Hadoop ❤ Cloud-native

Márton Elek

cloudera

What is Apache Hadoop Ozone?

Generic **Object store** based on Hadoop Storage layer.

aws S3 protocol

Hadoop FS

CSI

Apache Hadoop Ozone

hadoop.apache.org/ozone

THE APACHE SOFTWARE FOUNDATION

“Cloud native technologies empower organizations to build and run scalable applications in modern, dynamic environments such as public, private, and hybrid clouds.” (CNCF charter)

WHAT?

Cloud native computing uses an **open source** software stack to deploy applications as **microservices**, packaging each part into its own **container**, and dynamically orchestrating those containers to **optimize resource** utilization (cncf.io)

WHAT?

Personal definition of Cloud-Native?

Márton Elek

Apache Hadoop committer
Apache Ratis (Incubating) committer / PMC

elek@apache.org
@anzix

Márton Elek

Apache Hadoop committer
Apache Ratis (Incubating) committer / PMC

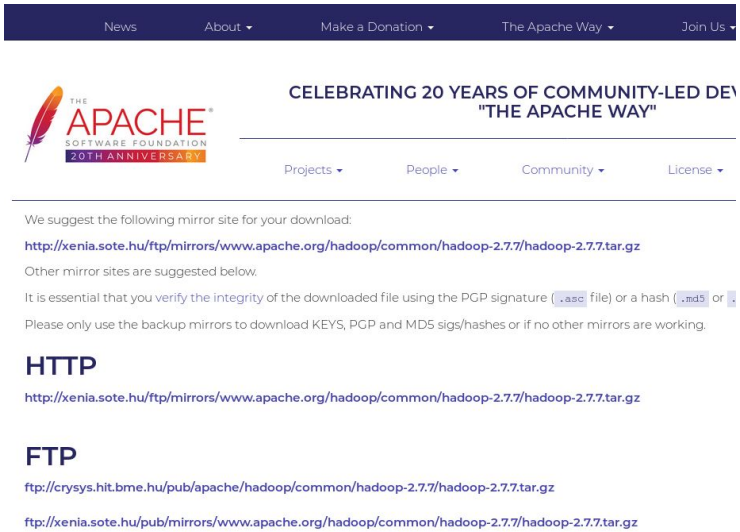
elek@apache.org
@anzix

Flokkr → https://github.com/flokkr (Like Bigtop but for Kubernetes)
Flekszible → https://github.com/elek/flekszible (Helm + Kustomize = Flekszible)

What is cloud-native (def1) ?

Hadoop 2.7 release?

- Download one tar

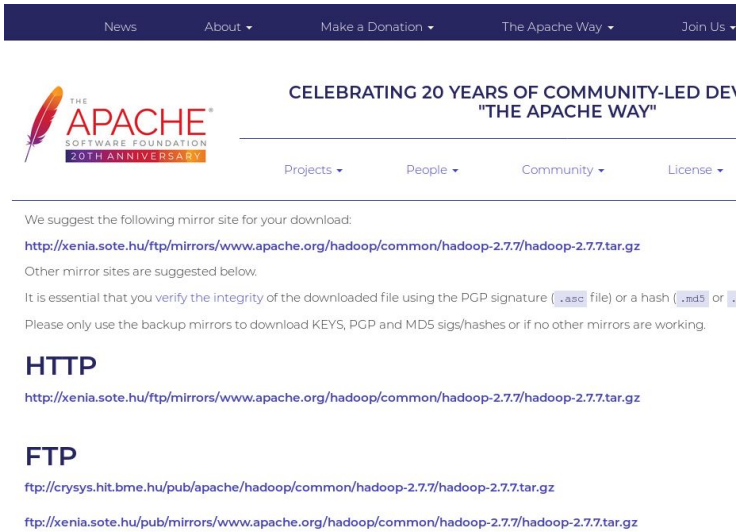


cloudera

What is cloud-native (def1) ?

Hadoop 2.7 release?

- Download one tar



Cloud-native Hadoop Ozone release?

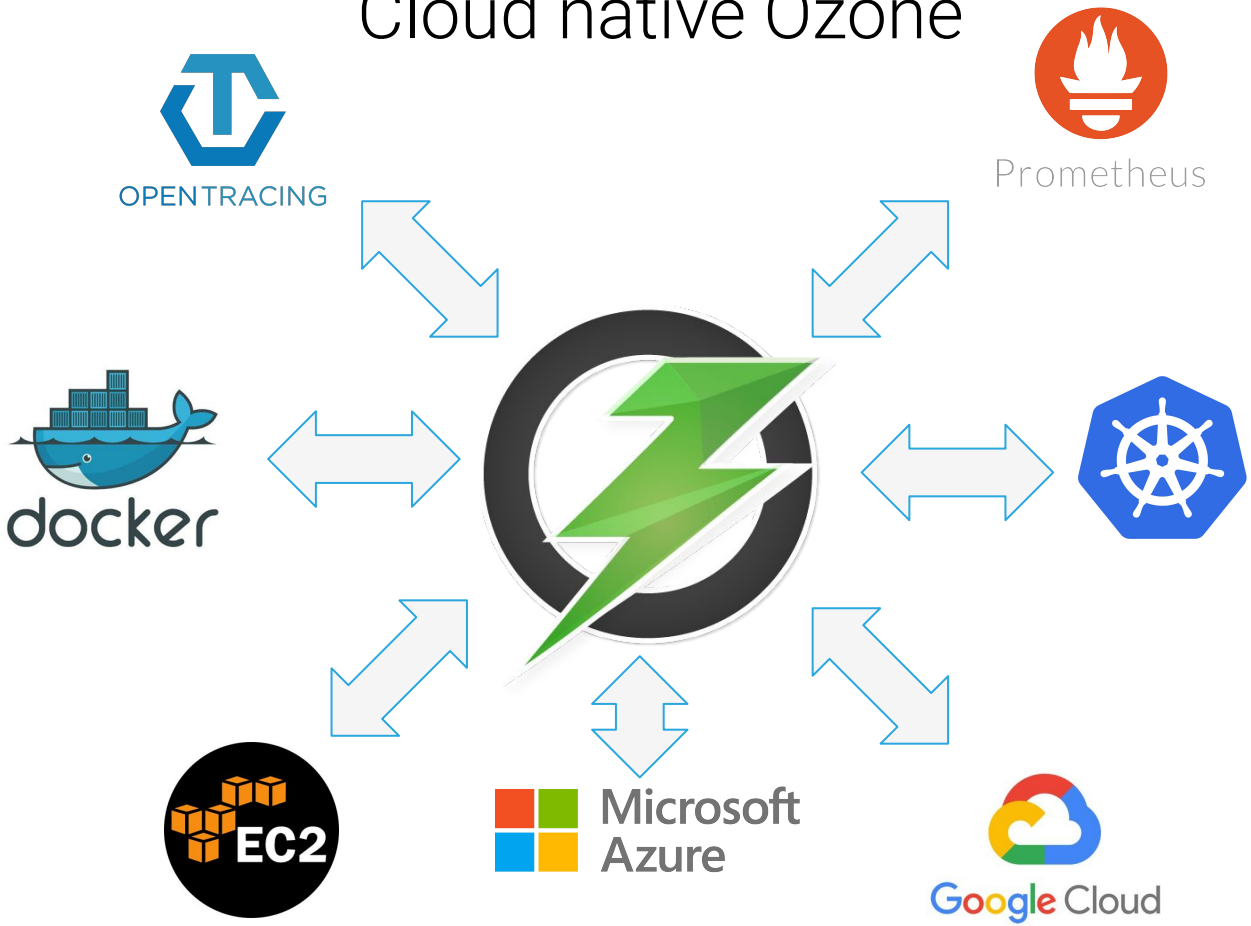
- tar xvzf + ./bin/hdfs start
- docker run apache/hadoop
- docker-compose up -d
- kubectl apply -f

...

Make it easy to start anywhere!

cloudera

Cloud native Ozone



Easy Start

Running Ozone from Docker Hub

You can try out Ozone using docker hub without downloading the official release. This makes it easy to explore Ozone.

Starting ozone inside a single container

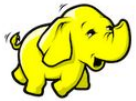
The simplest and easiest way to start an ozone cluster to explore what it can do is to start ozone via docker.

Ozone in Docker

Recommended

Running Ozone from an Offical Release.

Apache Ozone can also be run from the official release packages. Along with the official source releases, we also release a set of convenience binary packages. It is easy to run these binaries in different configurations.



Deploying Ozone on a physical cluster.

Ozone is designed to work concurrently with HDFS. The physical cluster instructions explain each component of Ozone and how to deploy with maximum control.

On-Prem Ozone Cluster

Deploying Ozone on K8s

Ozone is designed to work well under kubernetes. These are instructions on how to deploy Ozone on K8s platform. Ozone provides a replicated storage solution for K8s based Applications.

Ozone on Kubernetes

Deploy Ozone using MiniKube.

Ozone comes with a standard set of K8s resources. You can deploy them to MiniKube and experiment with the K8s based deployments.

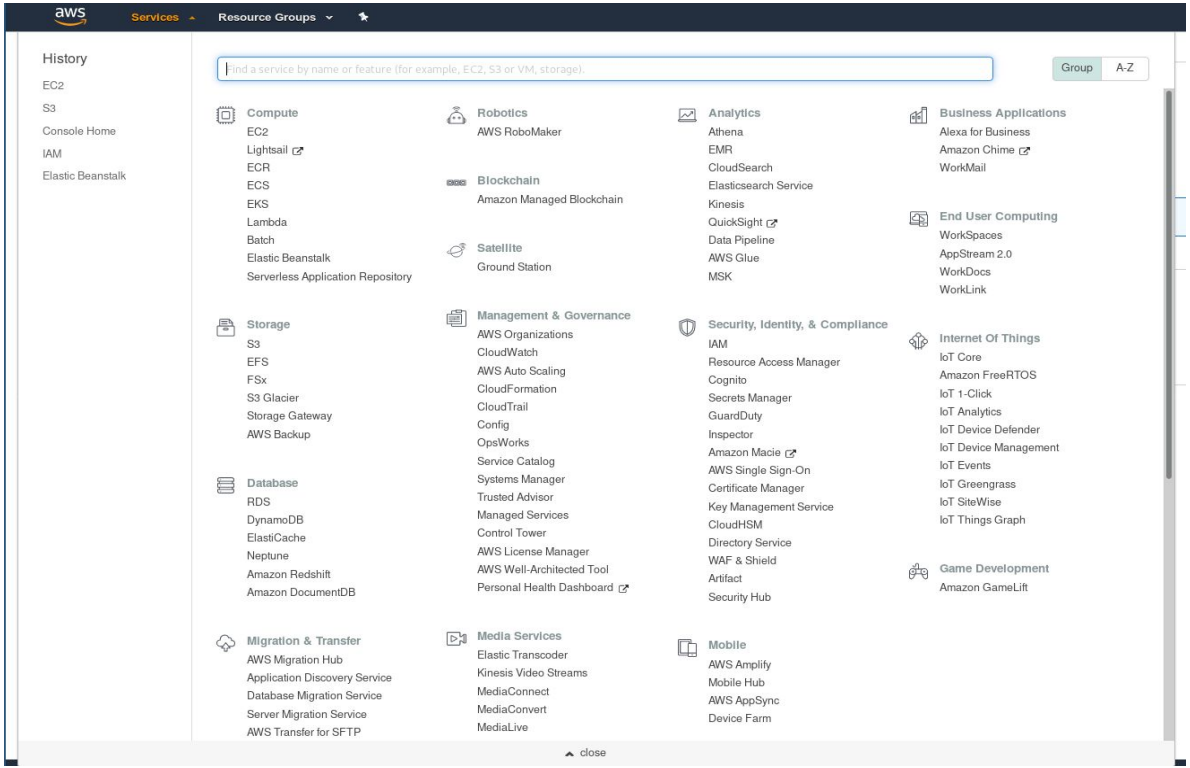
MiniKube Cluster

An Ozone cluster in Local Node.

We also ship standard docker files with official release, if you want to use them. These are part of official release and not depend upon Docker Hub.

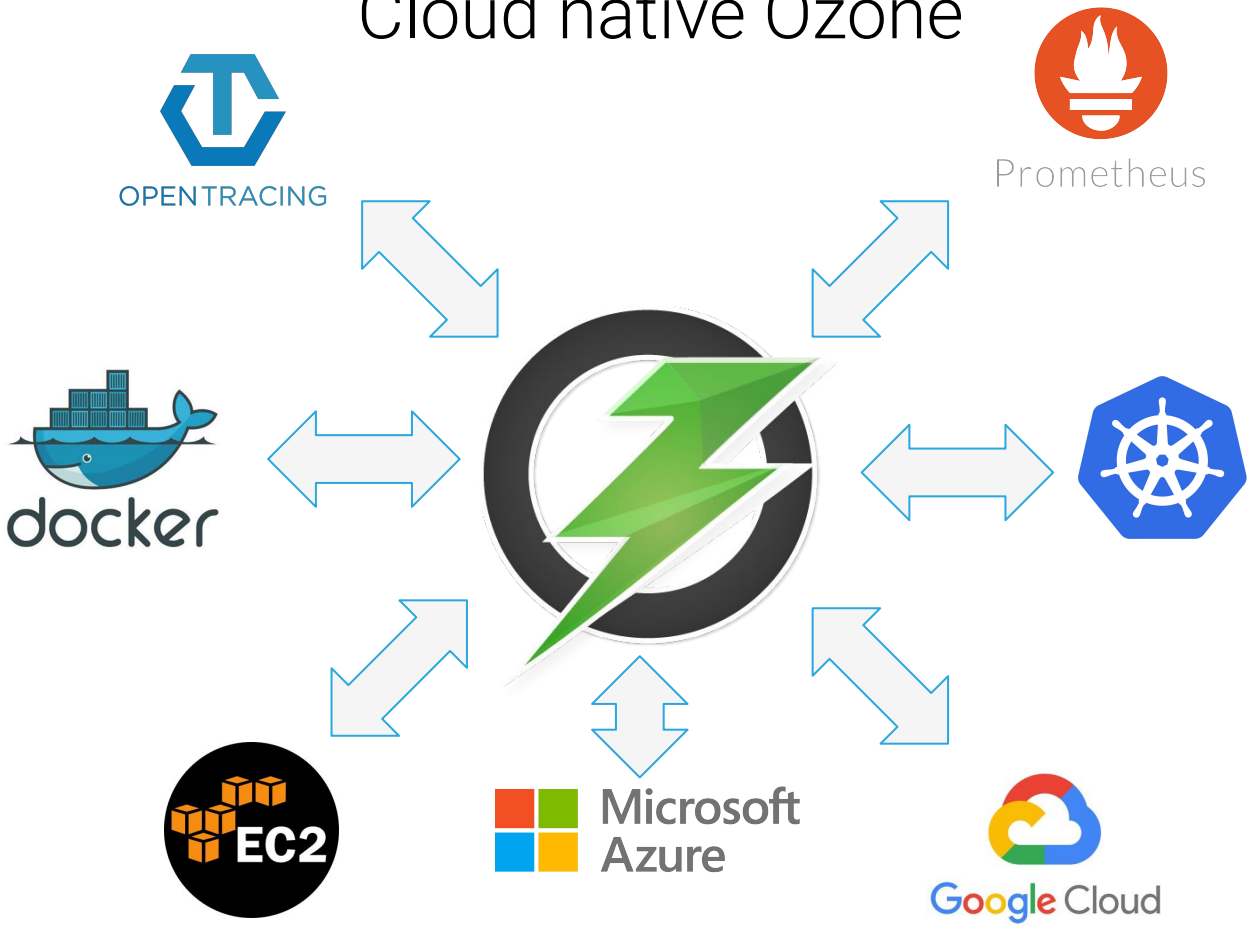
Multi-Container Cluster

Cloud-Native (def2)



cloudera

Cloud native Ozone



Cloud-native is a User Experience!

Hadoop ❤️ Containers

```
cd compose/ozonesecure
docker-compose up -d
```

Docker containers in Ozone development

- Easiest way to run Ozone pseudo cluster
  a. Different type of environment are supported (secure/non-secure/hadoop)

- Documentation := Text + Examples

- Can be used without additional build time overhead
  a. Yes: *docker run -v `pwd`:/opt/ozone ozone-runner datanode*
  b. Not: *docker build ….*

Testing ❤️ Containers

What is robot framework?

"**Robot Framework** is a generic open source automation framework for **acceptance testing**"

Easy to use language:

- Execute commands / File system operations
- Make assertions
- Test hierarchy / structure
- Extendable (in python)

+ Report generation

cloudera

smoketest/ozonefs/ozonefs.robot

```
*** Test Cases ***
Create volume and bucket
    Execute             ozone sh volume create http://om/fstest --user bilbo --quota 100TB --root
    Execute             ozone sh volume create http://om/fstest2 --user bilbo --quota 100TB --root
    Execute             ozone sh bucket create http://om/fstest/bucket1
    Execute             ozone sh bucket create http://om/fstest/bucket2
    Execute             ozone sh bucket create http://om/fstest2/bucket3

Run ozoneFS tests
                        Execute             ozone fs -mkdir -p o3fs://bucket1.fstest/testdir/deep
    ${result} =         Execute             ozone sh key list o3://om/fstest/bucket1 | jq -r '.[].keyName'
                        Should contain      ${result}         testdir/deep
                        Execute             ozone fs -copyFromLocal NOTICE.txt o3fs://bucket1.fstest/testdir/deep/
    ${result} =         Execute             ozone sh key list o3://om/fstest/bucket1  | jq -r '.[].keyName'
                        Should contain      ${result}         NOTICE.txt
```

```
*** Keywords ***
Execute
    [arguments]                     ${command}
    ${rc}                           ${output} =             Run And Return Rc And Output        ${command}
    Log                             ${output}
    Should Be Equal As Integers     ${rc}                   0
    [return]                        ${output}
```

```
Create volume and bucket
    Execute             ozone sh volume create http://om/fstest --user bilbo --quota 100TB --root
    Execute             ozone sh volume create http://om/fstest2 --user bilbo --quota 100TB --root
    Execute             ozone sh bucket create http://om/fstest/bucket1
    Execute             ozone sh bucket create http://om/fstest/bucket2
    Execute             ozone sh bucket create http://om/fstest2/bucket3

Run ozoneFS tests
                        Execute             ozone fs -mkdir -p o3fs://bucket1.fstest/testdir/deep
    ${result} =         Execute             ozone sh key list o3://om/fstest/bucket1 | jq -r '.[].keyName'
                        Should contain      ${result}           testdir/deep
                        Execute             ozone fs -copyFromLocal NOTICE.txt o3fs://bucket1.fstest/testdir/deep/
    ${result} =         Execute             ozone sh key list o3://om/fstest/bucket1  | jq -r '.[].keyName'
                        Should contain      ${result}           NOTICE.txt
```

smoketests Log

Generated
20190531 08:03:29 UTC+02:00
21 days 6 hours ago

Test Statistics

Total Statistics	Total	Pass	Fail	Elapsed	Pass / Fail
Critical Tests	93	43	50	00:09:43	
All Tests	93	43	50	00:09:43	

Statistics by Tag	Total	Pass	Fail	Elapsed	Pass / Fail
No Tags					

Statistics by Suite	Total	Pass	Fail	Elapsed	Pass / Fail
smoketests	93	43	50	00:10:27	
smoketests.auditparser	2	1	1	00:00:07	
smoketests.auditparser.Auditparser	2	1	1	00:00:07	
smoketests.basic	2	2	0	00:00:04	
smoketests.ozonefs	3	2	1	00:00:11	
smoketests.basic	2	2	0	00:00:07	
smoketests.basic	2	2	0	00:00:06	
smoketests.s3	34	12	22	00:04:08	
smoketests.s3.Awss3	1	0	1	00:00:46	
smoketests.s3.Bucketcreate	1	1	0	00:00:03	
smoketests.s3.Buckethead	1	1	0	00:00:03	
smoketests.s3.Bucketlist	1	1	0	00:00:04	
smoketests.s3.MultipartUpload	9	3	6	00:01:19	
smoketests.s3.Objectcopy	4	3	1	00:00:21	
smoketests.s3.Objectdelete	5	2	3	00:00:41	
smoketests.s3.Objectmultidelete	1	0	1	00:00:12	
smoketests.s3.Objectputget	10	0	10	00:00:35	
smoketests.s3.Webui	1	1	0	00:00:05	
smoketests.basic	5	2	3	00:00:49	
smoketests.basic.Basic	2	2	0	00:00:07	
smoketests.basic.Ozone-Shell	3	0	3	00:00:42	
smoketests.kinit	1	1	0	00:00:00	

SUITE **basic**

SUITE **ozonefs**

Full Name: smoketests.ozonefs
Documentation: Ozonefs test
Source: /opt/hadoop/smoketest/ozonefs/ozonefs.robot
Start / End / Elapsed: 20190531 07:56:50.907 / 20190531 07:57:01.770 / 00:00:10.863
Status: 3 critical test, 2 passed, 1 failed
3 test total, 2 passed, 1 failed

TEST **Create volume and bucket**

TEST **Check volume from ozonefs**

TEST **Run ozoneFS tests**

Full Name: smoketests.ozonefs.Run ozoneFS tests
Start / End / Elapsed: 20190531 07:56:58.087 / 20190531 07:57:01.769 / 00:00:03.682
Status: FAIL (critical)
Message: 1 != 0

KEYWORD commonlib. **Execute** ozone fs -mkdir -p o3fs://bucket1.fstest/testdir/deep

KEYWORD ${result} = commonlib. **Execute** ozone sh key list o3://om/fstest/bucket1 | grep -v WARN | jq -r '.[].keyName'

KEYWORD BuiltIn. **Should Contain** ${result}, testdir/deep

KEYWORD commonlib. **Execute** ozone fs -copyFromLocal NOTICE.txt o3fs://bucket1.fstest/testdir/deep/

Start / End / Elapsed: 20190531 07:57:00.293 / 20190531 07:57:01.769 / 00:00:01.476

KEYWORD ${rc}, ${output} = OperatingSystem. **Run And Return Rc And Output** ${command}

KEYWORD BuiltIn. **Log** ${output}

Documentation: Logs the given message with the given level.
Start / End / Elapsed: 20190531 07:57:01.768 / 20190531 07:57:01.768 / 00:00:00.000

07:57:01.768 INFO
```
2019-05-31 05:57:01 ERROR BlockOutputStreamEntryPool:299 - Try to allocate more blocks for write failed, already allocated 0 blocks for this write.
INTERNAL_ERROR org.apache.hadoop.ozone.om.exceptions.OMException: Allocate block failed.
        at org.apache.hadoop.ozone.om.protocolPB.OzoneManagerProtocolClientSideTranslatorPB.handleError(OzoneManagerProtocolClientSideTranslatorPB.java:715)
        at org.apache.hadoop.ozone.om.protocolPB.OzoneManagerProtocolClientSideTranslatorPB.allocateBlock(OzoneManagerProtocolClientSideTranslatorPB.java:738)
        at org.apache.hadoop.ozone.client.io.BlockOutputStreamEntryPool.allocateNewBlock(BlockOutputStreamEntryPool.java:248)
        at org.apache.hadoop.ozone.client.io.BlockOutputStreamEntryPool.allocateBlockIfNeeded(BlockOutputStreamEntryPool.java:296)
        at org.apache.hadoop.ozone.client.io.KeyOutputStream.handleWrite(KeyOutputStream.java:201)
        at org.apache.hadoop.ozone.client.io.KeyOutputStream.write(KeyOutputStream.java:193)
        at org.apache.hadoop.fs.ozone.OzoneFSOutputStream.write(OzoneFSOutputStream.java:46)
        at org.apache.hadoop.fs.FSDataOutputStream$PositionCache.write(FSDataOutputStream.java:57)
        at java.io.DataOutputStream.write(DataOutputStream.java:107)
        at org.apache.hadoop.io.IOUtils.copyBytes(IOUtils.java:96)
        at org.apache.hadoop.io.IOUtils.copyBytes(IOUtils.java:68)
        at org.apache.hadoop.io.IOUtils.copyBytes(IOUtils.java:129)
        at org.apache.hadoop.fs.shell.CommandWithDestination$TargetFileSystem.writeStreamToFile(CommandWithDestination.java:485)
        at org.apache.hadoop.fs.shell.CommandWithDestination.copyStreamToTarget(CommandWithDestination.java:407)
        at org.apache.hadoop.fs.shell.CommandWithDestination.copyFileToTarget(CommandWithDestination.java:342)
        at org.apache.hadoop.fs.shell.CopyCommands$CopyFromLocal.copyFile(CopyCommands.java:357)
        at org.apache.hadoop.fs.shell.CopyCommands$CopyFromLocal.copyFileToTarget(CopyCommands.java:365)
        at org.apache.hadoop.fs.shell.CommandWithDestination.processPath(CommandWithDestination.java:277)
        at org.apache.hadoop.fs.shell.CommandWithDestination.processPath(CommandWithDestination.java:262)
        at org.apache.hadoop.fs.shell.Command.processPathInternal(Command.java:367)
        at org.apache.hadoop.fs.shell.Command.processPaths(Command.java:331)
        at org.apache.hadoop.fs.shell.Command.processPathArgument(Command.java:304)
        at org.apache.hadoop.fs.shell.CommandWithDestination.processPathArgument(CommandWithDestination.java:257)
        at org.apache.hadoop.fs.shell.Command.processArgument(Command.java:286)
        at org.apache.hadoop.fs.shell.Command.processArguments(Command.java:270)
        at org.apache.hadoop.fs.shell.CommandWithDestination.processArguments(CommandWithDestination.java:228)
        at org.apache.hadoop.fs.shell.CopyCommands$Put.processArguments(CopyCommands.java:295)
        at org.apache.hadoop.fs.shell.CopyCommands$CopyFromLocal.processArguments(CopyCommands.java:385)
        at org.apache.hadoop.fs.shell.FsCommand.processRawArguments(FsCommand.java:120)
        at org.apache.hadoop.fs.shell.Command.run(Command.java:177)
        at org.apache.hadoop.fs.FsShell.run(FsShell.java:327)
        at org.apache.hadoop.util.ToolRunner.run(ToolRunner.java:76)
        at org.apache.hadoop.util.ToolRunner.run(ToolRunner.java:90)
        at org.apache.hadoop.fs.FsShell.main(FsShell.java:390)
copyFromLocal: Allocate block failed.
```

KEYWORD BuiltIn. **Should Be Equal As Integers** ${rc}, 0

Documentation: Fails if objects are unequal after converting them to integers.
Start / End / Elapsed: 20190531 07:57:01.768 / 20190531 07:57:01.769 / 00:00:00.001

07:57:01.769 INFO
```
Argument types are:
<type 'int'>
<type 'unicode'>
```

07:57:01.769 FAIL
```
1 != 0
```

Robot tests, ftw!

Tests are part of the distribution tar file

Can be executed in ANY environment

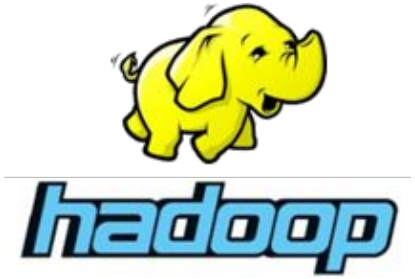
Part of the nightly / PR jenkins jobs

Can be used during the vote (!!!)

Hadoop ❤️ Kubernetes

Hadoop ? Kubernetes

Hadoop **VS** Kubernetes

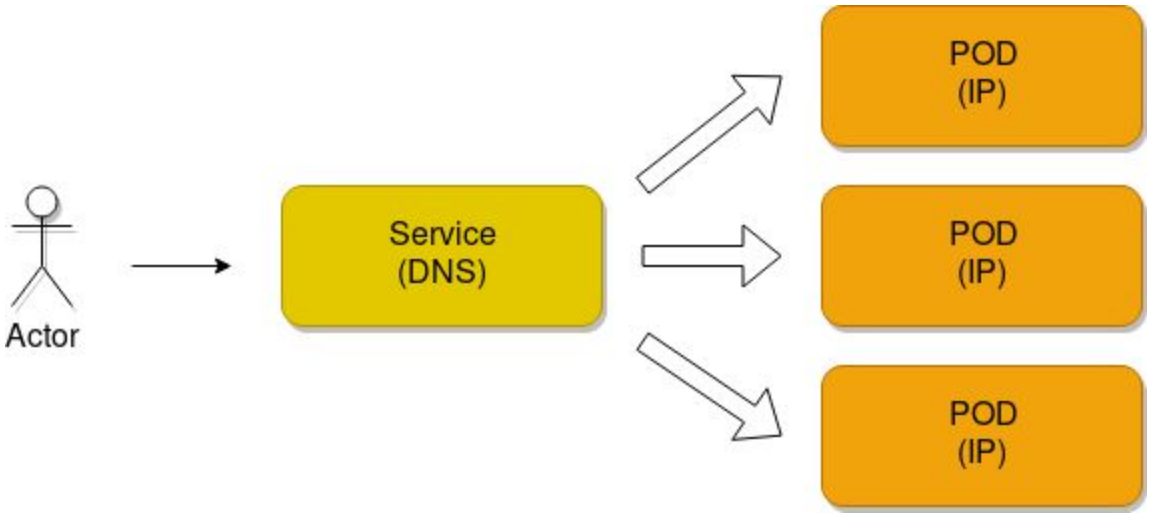


VS

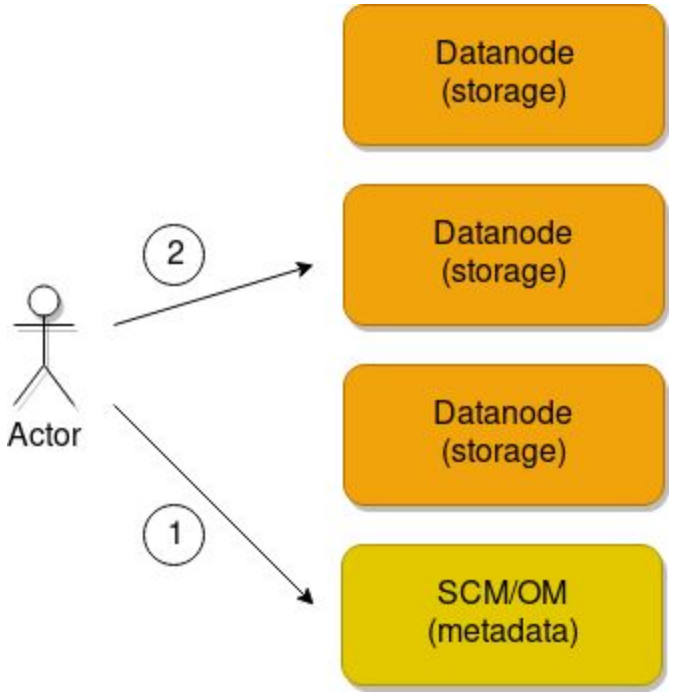
Hadoop ❤️ Kubernetes

Does Ozone run well in k8s?

Kubernetes for Stateless apps



Architecture of a storage app

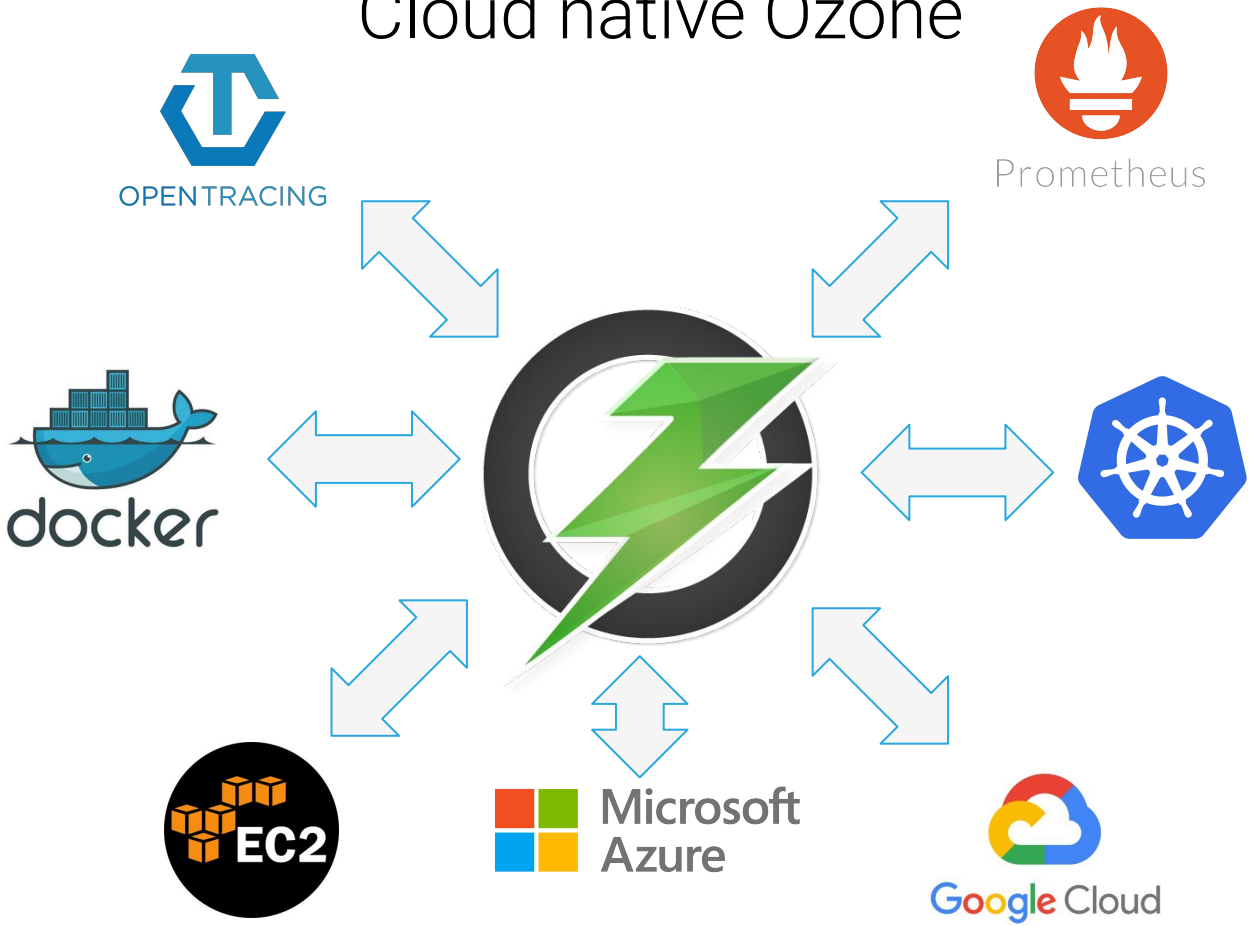


cloudera

Does Ozone run well in k8s?

Yes*

Cloud native Ozone



Visibility

Hadoop:

- Hadoop metrics
  - Custom implementation
  - Supports multiple sink implementation

Cloud-native

- Metrics
  - Prometheus
  - Visualization with other tools (eg. Grafana)

Visibility

Hadoop:

- Hadoop metrics
  - Custom implementation
  - Supports multiple sink implementation
- *HTrace (deprecated)*

Cloud-native

- Metrics
  - Prometheus
  - Visualization with other tools (eg. Grafana)
- Tracing
  - OpenTracing
  - OpenCensus

Visibility

Hadoop:

- Hadoop metrics
  - Custom implementation
  - Supports multiple sink implementation
- *HTrace (deprecated)*
- *Log4j (no collections)*

Cloud-native

- Metrics
  - Prometheus
  - Visualization with other tools (eg. Grafana)
- Tracing
  - OpenTracing
  - OpenCensus
- Log collection
  - Fluentd, Loki

Results

Hadoop:

- **Hadoop metrics**
  - Custom implementation
  - Supports multiple sink implementation
  - **Prometheus endpoint**
- *HTrace (deprecated)*
- **Log4j (no collections)**

Cloud-native

- Metrics
  - **Prometheus**
  - Visualization with other tools (***embedded Grafana***)
- Tracing
  - **OpenTracing**
  - OpenCensus
- ***Log collection***
  - Fluentd, Loki

Hadoop ❤️ Kubernetes

Kubernetes ❤️ Hadoop

aws S3 protocol

Hadoop FS

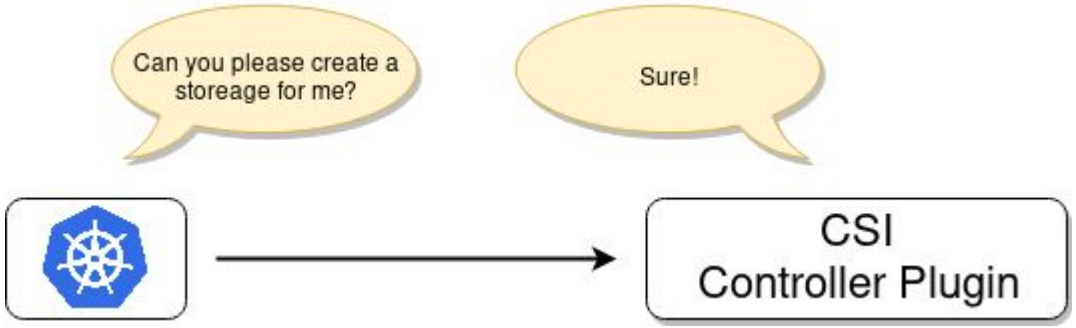
CSI

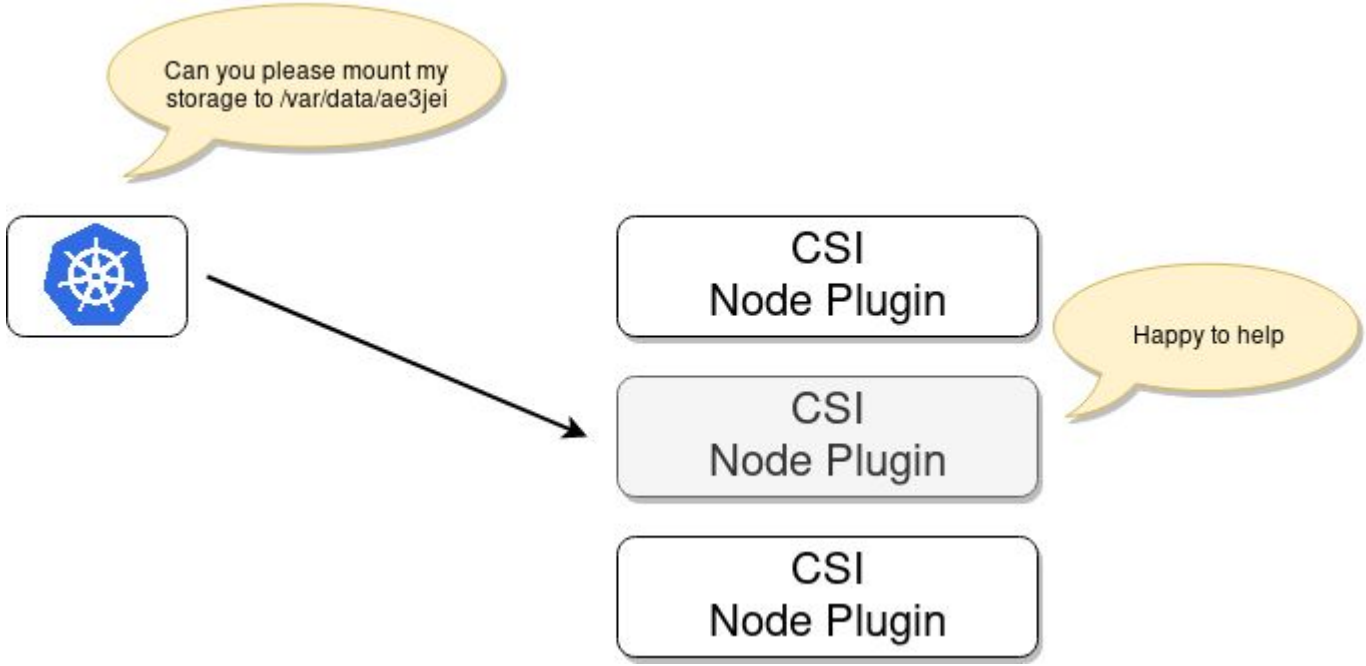
Apache Hadoop Ozone

THE APACHE SOFTWARE FOUNDATION

hadoop.apache.org/ozone

Container Storage Interface:
Vendor-neutral interface for volume management





cloudera

Demo

Results

CSI server is included (easy part)

- Create/delete volumes/buckets
- Mount/umount the volume as a real filesystem (on the right node)

Hard part is to mount the file system (data path):

- Present: mounting via S3 Fuse drivers
  - Multiple Implementation, Multiple caching strategy
- Future/WIP: Native Fuse adapter to mount Ozone buckets/Hadoop fs

Summary/Roadmap

What we have

- Docker based pseudo cluster definitions
- “Robot tests” (acceptance tests)
- Kubernetes deployment files
- Integration with cloud-native projects (prometheus, grafana, loki, opentracing)
- CSI server implementation

Not the end of the Journey:

- Support in Rook operator
- Automatic testing in K8s (acceptance tests, performance test, chaos tests)
- Service/configuration discovery (on-prem + Kubernetes)

Q&A

aws S3 protocol

Hadoop FS

CSI

Apache Hadoop Ozone

THE APACHE SOFTWARE FOUNDATION hadoop.apache.org/ozone

Márton Elek

elek@apache.org // @anzix

helm + kustomize =

https://github.com/elek/flekszible

cloudera

