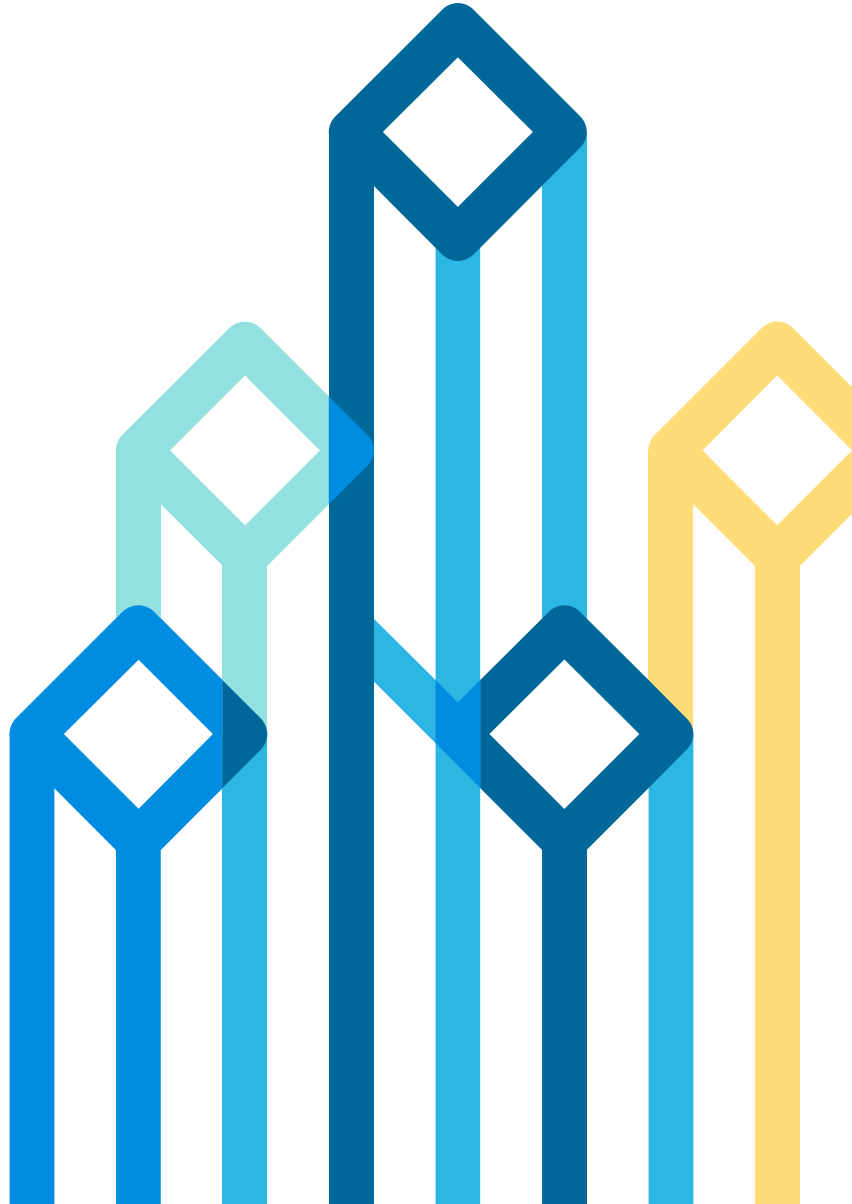
cloudera®

Building A Better Test Platform:

A Case Study of Improving Apache HBase Testing with Docker

Aleks Shulman, Dima Spivak



Outline

- About Cloudera
- Apache HBase
  - Overview
  - API compatibility
- API compatibility testing framework
  - Implementation and challenges
- Intro. to containers and Docker
- API compatibility testing framework, revisited
- Conclusions

About Cloudera

- Distributes CDH, Cloudera's 100% open-source distribution including Apache Hadoop.
- Distributes Cloudera Manager (CM), a proprietary monitoring and management layer atop CDH.
- Contributes heavily to the open source community.
  - Employs 50 committers, holding 84 committerships.
  - 18 Apache projects started by Clouderans, including Flume, Sqoop, Sentry, etc.

What is Apache HBase?

- “Apache HBase is the Hadoop database, a distributed, scalable, big data store.”
- Built atop HDFS.
- Low-latency, consistent, random-access.
- Modeled after Google’s BigTable paper.
  - Non-relational.
  - NoSQL.

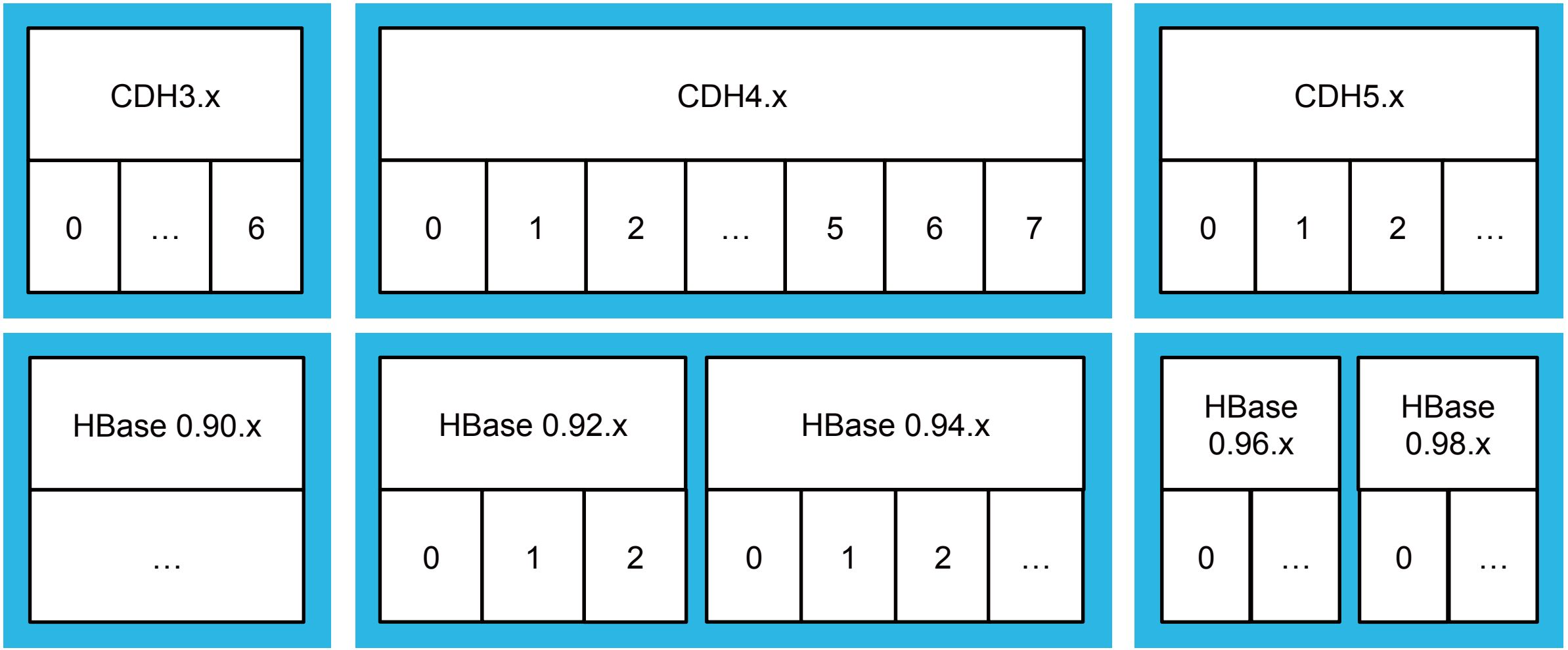
SQL vs. HBase

SQL	HBase API
CREATE	create
ALTER	modify*
DROP	disable, drop
INSERT	put
SELECT	get, scan
UPDATE	put
DELETE	delete

HBase API	
Java	Full-featured.
REST	Easy to use.
Thrift	Multi-language support.

cloudera

CDH and Apache HBase



API Compatibility

- RPC compatibility:
  - Incompatibilities around serialization/deserialization.

```
17:22:15  Exception in thread "main" java.lang.IllegalArgumentException: Not a host:port pair: PBUF
17:22:15  *
17:22:15   api-compat-8.ent.cloudera.com(
17:22:15  at org.apache.hadoop.hbase.util.Addressing.parseHostname(Addressing.java:60)
17:22:15  at org.apache.hadoop.hbase.ServerName.<init>(ServerName.java:101)
17:22:15  at org.apache.hadoop.hbase.ServerName.parseVersionedServerName(ServerName.java:283)
17:22:15  at org.apache.hadoop.hbase.MasterAddressTracker.bytesToServerName(MasterAddressTracker.java:77)
17:22:15  at org.apache.hadoop.hbase.MasterAddressTracker.getMasterAddress(MasterAddressTracker.java:61)
17:22:15  at org.apache.hadoop.hbase.client.HConnectionManager$HConnectionImplementation.getMaster(HConnectionManager.java:703)
17:22:15  at org.apache.hadoop.hbase.client.HBaseAdmin.<init>(HBaseAdmin.java:126)
17:22:15  at Client_4_3_0.setup(Client_4_3_0.java:716)
17:22:15  at Client_4_3_0.main(Client_4_3_0.java:63)
```

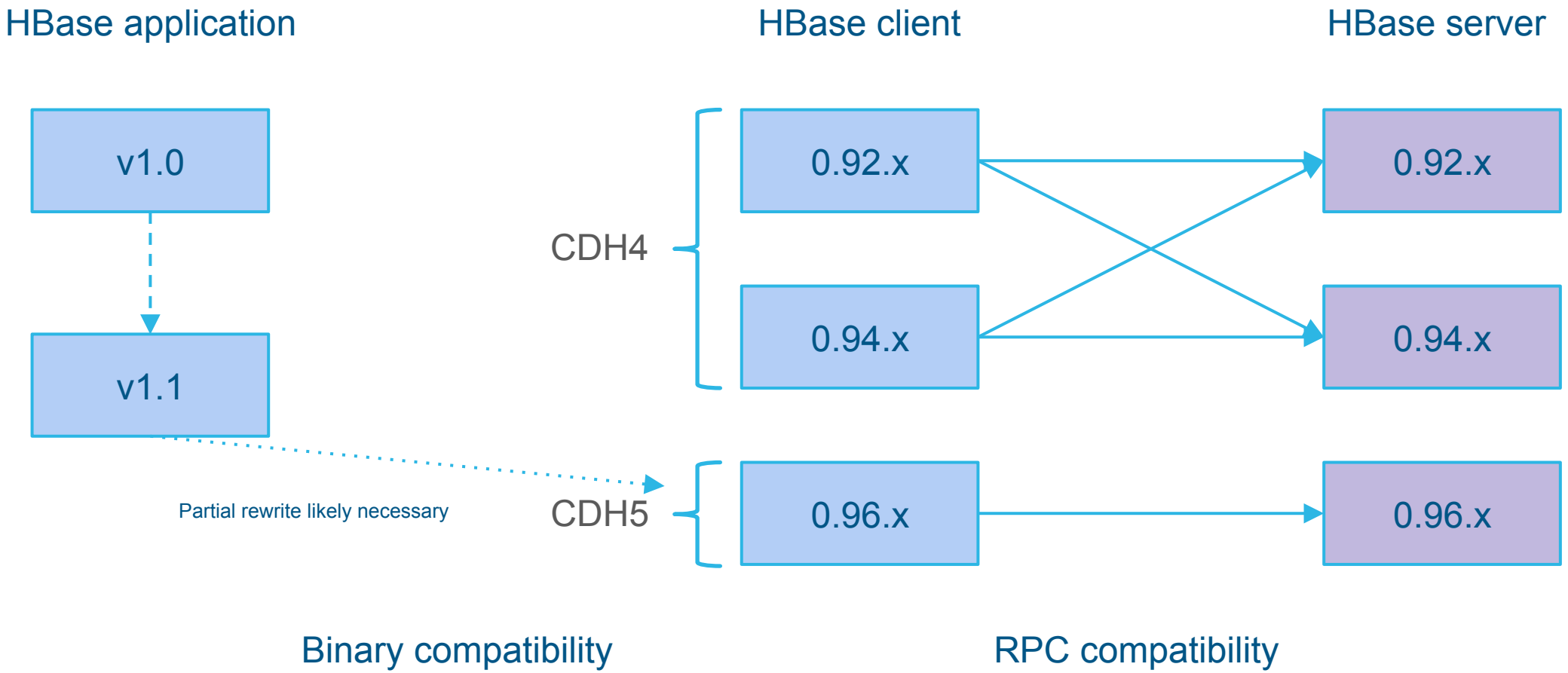
API Compatibility

- Binary compatibility:
  - Client dependencies changing in an incompatible way can result in a runtime exception.
  - Examples:
    - HTable constructors removed in CDH4.2.

```
public HTable(final String tableName)
public HTable(final byte[] tableName)
```

    - HBASE-8273: Change of HColumnDescriptor setter return types.
      - Previously returned void, builder pattern changed this.

API Compatibility



Testing

1. Build cluster with version X.
2. Run API client with version Y.

$$X \neq Y$$

Configuration Matrix		cdh4.0.0	cdh4.0.1	cdh4.1.0	cdh4.1.1	cdh4.1.2	cdh4.1.3	cdh4.1.4	cdh4.2.0	cdh4.2.1	cdh4.3.0	cdh4Nightly
rpc	cdh4.0.0	●	●	●	●	●	●	●	●	●	●	●
	cdh4.0.1	●	●	●	●	●	●	●	●	●	●	●
	cdh4.1.0	●	●	●	●	●	●	●	●	●	●	●
	cdh4.1.1	●	●	●	●	●	●	●	●	●	●	●
	cdh4.1.2	●	●	●	●	●	●	●	●	●	●	●
	cdh4.1.3	●	●	●	●	●	●	●	●	●	●	●
	cdh4.1.4	●	●	●	●	●	●	●	●	●	●	●
	cdh4.2.0	●	●	●	●	●	●	●	●	●	●	●
	cdh4.2.1	●	●	●	●	●	●	●	●	●	●	●
	cdh4.3.0	●	●	●	●	●	●	●	●	●	●	●
	cdh4Nightly	●	●	●	●	●	●	●	●	●	●	●
binary	cdh4.0.0	●	●	●	●	●	●	●	●	●	●	●
	cdh4.0.1	●	●	●	●	●	●	●	●	●	●	●
	cdh4.1.0	●	●	●	●	●	●	●	●	●	●	●
	cdh4.1.1	●	●	●	●	●	●	●	●	●	●	●
	cdh4.1.2	●	●	●	●	●	●	●	●	●	●	●
	cdh4.1.3	●	●	●	●	●	●	●	●	●	●	●
	cdh4.1.4	●	●	●	●	●	●	●	●	●	●	●
	cdh4.2.0	●	●	●	●	●	●	●	●	●	●	●
	cdh4.2.1	●	●	●	●	●	●	●	●	●	●	●
	cdh4.3.0	●	●	●	●	●	●	●	●	●	●	●
	cdh4Nightly	●	●	●	●	●	●	●	●	●	●	●

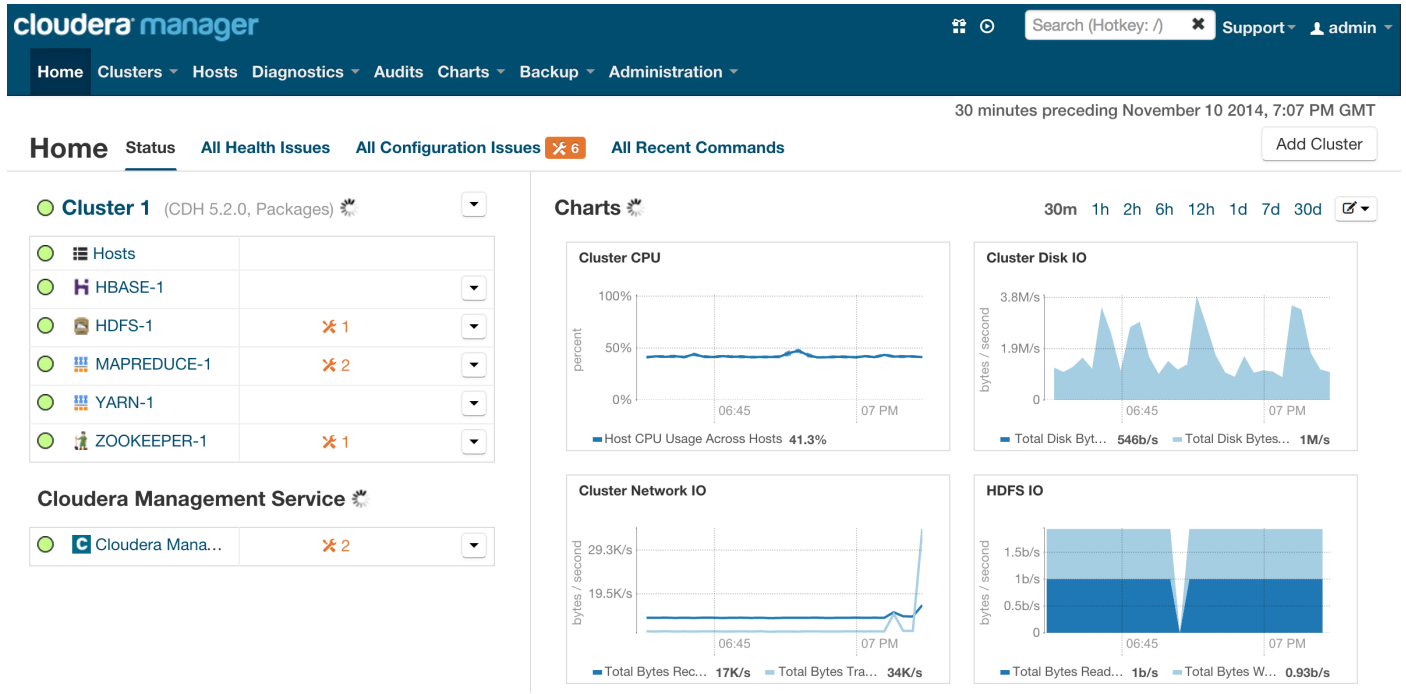
API testing implementation

- Provisioning
  - Provision host on private cloud.
  - Static VMs.
- Tarball framework
  - Compile and package CDH server bits (version X).
  - Start up the services as a single-node pseudo-distributed cluster (version X).
  - Compile and run CDH client (version Y).
- Problems
  - Failure modes
    - Infrastructure (VMs, hosts).
    - Artifacts.
  - Additional complexity
    - Special configurations (e.g. Kerberos).
    - Clean-up jobs for hosts.
    - Adding/maintaining server and client versions.

No buffer between provisioning, packaging, deployment, and running tests.

Implementation alternatives

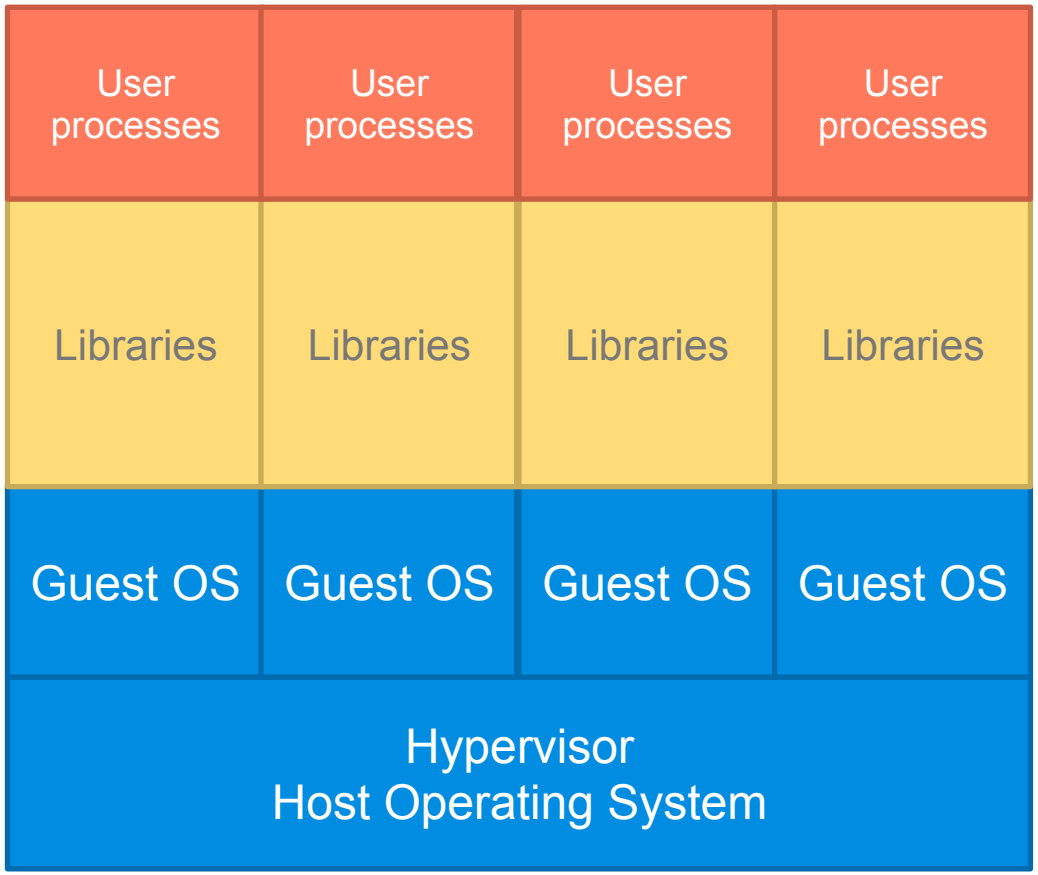
- Can we reuse existing tools?
  - CDH packaging
  - Cloudera Manager*
- Can we maximize utilization of computing resources?



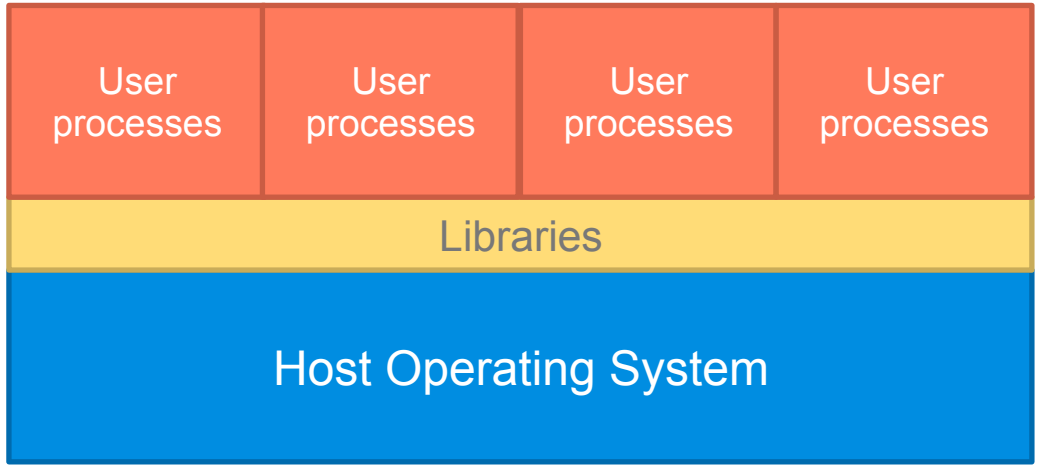
Linux Containers

- Isolated environments on a Linux host
  - cgroups (2.6.24+)
    - Isolating resources (memory, CPU, networking)
  - Namespace isolation (filesystems, process trees)

Linux Containers



Virtual Machines



Containers

Linux Containers

Advantages

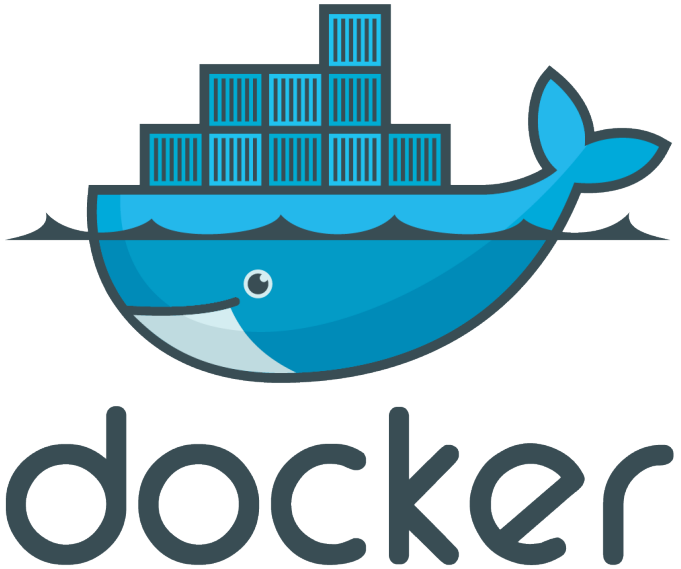
- Low overhead
- Fast startup and shutdown

Disadvantages

- Only host kernel-compatible operating systems
- Less isolation*

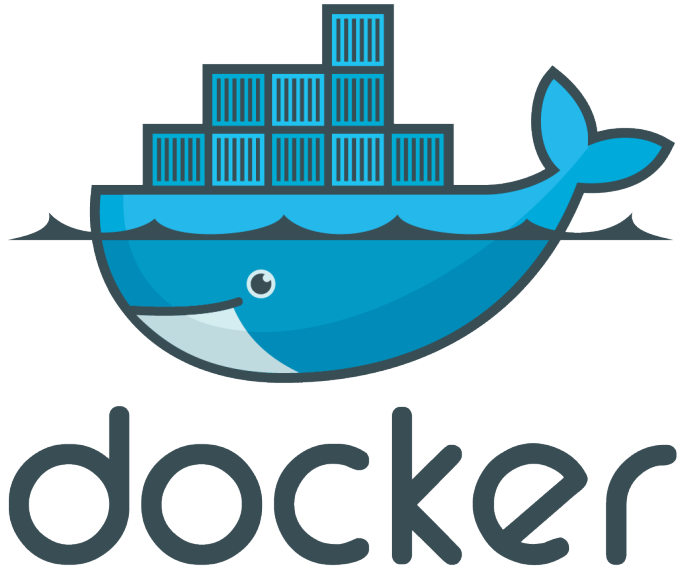
Docker

- User front-end for containers
  - Container management (start, stop, pause)
  - Images (templates for containers)
  - Registries (repository for images)



Docker

- `docker run`
  - Start container from image
- `docker build`
  - Create image from Dockerfile
- `docker commit`
  - Create image from container



Use cases

- Single-process applications

```
root@savard:~# docker run ubuntu:12.04 echo "Hello world"
Hello world
root@savard:~# _
```

- Persistent containers
  - Containers as daemons.
    - init process

```
root@savard:~# docker run -d ubuntu:12.04 /sbin/init
```

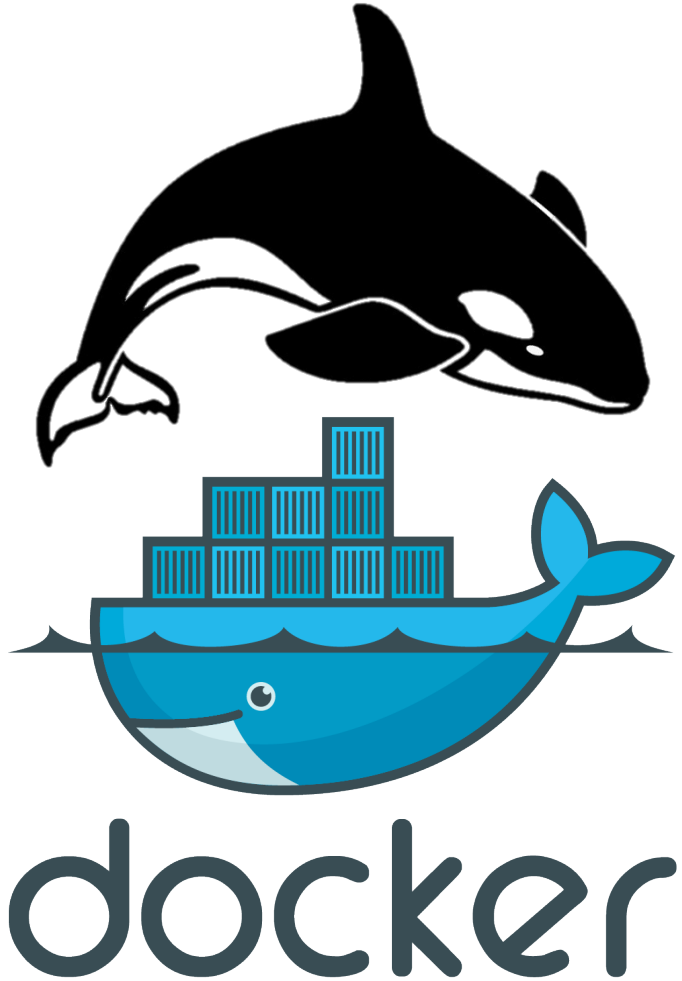
API testing implementation, revisited

- Problems
  - Failure modes
    - Infrastructure (VMs, hosts).
    - Artifacts.
  - Additional complexity
    - Special configurations (e.g. Kerberos).
    - Clean-up jobs for hosts.
    - Adding/maintaining server and client versions.
- Solutions
  - Portability
    - Images, internal registry.
    - Artifacts built in.
  - Reduce complexity
    - Package configurations in images.
    - Containers initialize to same state.
    - No need to maintain all-in-one image creation logic.

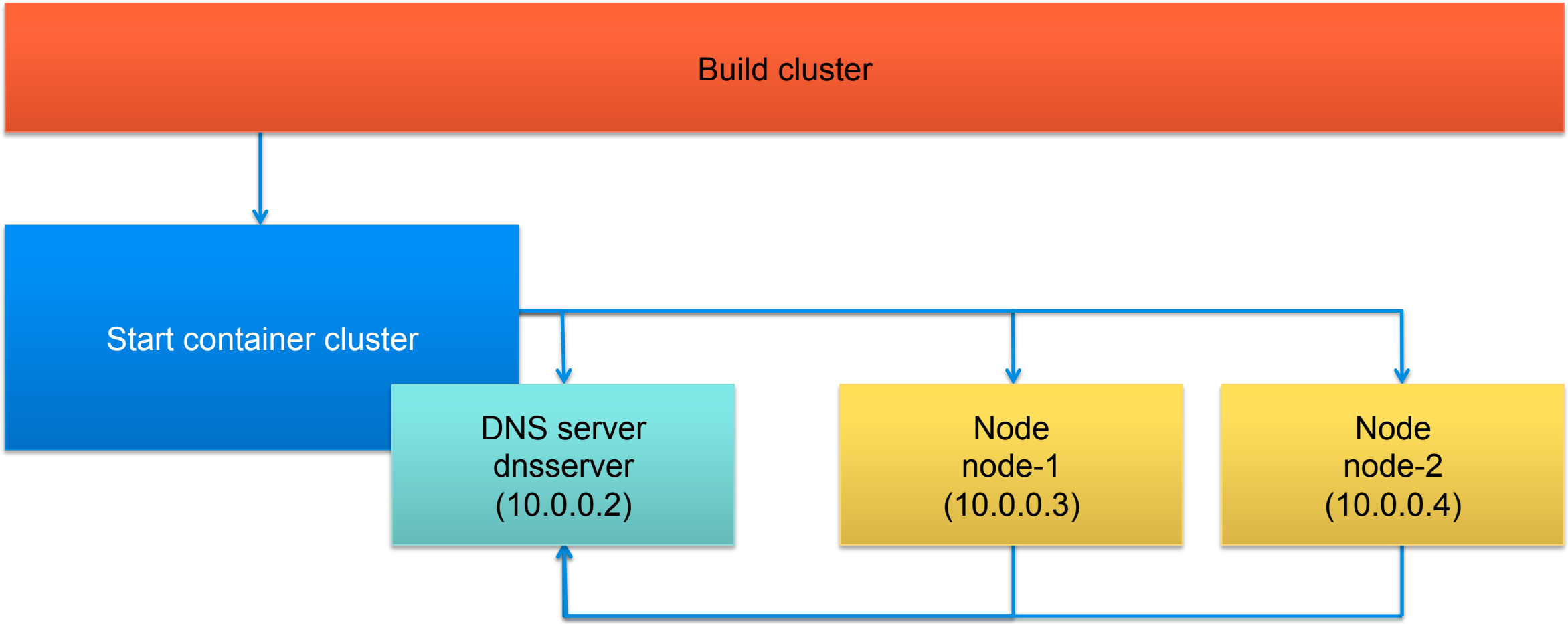
Separated packaging, cluster deployment, and running tests; environment becomes a test parameter.

API testing implementation, revisited

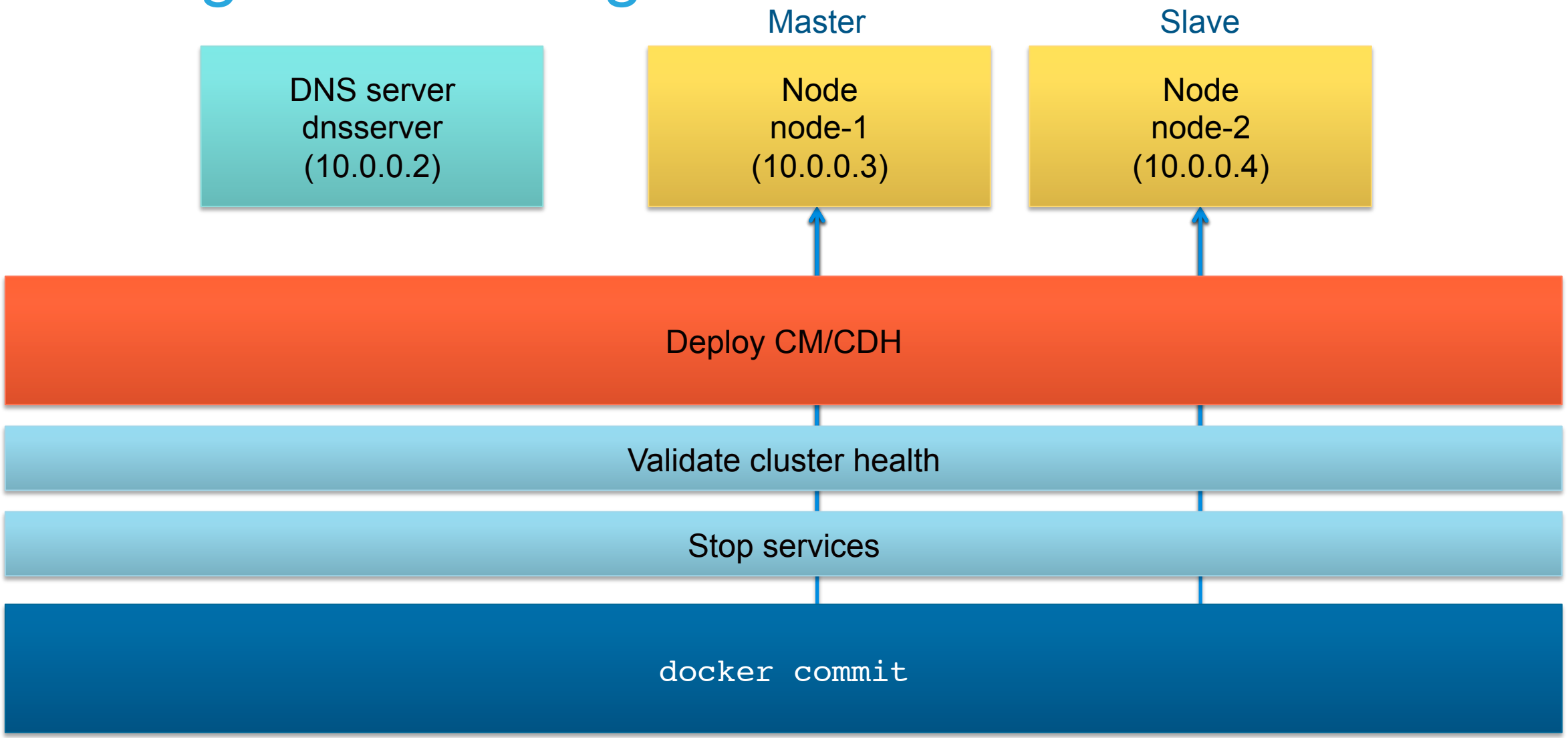
- HBase server
  - Persistent containers.
  - Run CM/CDH in a set of containers.
    - 1 container per CM host.
    - 1 container for DNS server.
  - Create any size cluster from 2 images.
- Provision physical host once, switch cluster versions easily.
- API client
  - Single-process application.



Building cluster images

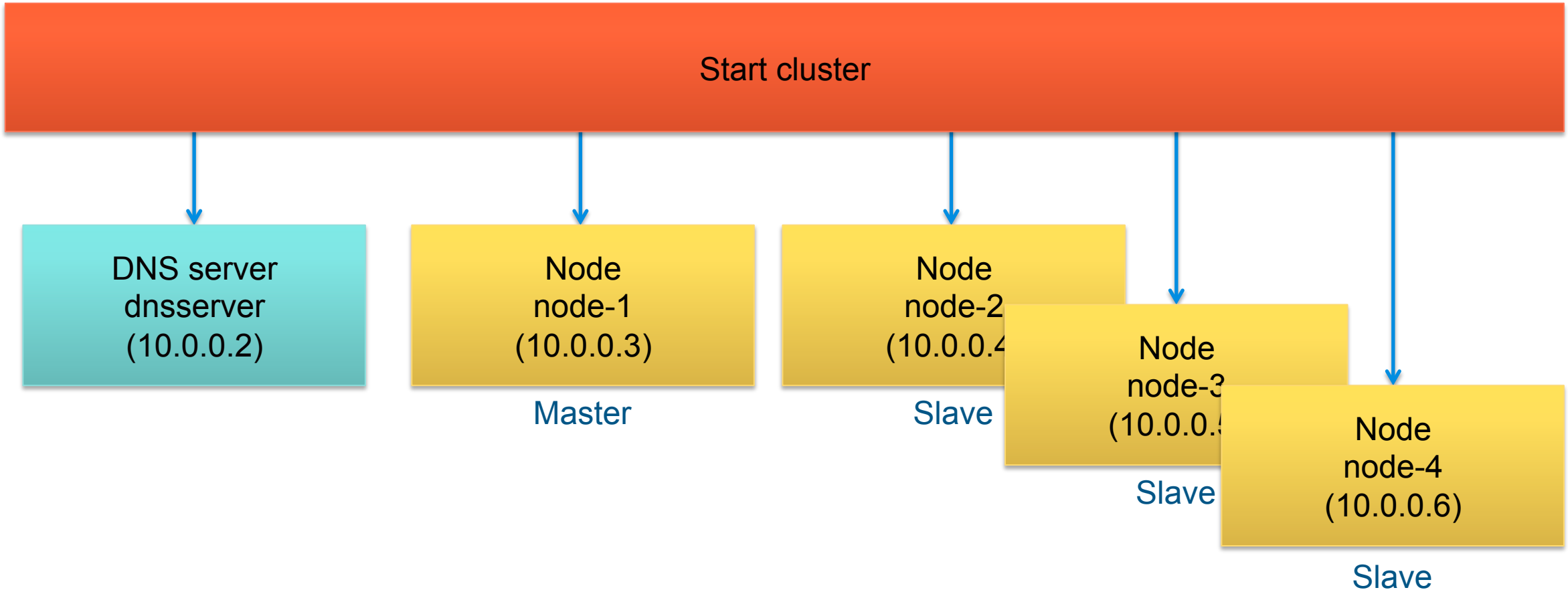


Building cluster images

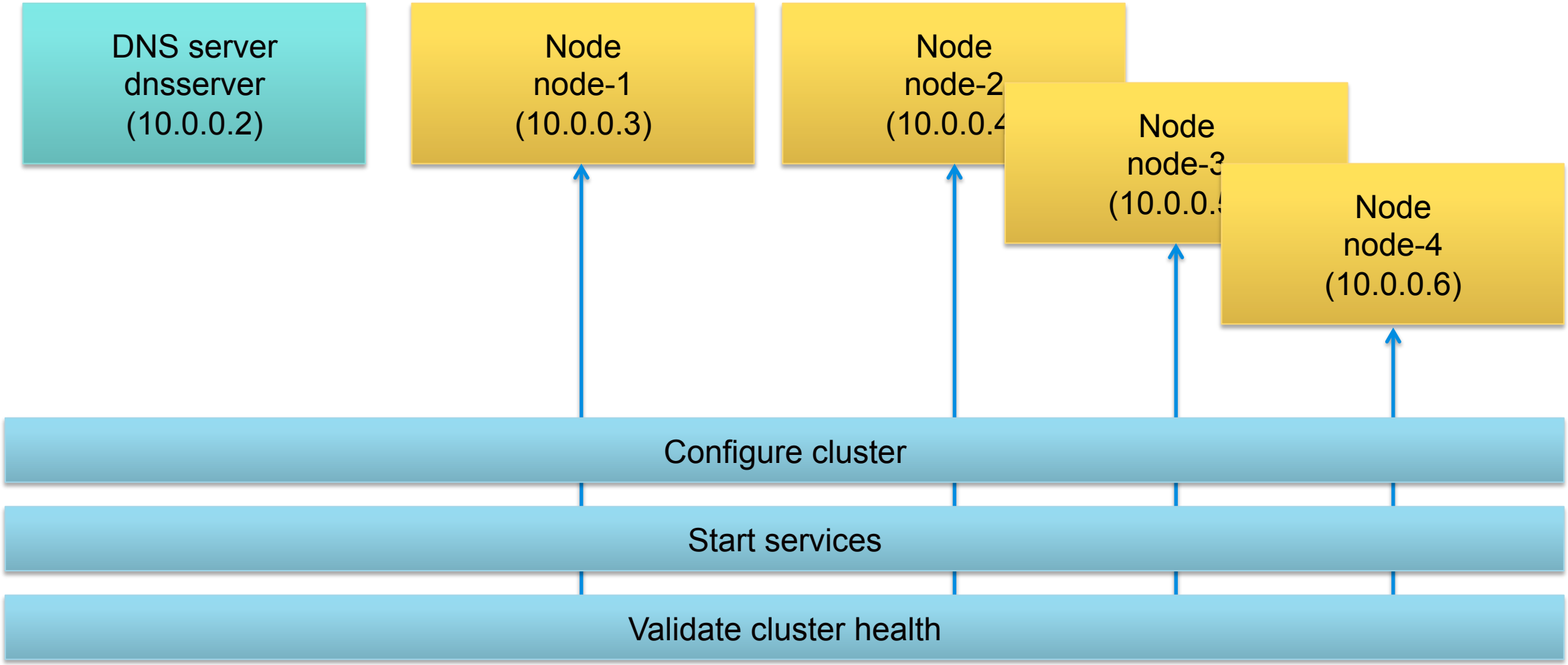


cloudera

Starting cluster containers

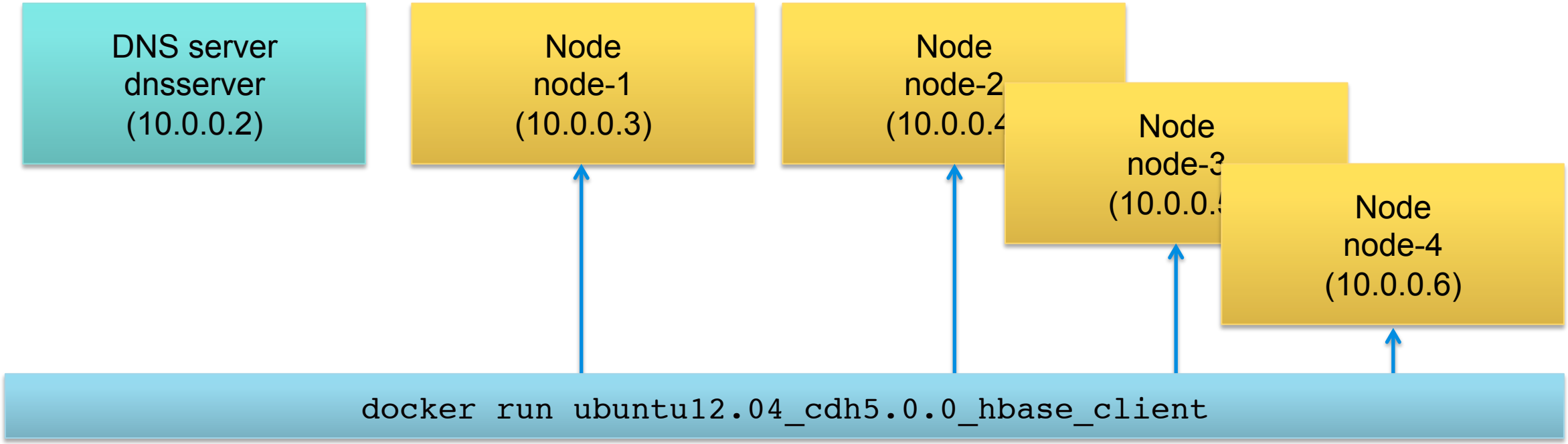


Starting cluster containers



cloudera

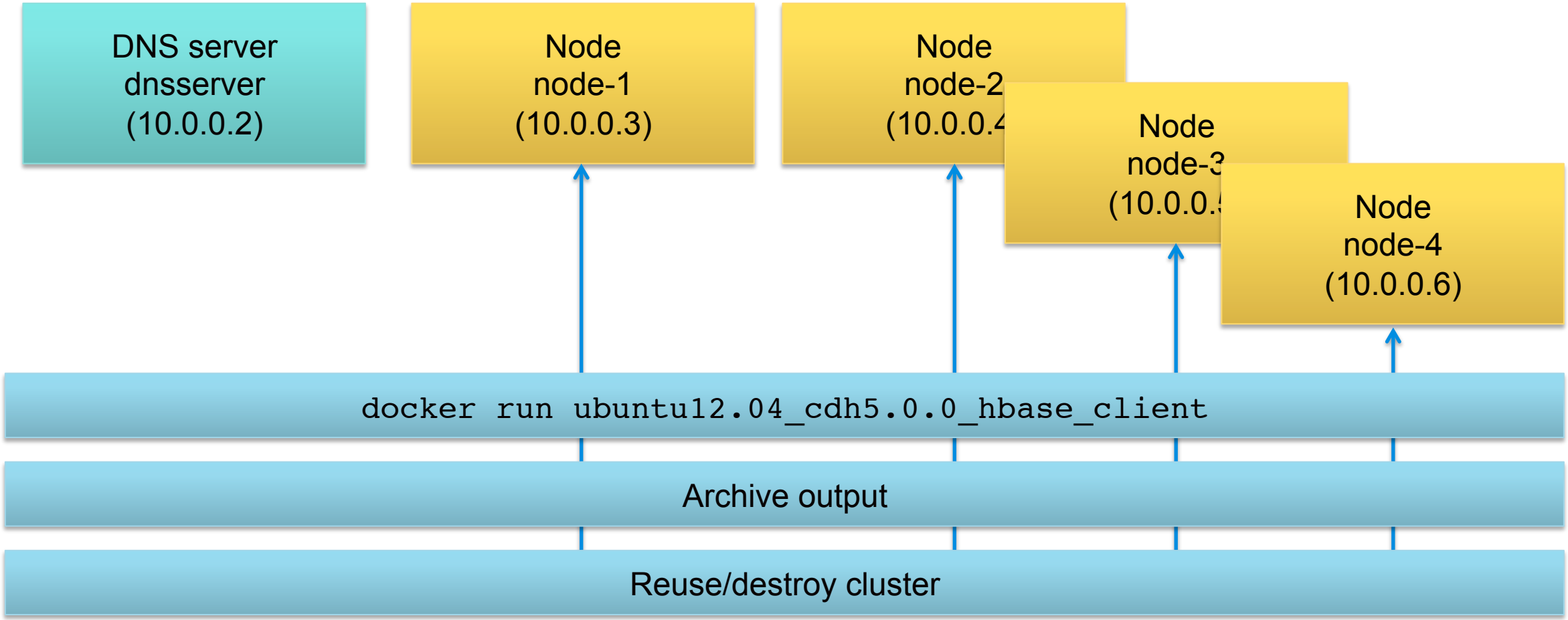
Running API client



- Script executing org.apache.hadoop.hbase.client.TestFromClientSide on distributed cluster.
- Pass in environmental variables:

```
-e "HBASE_ZK_QUORUM=node-1.internal"
-e "MODE=<rpc|binary>"
```

Running API client



Conclusions

- Recognized automation that was more trouble than it was worth.
  - More effort into setting up test environment than on running the tests.
- Overhauled test framework using Docker.
  - Maximize utilization of computing resources.
  - Made environment a test parameter.
- Further work:
  - Consider workflows beyond compatibility testing.
    - Upgrade testing.
    - Failure injection.

cloudera