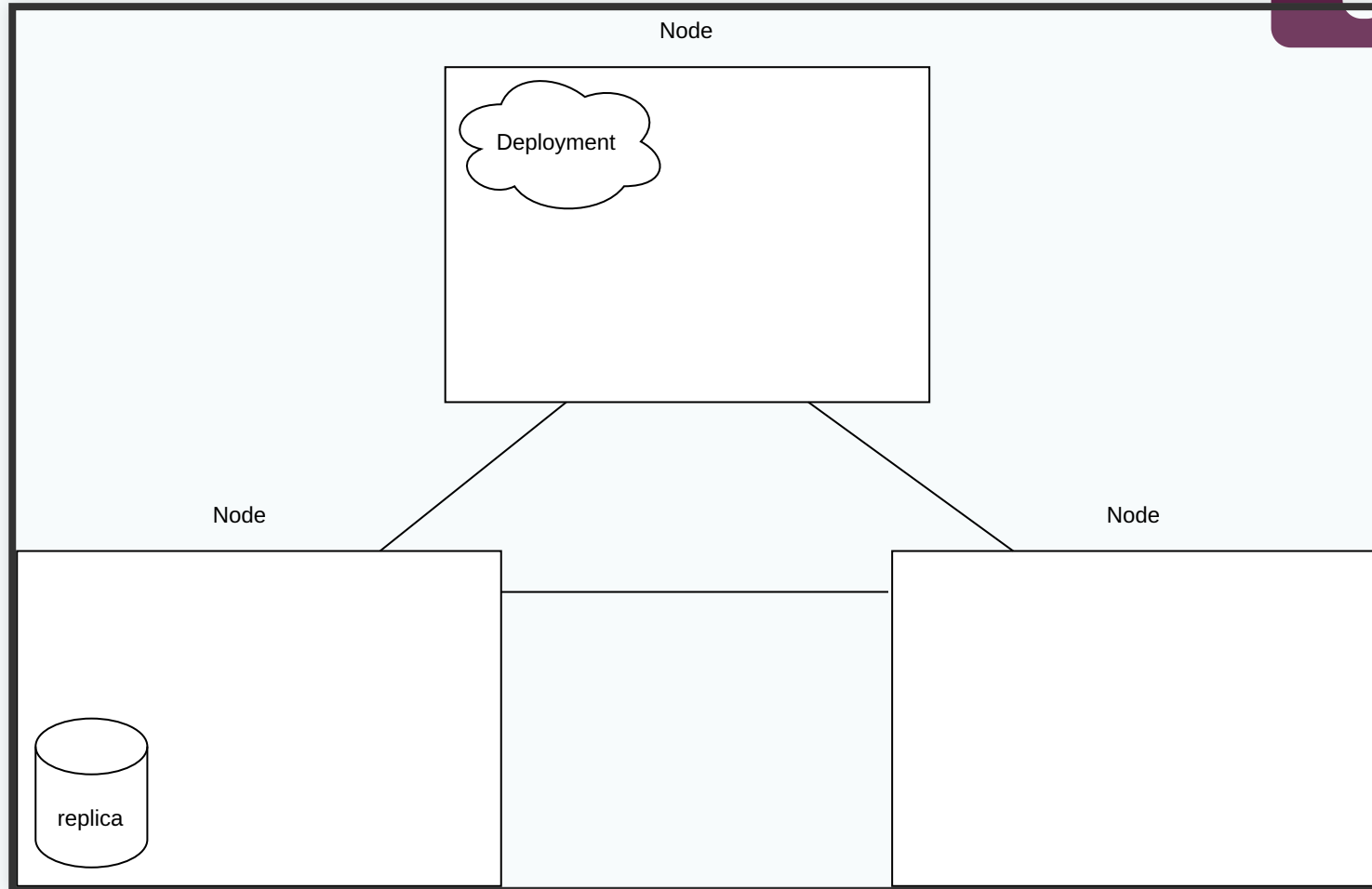
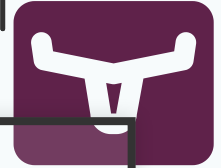
# ANALYSE

1. Configuration cluster IICT
2. Read/Write
3. Read/Write bandwidth
4. Sequential Read/Write bandwidth
5. Read/Write mix

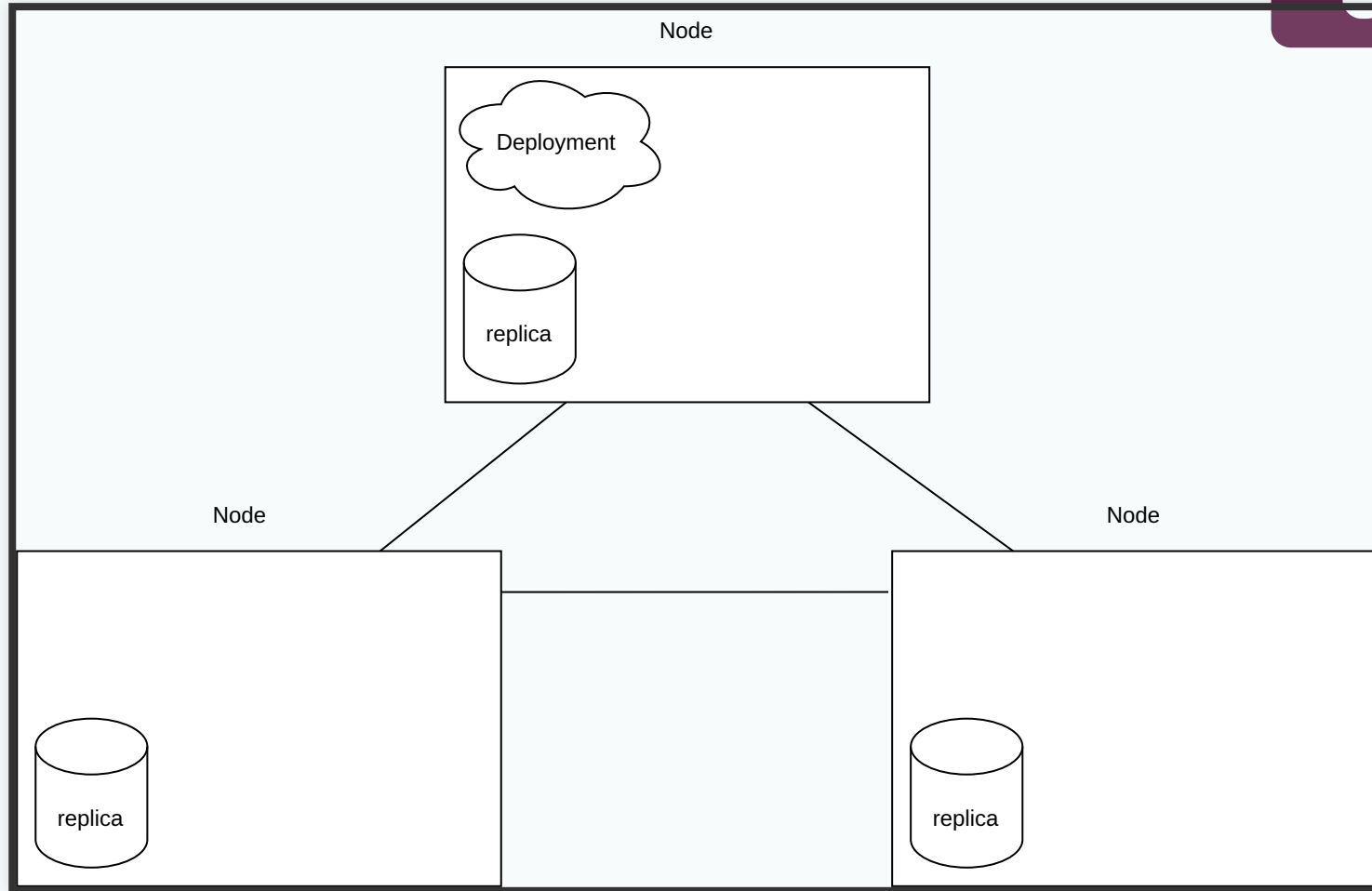
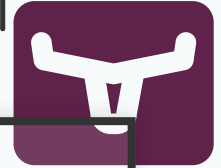
# ANALYSE - CONFIGURATION CLUSTER IICT



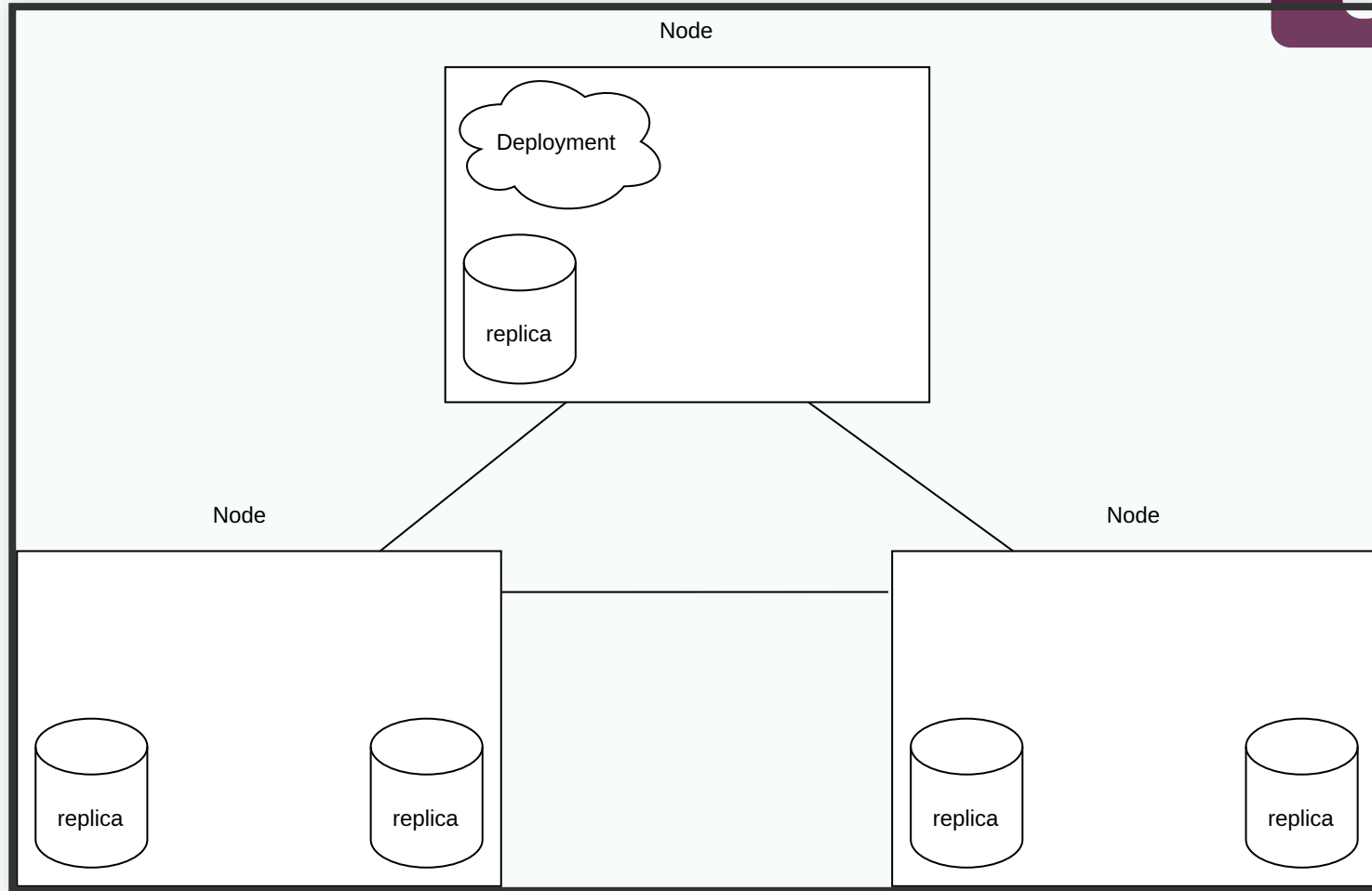
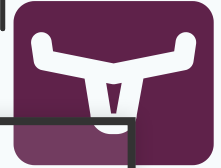
# ANALYSE - CONFIGURATION CLUSTER IIC



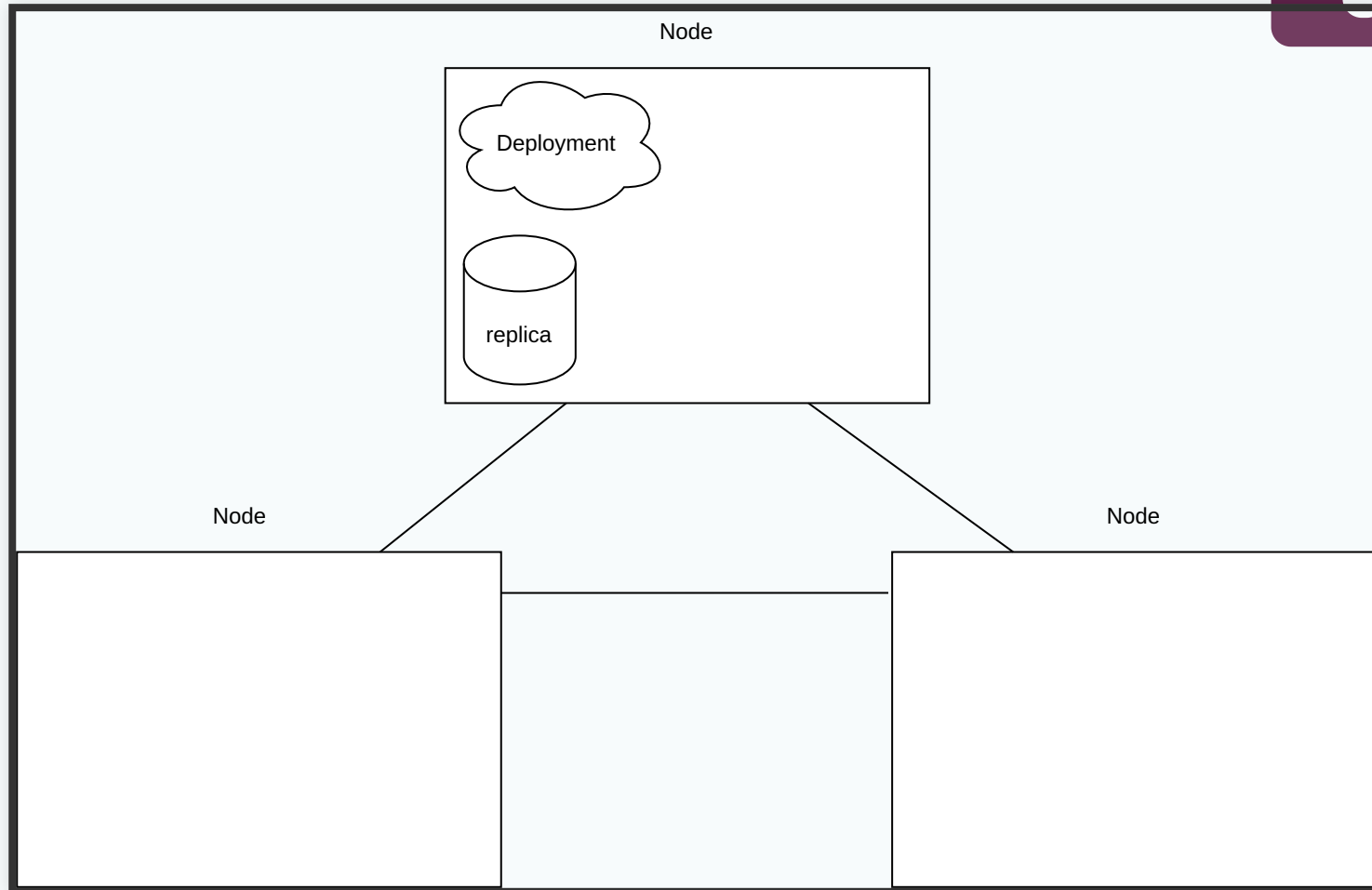
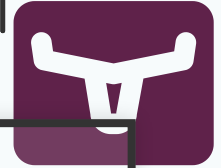
# ANALYSE - CONFIGURATION CLUSTER IIC



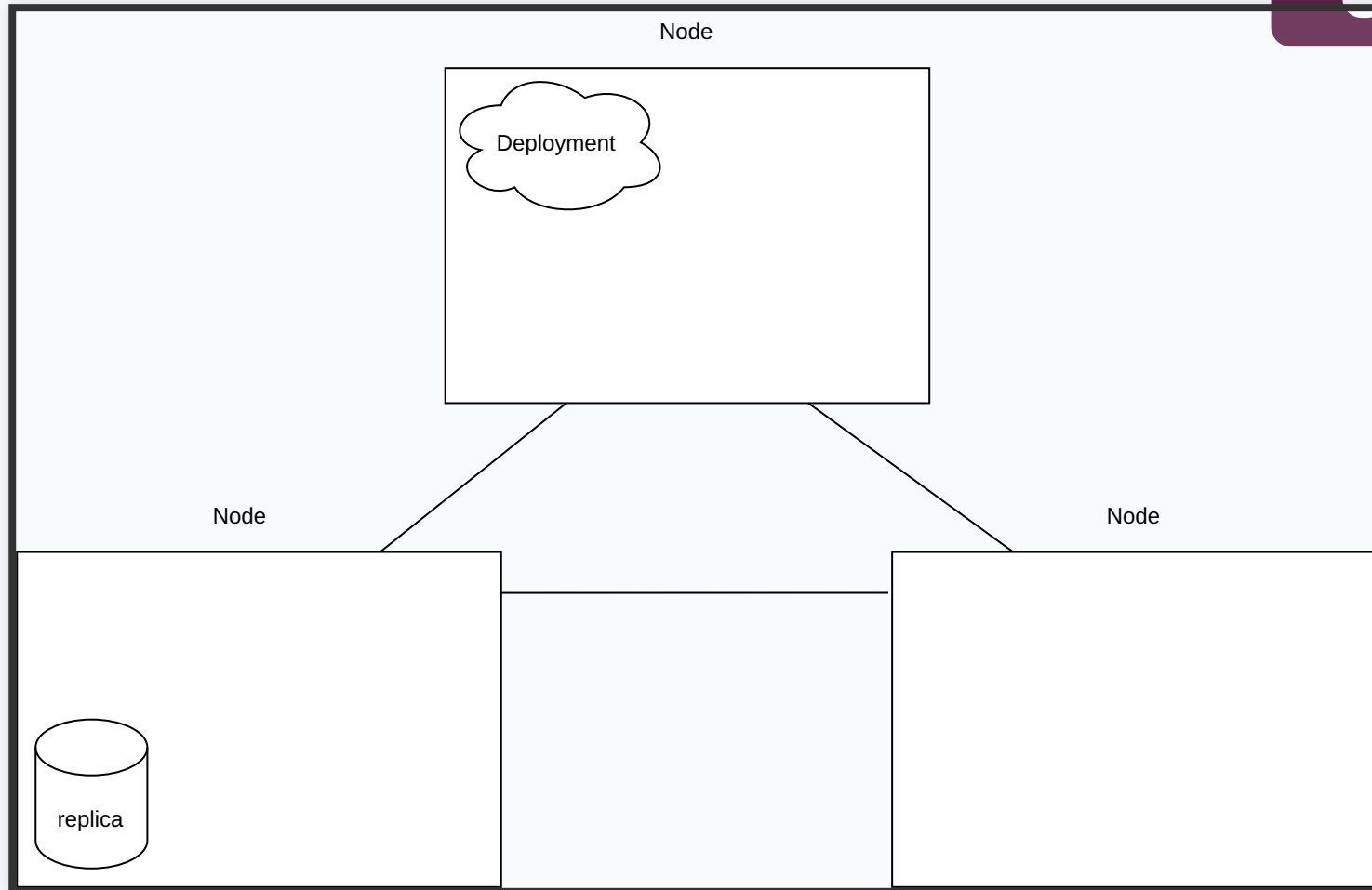
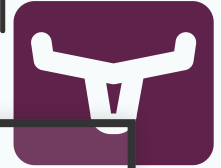
# ANALYSE - CONFIGURATION CLUSTER IIC



# ANALYSE - CONFIGURATION CLUSTER IIC

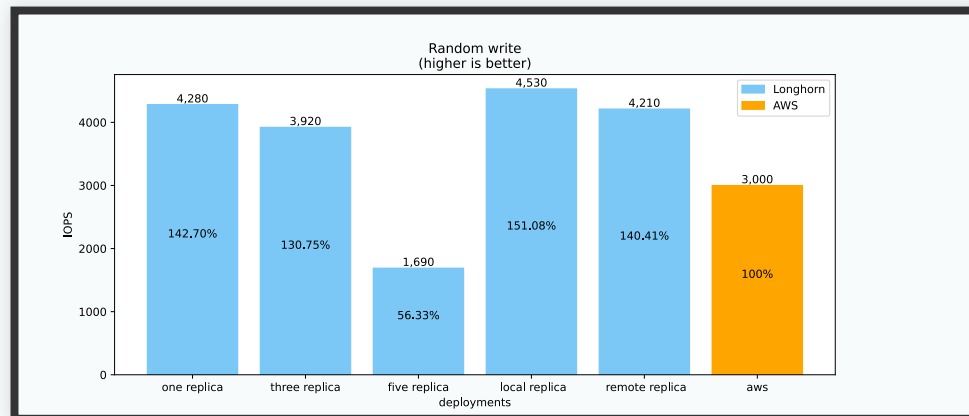
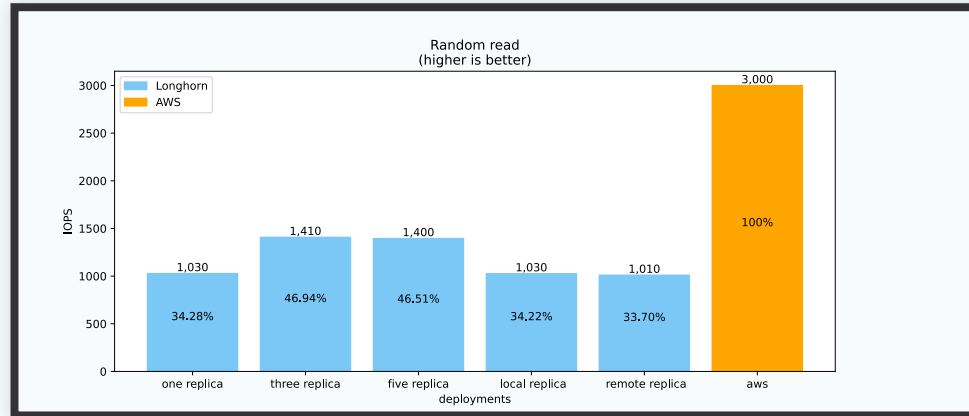


# ANALYSE - CONFIGURATION CLUSTER IIC

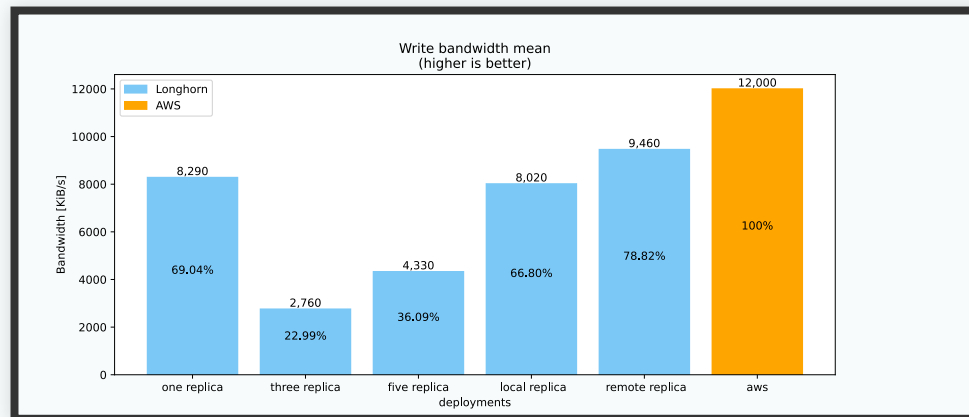
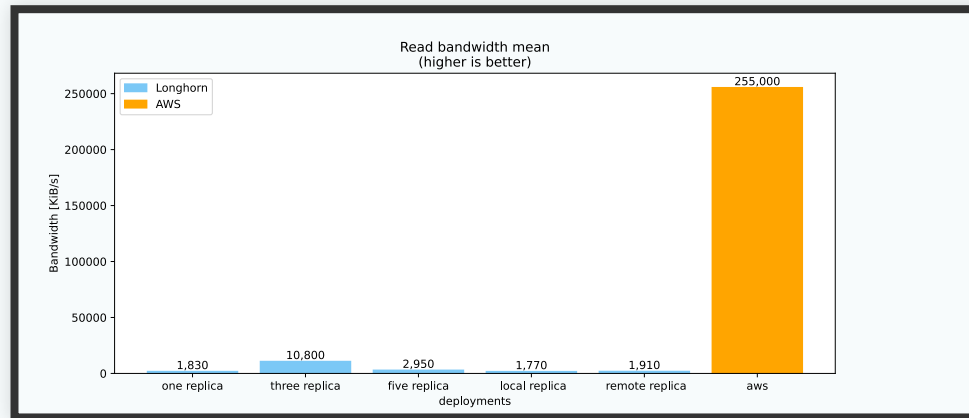




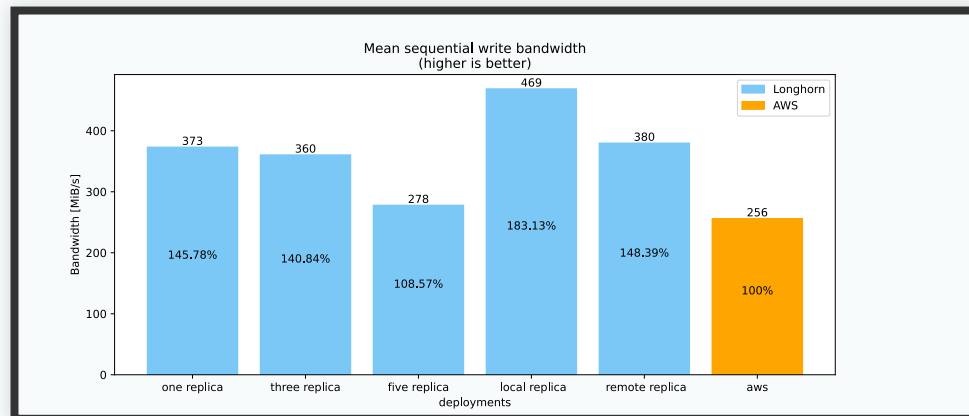
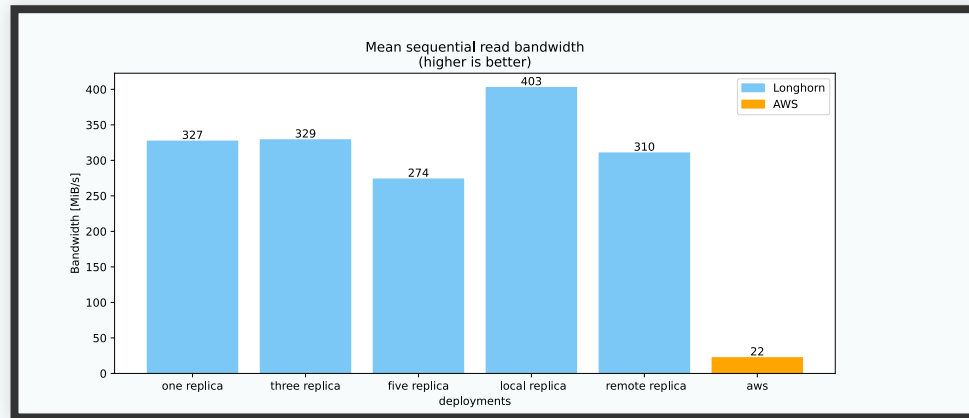
# ANALYSE - READ/WRITE



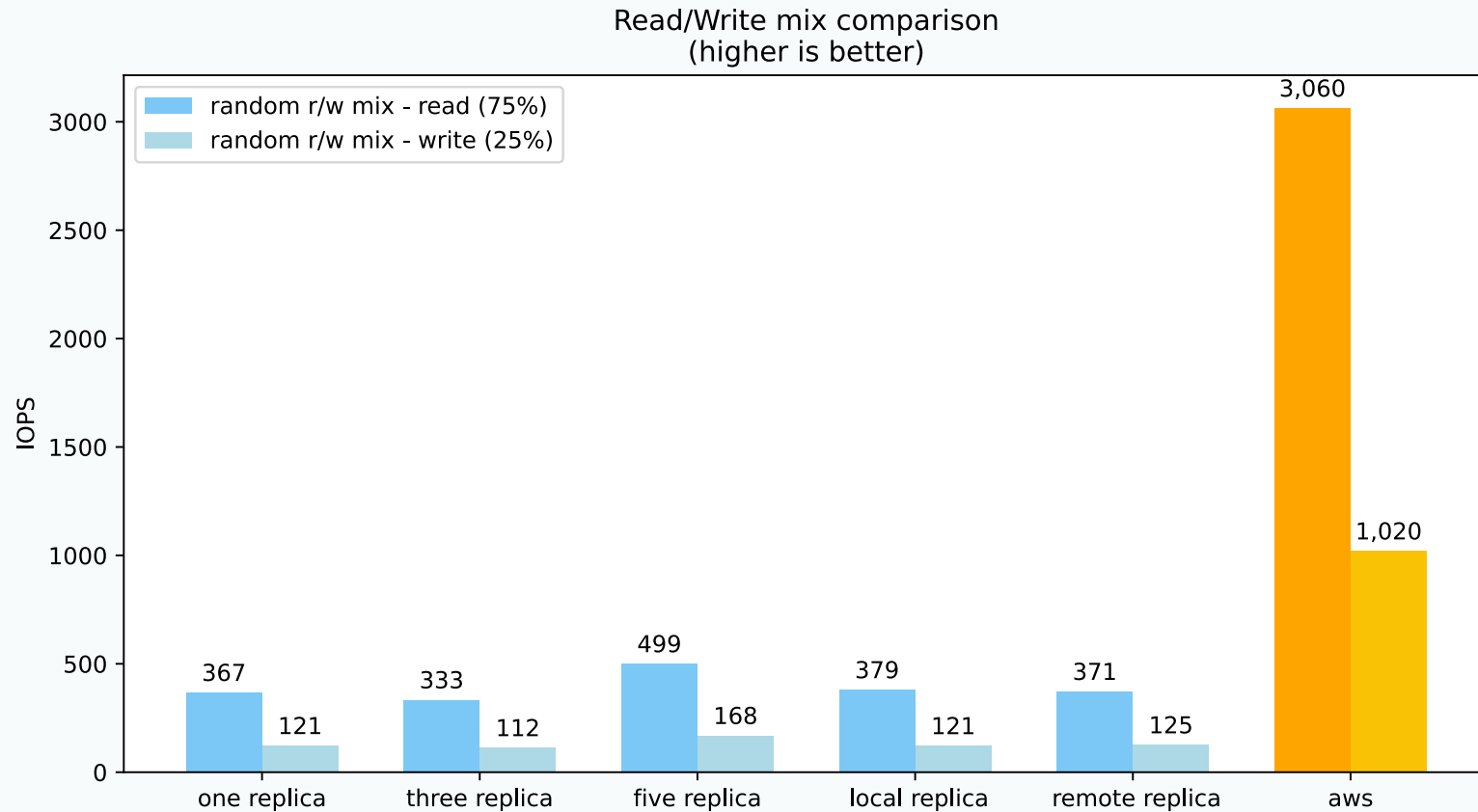
# ANALYSE - READ/WRITE BANDWIDTH



# ANALYSE - SEQUENTIAL READ/WRITE BANDWIDTH



# ANALYSE - READ/WRITE MIX



# CONCLUSION

1. Comparatif
2. Proof of Concept
3. Documentation

# CONCLUSION - COMPARATIF

- Rook+Ceph > Longhorn
- Longhorn est facile à installer
- OpenEBS a de mauvaises performances

# CONCLUSION - PROOF OF CONCEPT

Dans le cas d'utilisation général:

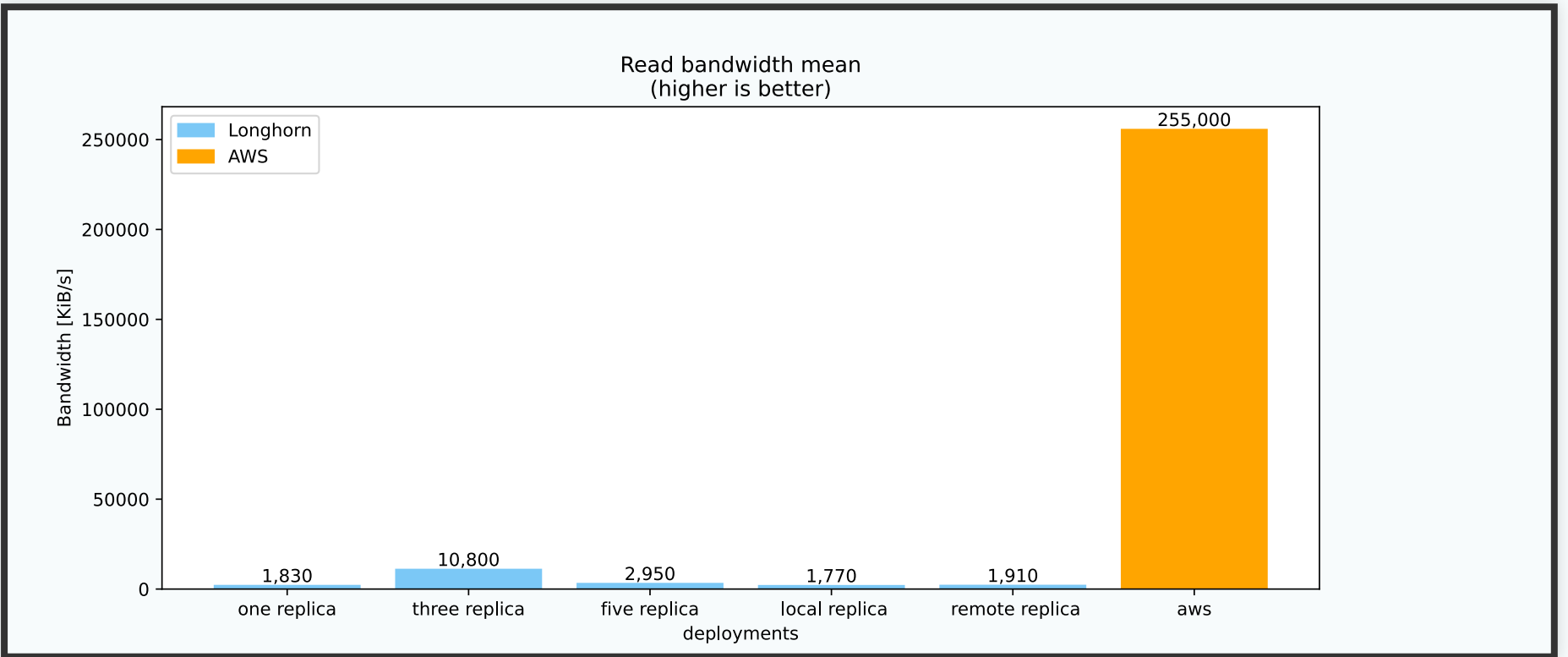
- Longhorn a des performances respectables comparé aux volumes EBS
- L'offre d'AWS est très variée
- Utiliser FIO/dbench permet un benchmark détaillé

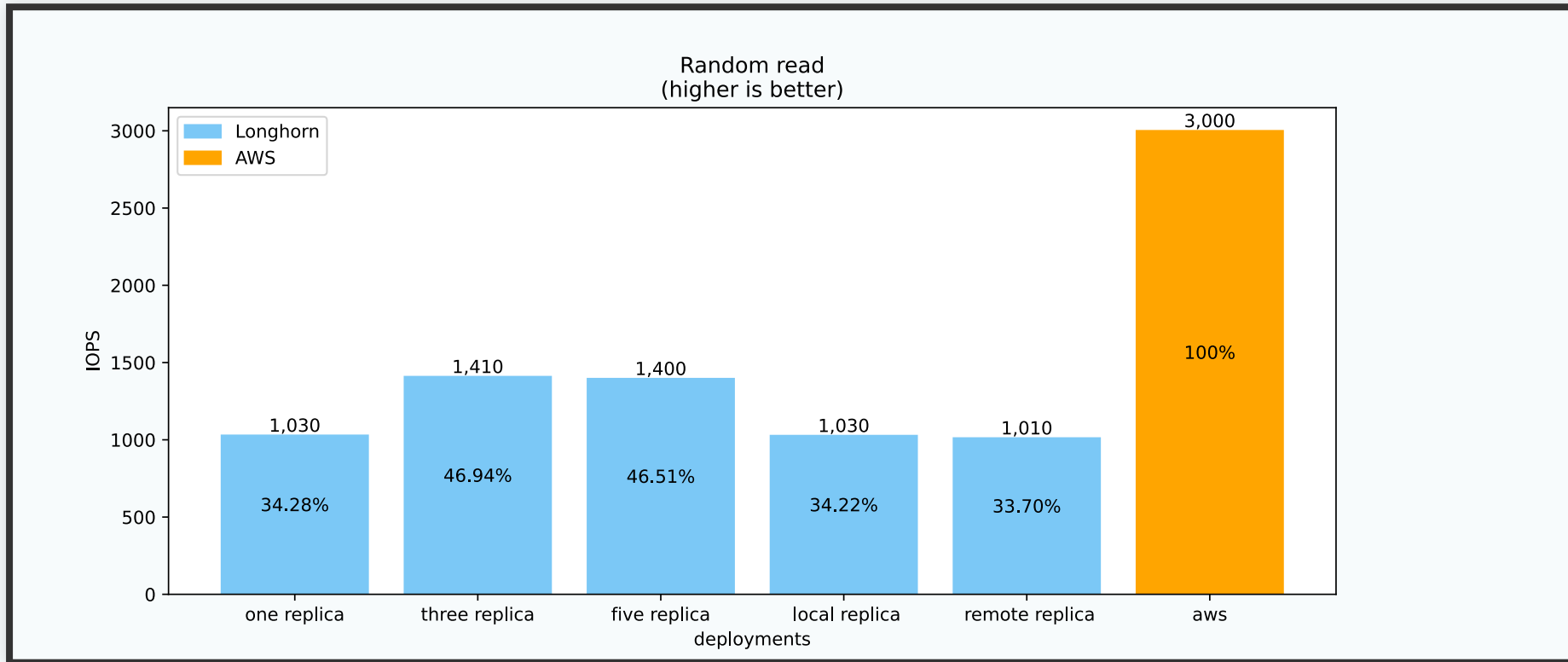
# CONCLUSION - DOCUMENTATION

- Guide utilisateur
- Jupyter notebook (graphiques)
- Rapport final

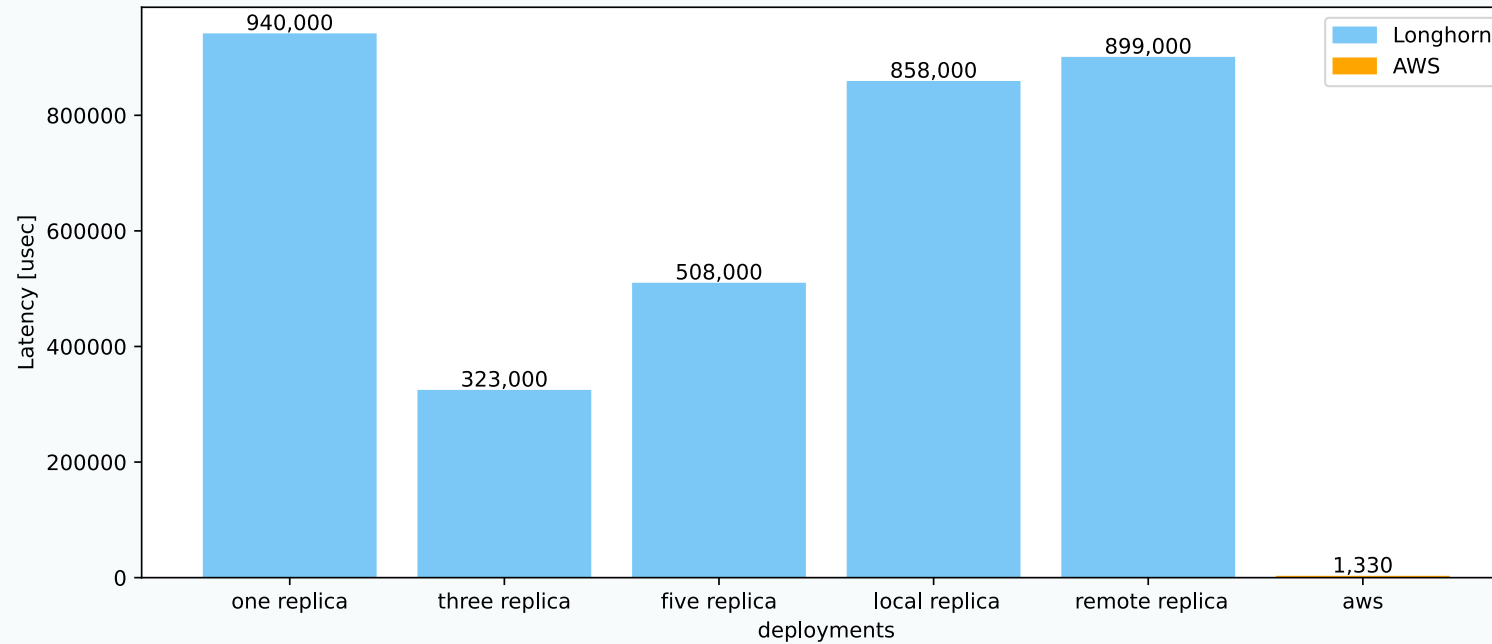


QUESTIONS ?

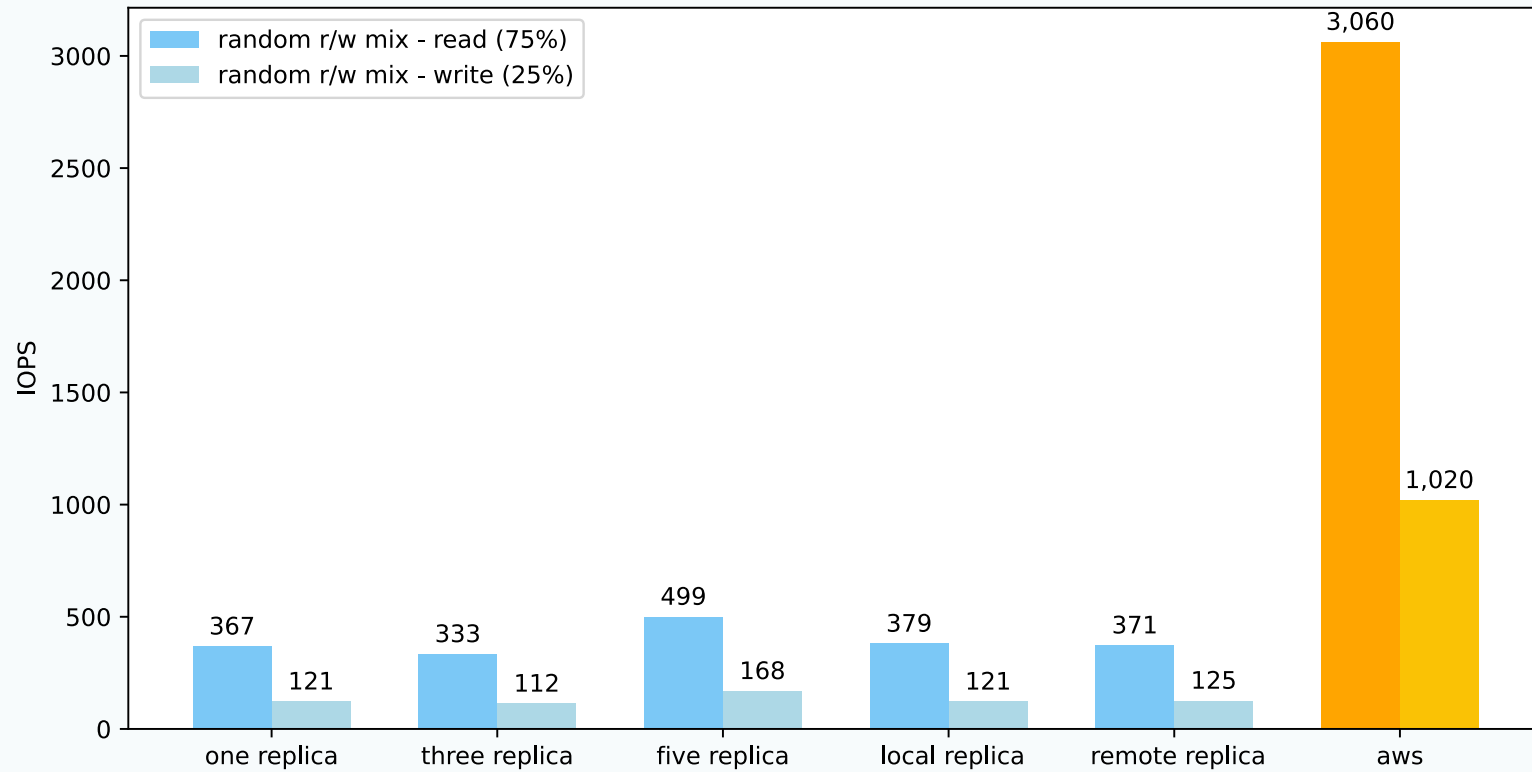




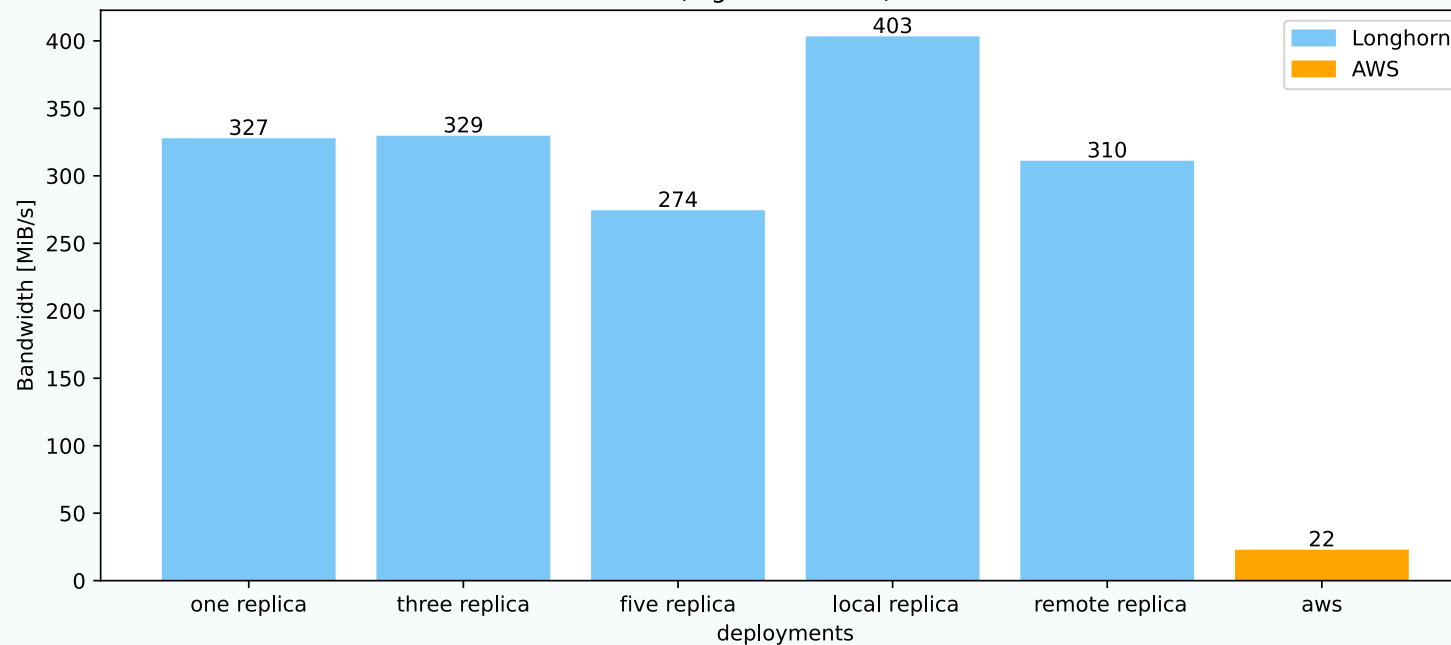
Mean read latency  
(lower is better)



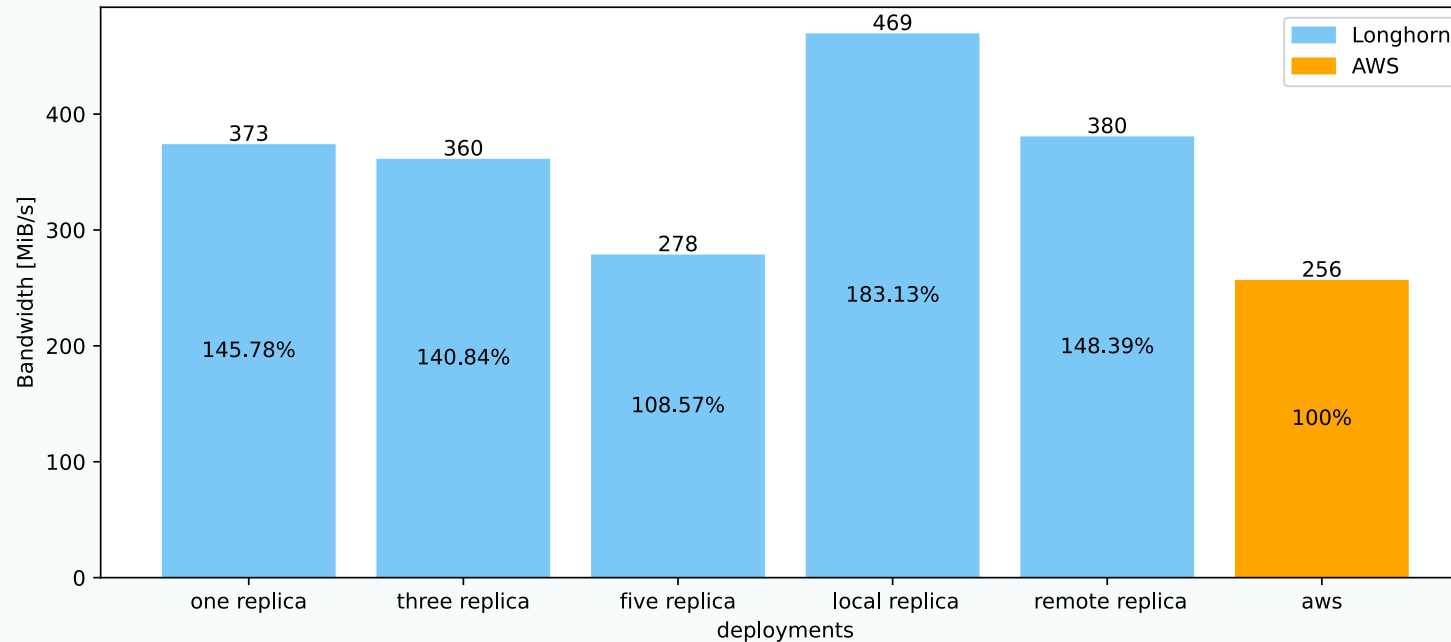
Read/Write mix comparison  
(higher is better)



Mean sequential read bandwidth  
(higher is better)



Mean sequential write bandwidth  
(higher is better)



Write bandwidth mean  
(higher is better)

