

Apache HBase Deploys

Michael Stack
GOTO Amsterdam 2011

Me

- Chair of Apache HBase Project
- Committer since 2007
- Committer and member of Hadoop PMC
- Engineer at StumbleUpon in San Francisco



Overview

1. Quick HBase review
2. HBase deploys

HBasics

- An open source, distributed, scalable datastore
- Based on Google BigTable Paper[2006]

More HBasics



- Apache Top-Level Project: hbase.apache.org
 - SU, FB, Salesforce, Cloudera, TrendMicro, Huawei
- Built on Hadoop HDFS & Zookeeper
 - Stands on shoulders of giants!
- NOT an RDBMS
 - No SQL, No Joins, No Transactions, etc.
 - Only CRUD+S(can)... and Increment
- Not a drop-in replacement for...

Hadoop HDFS is a fault-tolerant, checksummed, scalable distributed file system



HBase is all about...



- Near linear as you add machines
- Size-based autosharding
- Project Goal: *“Billions of rows X millions of columns on clusters of ‘commodity’ hardware”*

HBase lumped with...

- Other BigTable 'clones'
 - Same datamodel: Hypertable, Accumulo
 - Similar: Cassandra
- NoSQL/NotSQL/NotJustSQL/etc
- Popular, 'competitive' mostly OSS space
 - Millions of \$\$\$\$!



HBase Data Model

- Tables of Rows x Column Families
- Columns are members of a Column Family
 - Column name has CF prefix: e.g *foo:bar*
- Columns created on the fly
- CF up-front as part of schema definition
 - Per CF TTL, versions, blooms, compaction

More Data Model

- Cells are *versioned*
 - Timestamp by default
- Strongly consistent view on row
 - increment, checkAndSet
- All are byte arrays; rows, columns, values
- All SORTED byte-lexicographically
 - Rows, columns in row, versions in column

Bigtable is...

“...a sparse, distributed, persistent, multi-dimensional sorted map. The map is indexed by a row key, column key, and a timestamp; each value in the map is an uninterpreted array of bytes”

Can think of HBase as this too....

Table =>

Row =>

Column =>

Version => Cell

Architecture

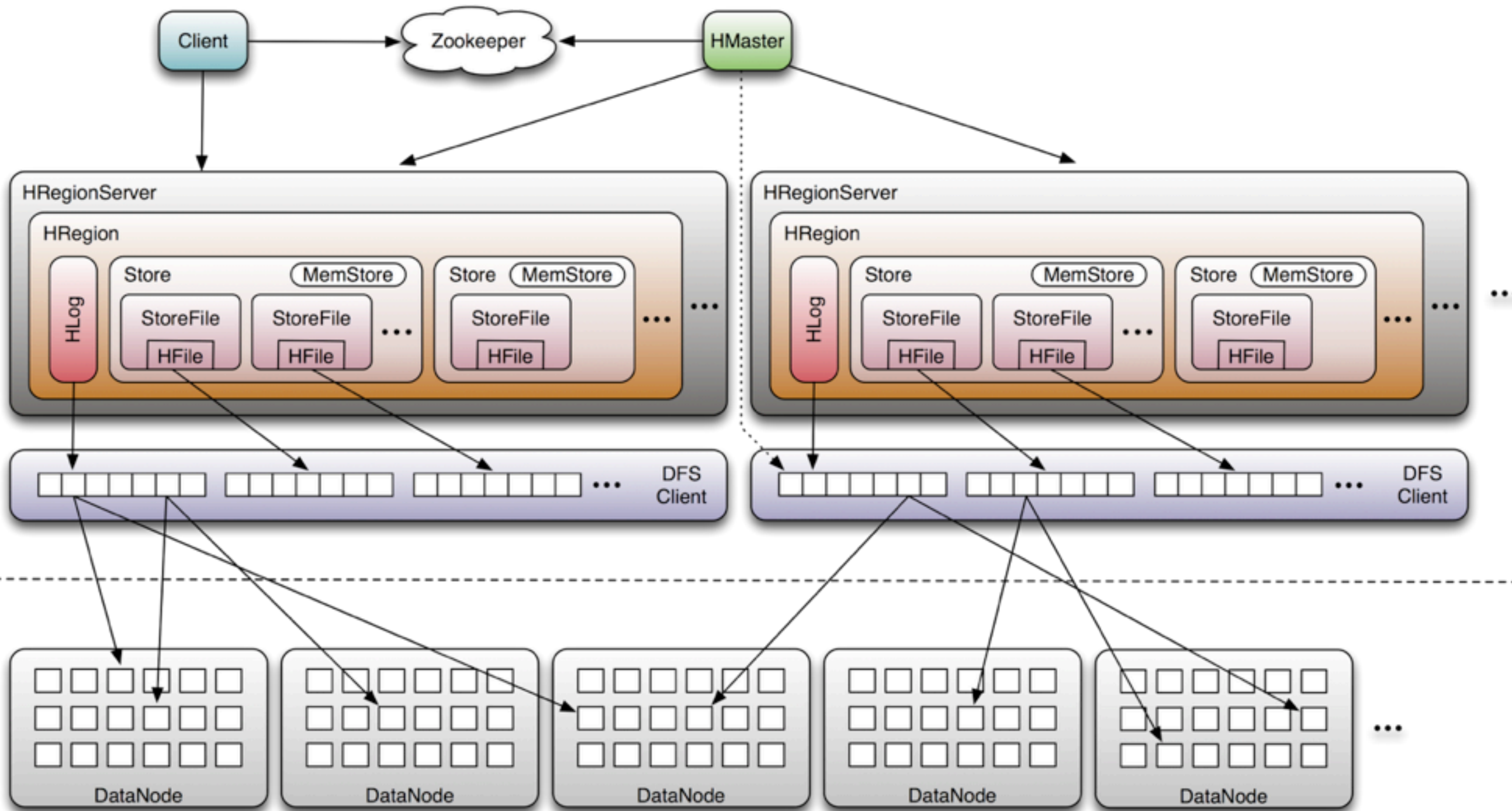
- Table dynamically split into tablets/“regions”
- Each region a contiguous piece of the table
 - Defined by [startKey, endKey)
- Region automatically splits as grows
- Regions are spread about the cluster
- Load Balancer
 - Balances regions across cluster

More Architecture

- HBase cluster is made of
 - Master(s)
 - Offline, janitorial, boot cluster, etc
 - Slave RegionServers
 - Workers, carry Regions, Read Writes
- ...all running on top of an HDFS cluster
- ...making use of a ZooKeeper ensemble

HBase

Hadoop



Out-of-the-box

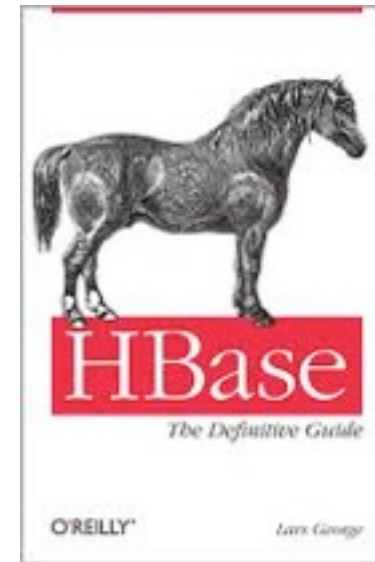
- Java API but also thrift & REST clients
- UI, Shell
- Good Hadoop MapReduce connectivity
 - PIG, Hive & Cascading source and sink
- Metrics via Hadoop metrics subsystem
- Server-side filters/Coprocessors
- Hadoop Security
- Replication
- etc

When to use it

- Large amounts of data
 - 100s of GBs up to Petabytes
- Efficient random access inside large datasets
 - How we compliment the Hadoop stack
- Need to scale gracefully
 - Scale writes, scale cache
- Do NOT need full RDBMS capabilities

For more on HBase

- Lars' George's book
- hbase.org/book.html



APACHE
HBASE

Copyright © 2011 Apache Software Foundation

Revision History

Revision 0.93-SNAPSHOT	
Adding first cuts at Configuration, Getting Started, Data Model	
Revision 0.89.20100924	5 October 2010
Initial layout	

Abstract

This is the official book of Apache HBase, a distributed, versioned, column-oriented database built on top of Apache Hadoop.

Table of Contents

[Preface](#)

[1. Getting Started](#)

[1.1. Introduction](#)

[1.2. Quick Start](#)

[1.2.1. Download and unpack the latest stable release.](#)

Six Deploys Lessons Learned

Variety of deploys
Variety of experience levels





StumbleUpon

- *“StumbleUpon helps you discover interesting web pages, photos and videos recommended by friends and like-minded people, wherever you are.”*
- 1+B “stumbles” a month
- 20M users and growing
 - Users spend ~7 hours a month ‘stumbling’
- Big driver of traffic to other sites
 - In US, more than FB and Twitter



HBase @ SU

- Defacto storage engine
 - All new features built on HBase
 - Access is usually php via thrift
 - MySQL is a shrinking core ('legacy')
- HBase is down, SU is down
- 2 1/2 years in production
- Long-term supporter of HBase project



HBase: The Enabler

- *A throw nothing away culture*
- Count and monitor everything culture
- Developers don't have to 'think' about...
 - Scaling, schema (easy iterate), caching
- Streamlines dev.... because small ok too



HBase: In Action

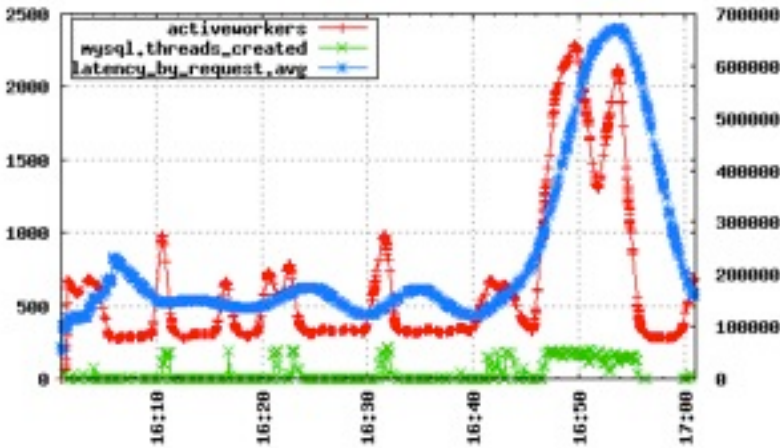
- Everyone uses HBase (SU is eng-heavy)
 - From sophisticated to plain dumb distributed hash Map/Queues
- Platform support team small
 - Ergo, our setup is a bit of a mess
 - ~250 tables on low-latency cluster
- Replicate for backup and to get near-real time data to batch cluster




Lessons Learned

- Educate eng. on how it works
 - E.g. Bad! Fat row keys w/ small int values
- Study production
 - Small changes can make for big payoff
 - Aggregating counters in thrift layer
 - Merging up regions so less is better

OpenTSDB



- Distributed, scalable Time Series Database
- Collects, stores & serves metrics on the fly
 - No loss of precision, store all forever
 - Runs on HBase
- Benoît Sigoure, devops at SU 
- Eyes and Ears on systems at SU > 1 year
 - Replaced Ganglia, Munin, Cacti mix

OpenTSDB Architecture

- Collectors on each host
 - An async non-blocking HBase client
 - Reverse engineered from scratch
 - Python's Twisted Deferred pattern
- One or more shared-nothing TSDB daemons
 - Chipmunk across HBase outages

OpenTSDB Stats

- 1B metrics/day @ SU
- 130B (and rising) metrics, just over 1TB
- Compact and compacting schema
 - Three-bits per metric or attribute
 - 2-3 bytes per datapoint
 - Roll ups the hour (6x compression)
- Read compacted metrics at 6M/second

Lessons Learned

- Play to the HBase data model
 - TSDB queries scans over time ranges
- Obsessing over schema and representation
 - Big payoffs in storage and perf

Realtime Hadoop

- *“Recently, a new generation of applications has arisen at Facebook that require very high write throughput and cheap and elastic storage, while simultaneously requiring low latency and disk efficient sequential and random read performance.”*

Apache Hadoop goes Realtime at FB <http://borthakur.com/ftp/RealtimeHadoopSigmod2011.pdf>

- Facebook Messaging
- ODS (Facebook Metrics)
- Facebook Insights (Analytics)
- Others to come...

WorldTour! this summer

SIGMOD

The scale here brings on nosebleeds!

Facebook Messages

- Unifies FB messages, chats, SMS, email
- Appserver on HBase/HDFS+Haystack
- Sharded userspace
 - 100 node cells, 20 nodes a rack
- Started with 500M users
 - Millions of messages and billions of instant messages per day
- Petabytes

Lessons Learned #1

- Had HDFS expertise and dev'd it in HBase
 - Willing to spend the dev to make it right
- **Studied cluster in production!**
 - Added many more metrics
 - Focus on saving iops and max'ing cache
- Iterated on schema till got it right
 - Changed it three times at least
 - MapReduce'd in-situ between schemas

Homogeneous
single-app use-case

Lessons #2

- After study, rewrote core pieces
 - Blooms and our store file format
 - Compaction algorithm
- Found some gnarly bugs
- Locality -- inter-rack communication can kill
- Big regions -- GBs -- that don't split
 - Less moving parts

FB Parting Note

- Good HBase community citizens
- Dev out in Apache, fostering community dev w/ meetups
- Messages HBase branch up in Apache



Web Crawl Cache

- Yahoo! cache of Microsoft Bing! crawl
- 'Powers' many Y! properties



WCC Challenge

- High-volume continuous ingest (TBs/hour)
 - Multiple continuous ingest streams
- Concurrently, wide spectrum of read types
 - Complete scans to single-doc lookup
- Scale to petabytes
 - While durable and fault tolerant



WCC Solution

- Largest ‘known’ contiguous HBase cluster
 - 980 2.4GHz, 16-core, 24 GB, 6 X 2TB
 - Biggest table has 50B docs (best-of)
 - Multi-Petabyte
- Loaded via bulk-load and HBase API
- Coherent “most-recent” view on crawl
 - Can write ‘out-of-order’
- In production since 2010/10



Lessons Learned

- Turn off compactions, manage externally
- Improvements to bulk loader
 - Parallelization, multi-column family
- W/o GC tunings, servers fell over



yfrog

Anthony Weiner in
boxers' with
equipment in outline
hosted by yfrog

- Image-hosting
 - “share your photos and videos on twitter”
- Images hosted in HBase cluster of 60 nodes
 - >250M photos ~500kb on average
 - ~0.5ms puts, 2-3ms reads (10ms if disk)
- Envable architecture -- simple
 - Apache=>Varnish=>ImageMagick=>REST=>HBase



yfrog

- NOT developers but smart ops
- VERY cost consious
 - All ebay “specials”, hbase nodes < 1k\$
- EVERYTHING went wrong
 - Bad configs
 - HW fails, nodes and switches



yfrog Issues

- App tier bugs flood HBase
- Bad RAM crash nodes for no apparent reason
- Bad glibc in FCI3 had race, crash JVM
- Slow boxes could drag down whole cluster
- Wrong nproc setting -- 1024 thread limit
- Auto-tuning GC ergonomically sets NewSize too small -- CPU 100% all the time



Lessons Learned

- More smaller nodes is better than less bigger nodes
- Lots of RAM is good up to a point
 - Avoid swap, big NewSize (Bigger CPU if can't clear it in time)
- Check network HW for packet drops
 - ping -f between nodes for packetloss
- Graphs: Trends over time critical



C.O.R.E

- Content Optimization & Relevance Engine
- HomePage 'Today' module, News, Finance
- 100GB of user feedback every 5 mins



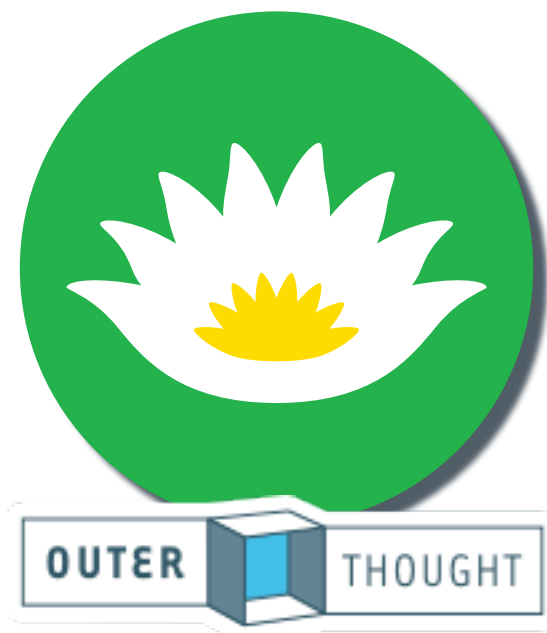
More C.O.R.E

- Built on HBase, MapReduce, & PIG
 - 300x2 node HBase cluster modeling
 - 150x2 node HBase cluster for analytics
- Billions of user events in HBase
- Machine learning modules applied to HBase data to predict next click
- Nitin Motgi, Amit Phadke, Douglas Campbell & Albert Shau



Lessons Learned

- Keep up w/ latest HBase
- Understand data and key distribution
 - Design schema accordingly
- Use filters to reduce returned data
- Manage major compactions externally
- Monitoring



Shoutout: lily

- Scalable Content Repository
- Open-source (ASL)
- Data stored and archived in HBase
 - Indexing Library for secondary indices
- HBase RowLog for SOLR WAL & queueing
- Real live customers!

APACHE
HBASE



Apache
Solr

Conclusion

- Know how HBase works
 - Play to the data model and not against it
- Understand your data; schema accordingly
- HBase is the canary when HW issues
- Be prepared iterate on configs, schemas
 - And to add customizations or backport
- Devops rather than ops/dev firewall

Thank you

- stack@apache.org
- hbase.apache.org

