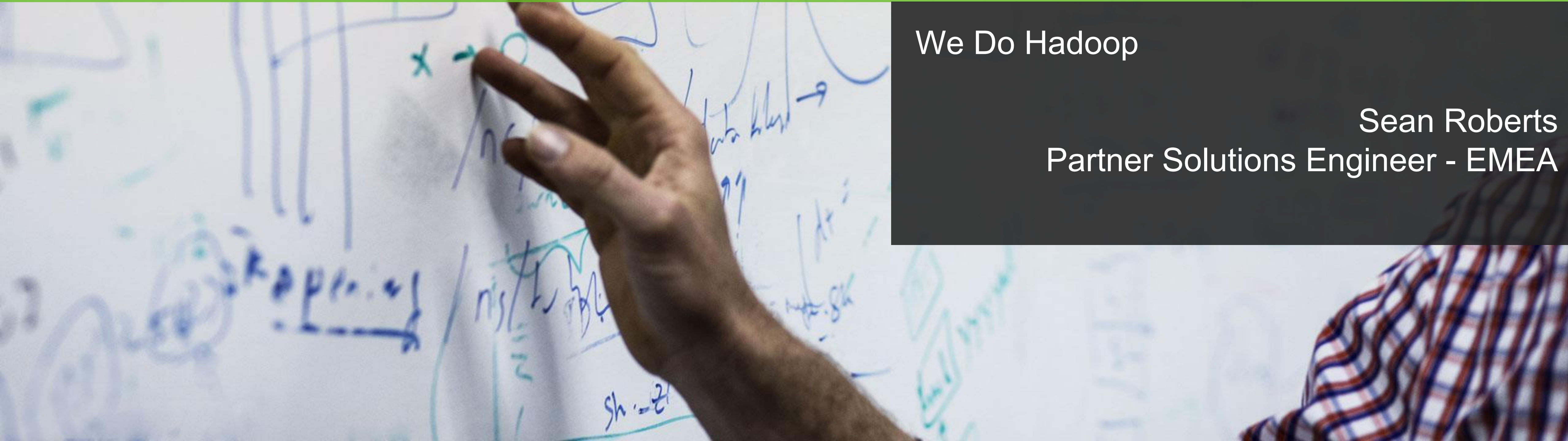
Hortonworks

Operational Best Practices Workshop

We Do Hadoop

Sean Roberts
Partner Solutions Engineer - EMEA



$(whoami)

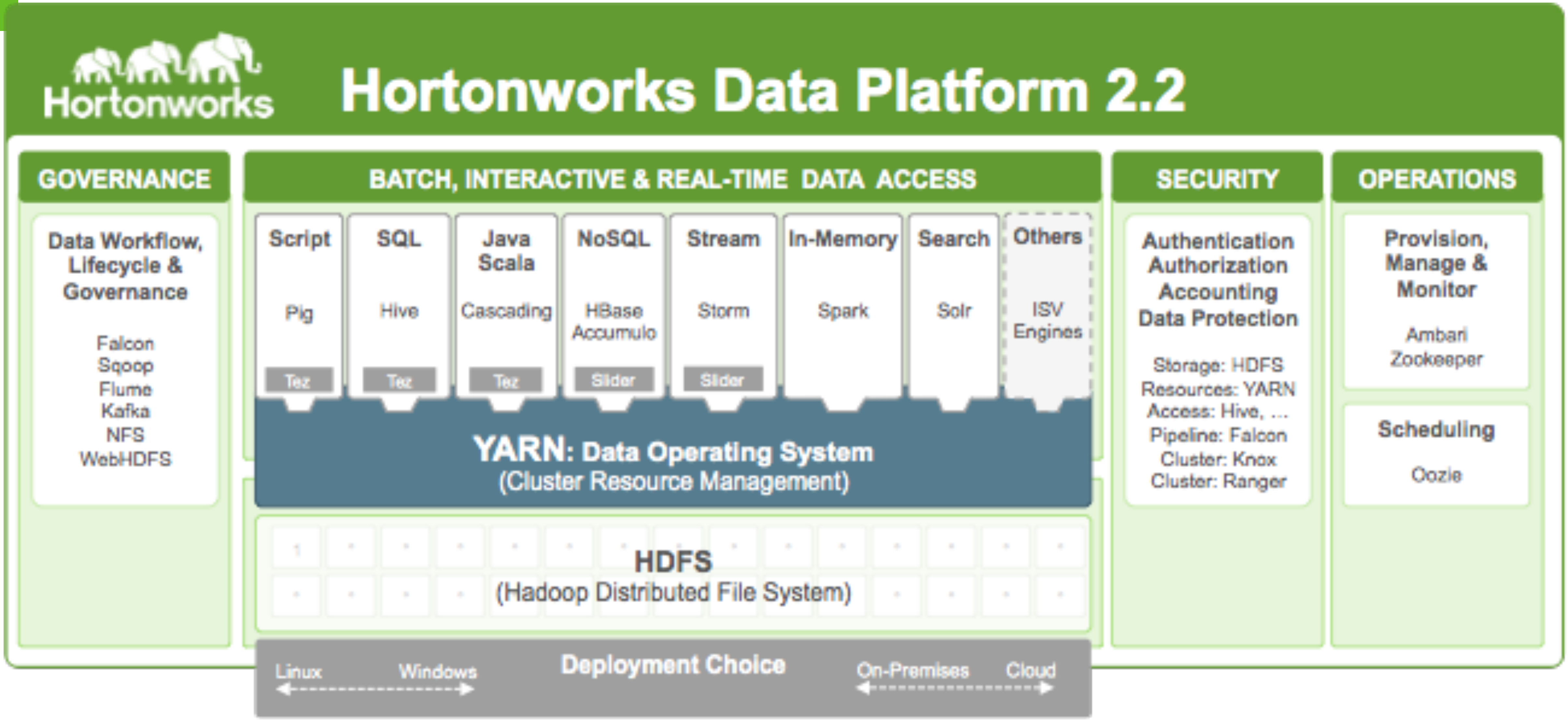
Sean Roberts

Partner Solutions Engineer
London, EMEA & everywhere

@seano

Operations

- Plan
- Provision & Deploy
- Secure
- Manage
- Workshop: Multi-Tenancy


Hortonworks
Hortonworks Data Platform 2.2
GOVERNANCE
Data Workflow, Lifecycle & Governance
Falcon
Sqoop
Flume
Kafka
NFS
WebHDFS
BATCH, INTERACTIVE & REAL-TIME DATA ACCESS
Script
Pig
Tez
SQL
Hive
Tez
Java Scala
Cascading
Tez
NoSQL
HBase
Accumulo
Slider
Stream
Storm
Slider
In-Memory
Spark
Search
Solr
Others
ISV Engines
YARN: Data Operating System
(Cluster Resource Management)
HDFS
(Hadoop Distributed File System)
Deployment Choice
Linux
Windows
On-Premises
Cloud
SECURITY
Authentication
Authorization
Accounting
Data Protection
Storage: HDFS
Resources: YARN
Access: Hive, ...
Pipeline: Falcon
Cluster: Knox
Cluster: Ranger
OPERATIONS
Provision, Manage & Monitor
Ambari
Zookeeper
Scheduling
Oozie

HDP - Runs Everywhere

On-Premises

HDP on Your Hardware

- Linux
- Windows

HDP on Appliance

Turnkey Hadoop Appliances

- Teradata
- Microsoft
- PSSC Labs

Cloud

Infrastructure as a Service (IAAS)

- Microsoft Azure
- Amazon EC2
- Rackspace
- Google Compute

Hadoop as a Service (HAAS)

- Microsoft HDInsight
- Rackspace

Virtualized, Container, ...

Your deployment of Hadoop

- VMWare, HyperV, XEN, ...
- OpenStack
- Docker

Plan - Cluster Design

Design choices, component distribution, and an example

More is better

- More is better than bigger when it comes to nodes
  - Faster healing
    - *Nodes with larger storage* ***take longer*** *to recover*
  - More nodes = more resilience, parallelism, power!
  - More racks = ...

Component Layout

All recommendations should be weighed against use cases

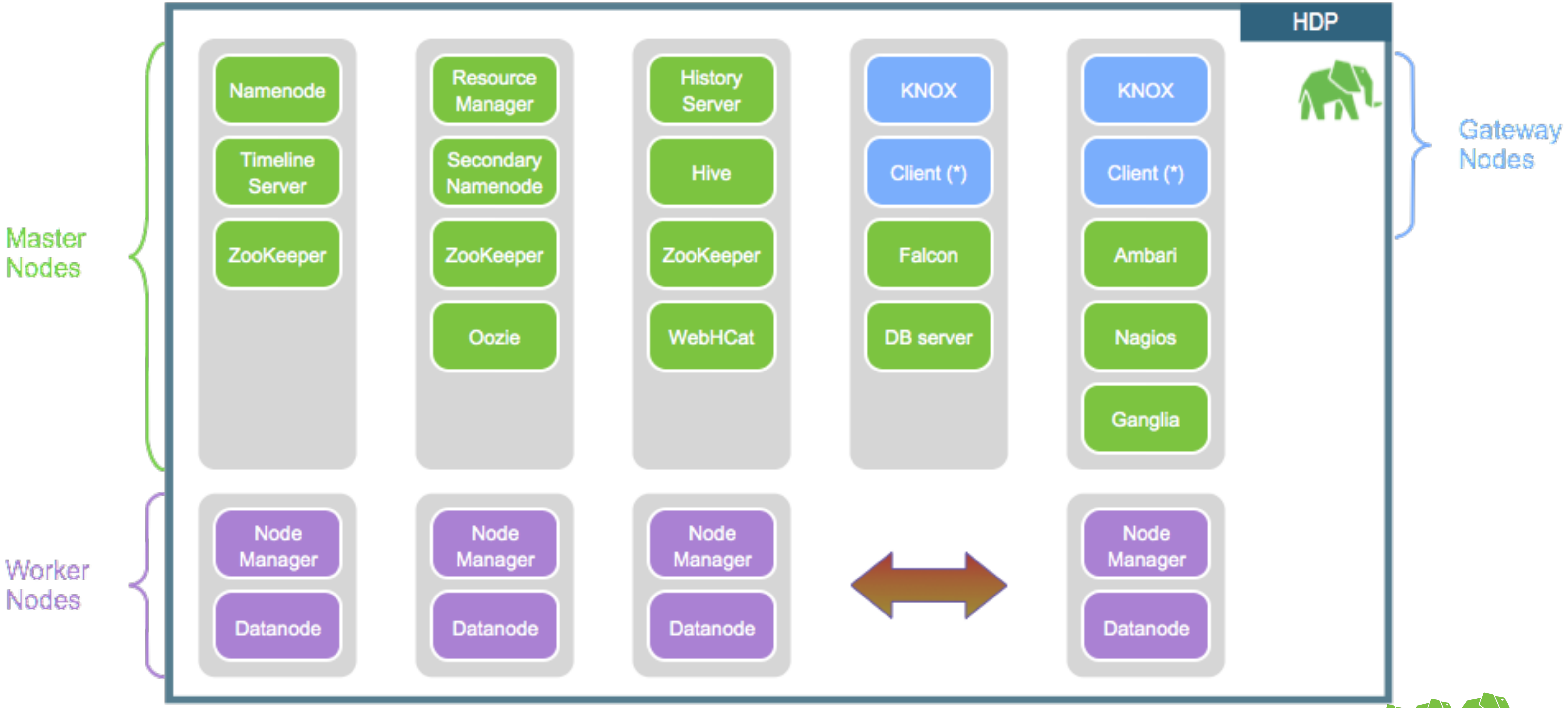
Master components

- Distribute across racks to spread risk
- As cluster grows, distribution will change

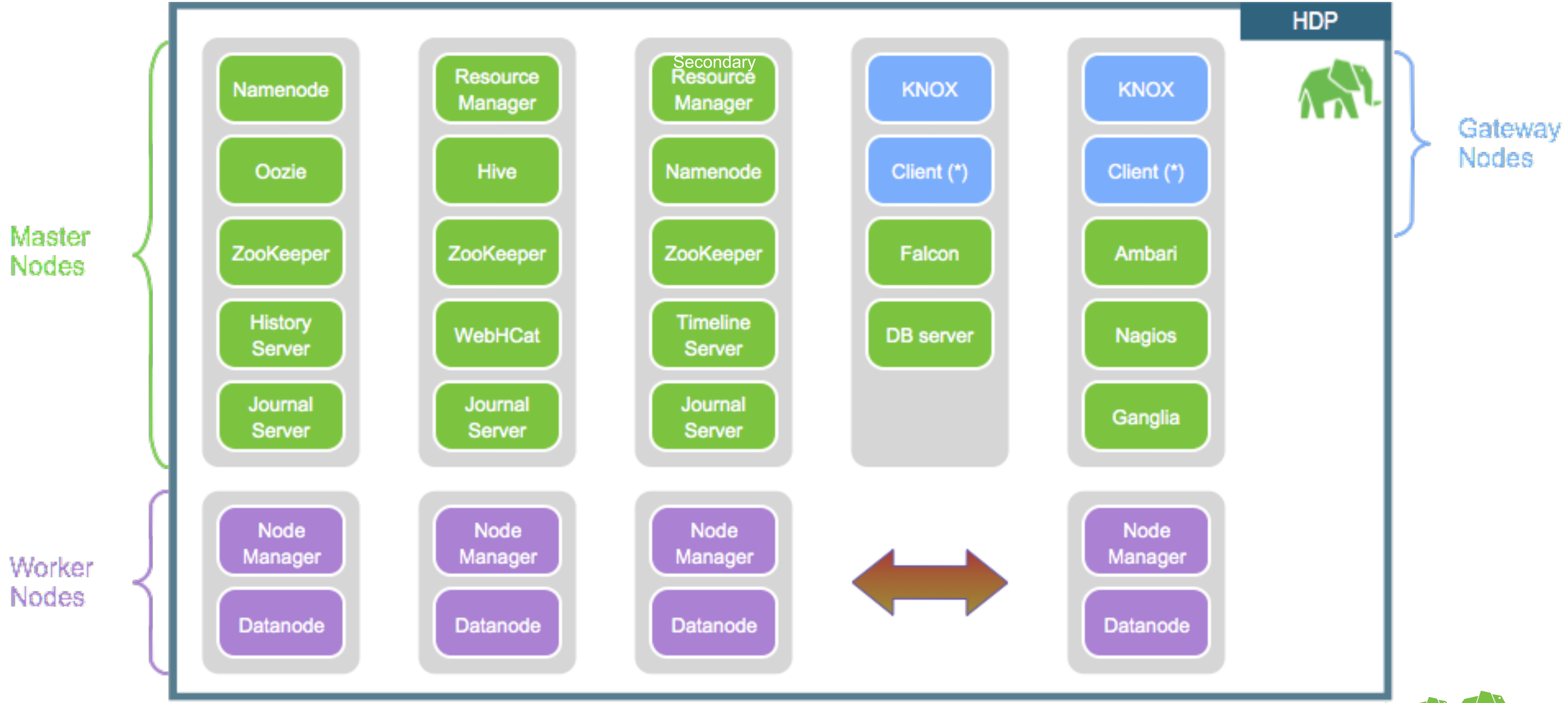
Worker components

- Typically identical across all worker nodes

Small Cluster without HA



Small Cluster with HA



Hortonworks®

Component Layout

Multiple gateway nodes *(load balanced)*

- HA/distribute client services: SSH, Knox, Ambari View instances, data loaders,
- Host other services: ntp, package repositories, Ambari View servers

5 ZooKeepers for greater reliability *(default is 3)*

- Also eases taking them down for maintenance
- >5 will be slower due to more voters in elections
- Consider fast disks (SAS, SSD)

Some like Virtualized Masters

- Master services are not typically IO heavy / SAN compatible
- Eases hardware replacement *(live vm migration)*

Layout of External Databases

Several components require external databases:

- **Ambari**: PostgreSQL
- **Hive**: MySQL
- **Oozie**: Derby
- **Ranger:** MySQL

Supported: Oracle, MySQL, PostgreSQL

- Consider using same technology for ease of management
- Use the same servers where possible
- Let your DBAs manage so backups, HA, … are taken care of

Heavy usage of Falcon+Oozie or Ambari, may require dedicated instances

Development cluster

Your development processes still apply

Ideally have a separate cluster for dev/test

- Many have dev/test combined
- Cloud: for scale up/down depending on need
- Often the pilot cluster becomes the dev cluster

Smaller but same configuration

- e.g. If prod has Kerberos, HA, etc., then dev should too

Plan - Hardware Selection

find your sweet spot

Sizing: Typical Worker

CPU

- Dual socket / 8-12 core each

RAM

- Typical: 128GB
- Not uncommon: 256GB

HDFS Storage

- 8-12 x 2-3TB *(NL-SAS/SATA)*
- 1TB for performance focus
- 4TB+ for storage archive focus

Hardware sizing

What is your workload?

- Balanced: most common
- Memory: such as Spark
- Compute: such as Storm
- Storage: such as archive

Mixed hardware in a cluster

- Ambari Configuration groups
- YARN Labels: Pin processing to subset of hardware
- Heterogeneous Storage

Sizing: Storm & Kafka

Storm & Kafka

- Deploy together
  - Storm is **compute** bound
  - Kafka is **disk** bound
- Same hardware as typical worker
- RAM: 128GB minimum
- Disks: 4-6 disks enough for most Kafka workloads
  - Assuming log retention of 2-3 days
- *Nimbus should be on a master with HA hardware*

Storage Configuration

Master nodes

- RAID-10
- O/S + Data disks

Data nodes

- Single O/S disk or RAID-1
  - Consider RAID-1 in small clusters where losing a single node could be a problem

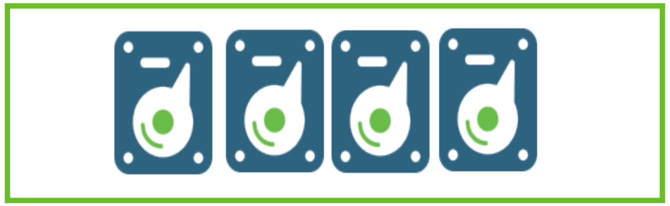
Data node HDFS

- RAID-0 per disk or JBOD
  - e.g.12 disks = 12 x 1 disk arrays
  - *Ensure they are mounted separately!*
- Data nodes survive disk loss:
  - `` `dfs.datanode.failed.volumes.tolerated` ``

Storage Types

Before

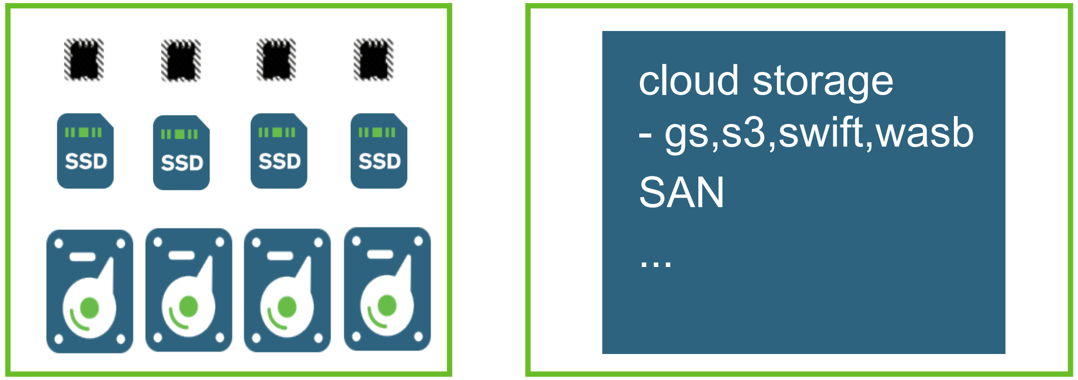
- DataNode is a single storage
- Storage is uniform - Only storage type Disk
- Storage types hidden from the file system



All disks as a single storage

New Architecture

- DataNode is a collection of storages

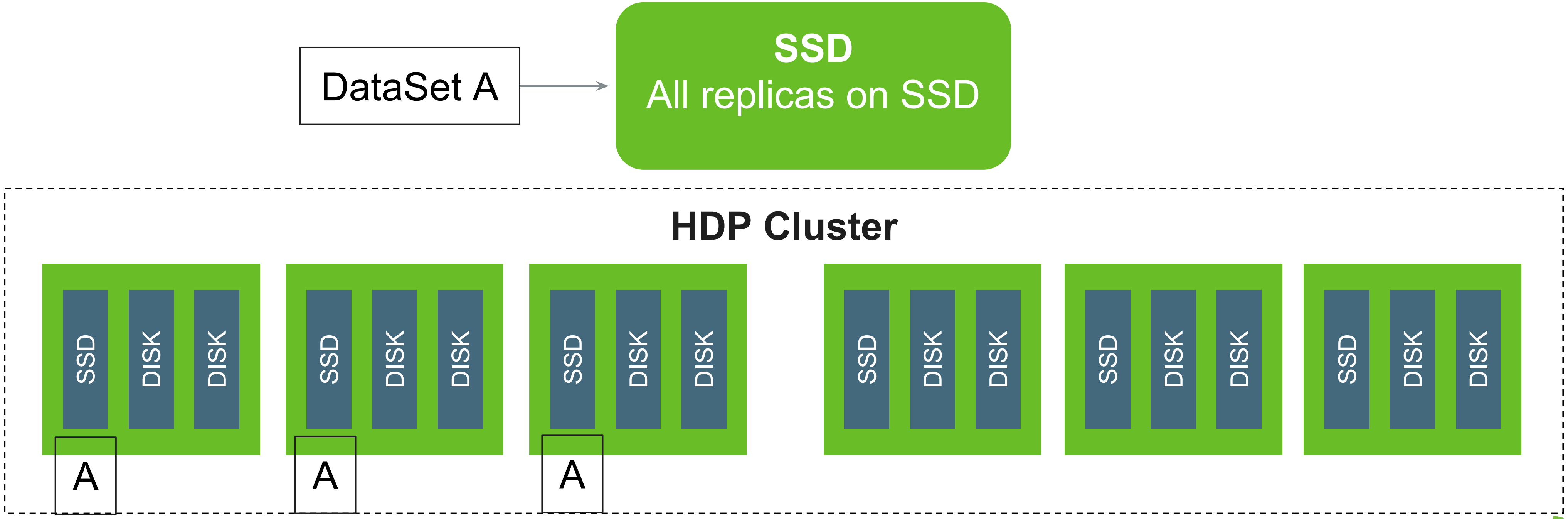


cloud storage
- gs,s3,swift,wasb
SAN
...

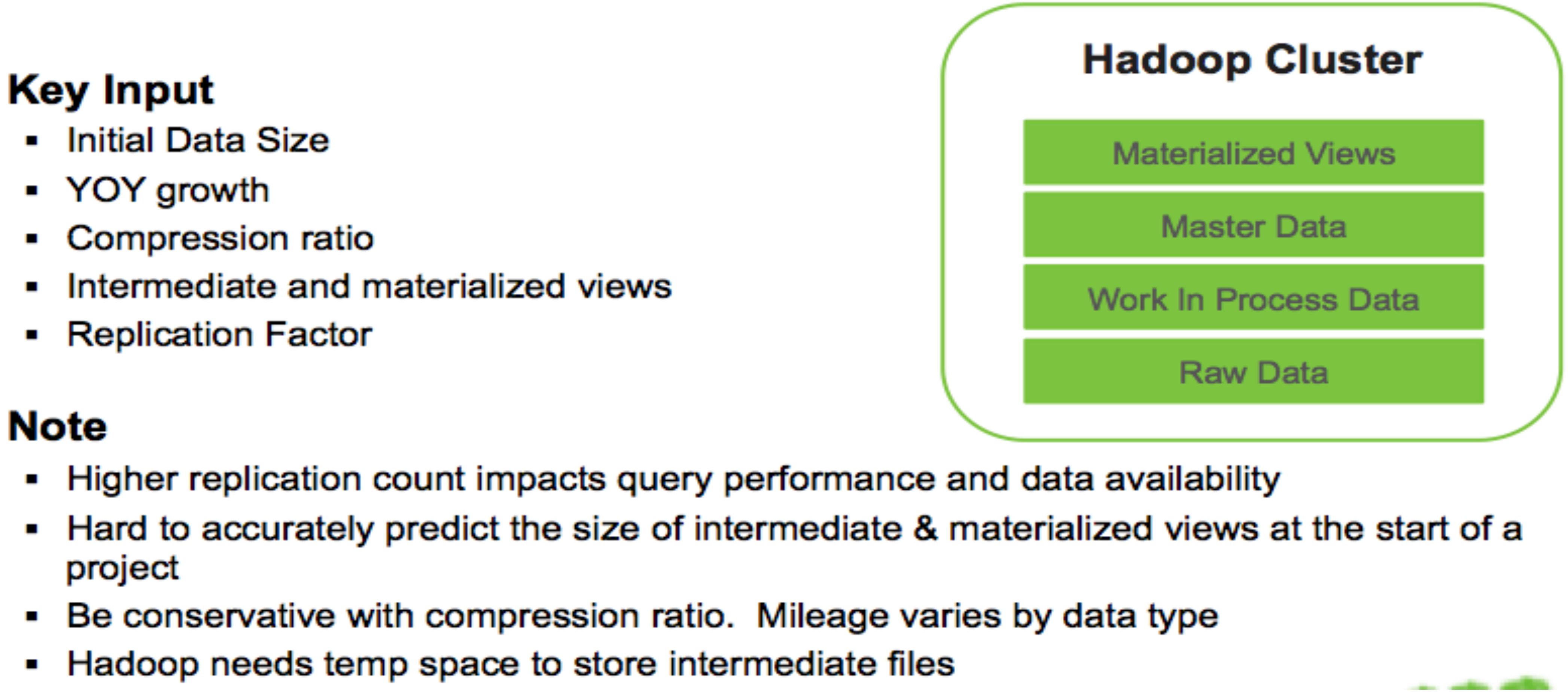
Collection of tiered storages

Storage Types & Policies

Archival Storage
Flash Storage
Single replica In-Memory Tier (Tech Preview)



Storage Calculator



Key Input

- Initial Data Size
- YOY growth
- Compression ratio
- Intermediate and materialized views
- Replication Factor

Note

- Higher replication count impacts query performance and data availability
- Hard to accurately predict the size of intermediate & materialized views at the start of a project
- Be conservative with compression ratio. Mileage varies by data type
- Hadoop needs temp space to store intermediate files

Hadoop Cluster

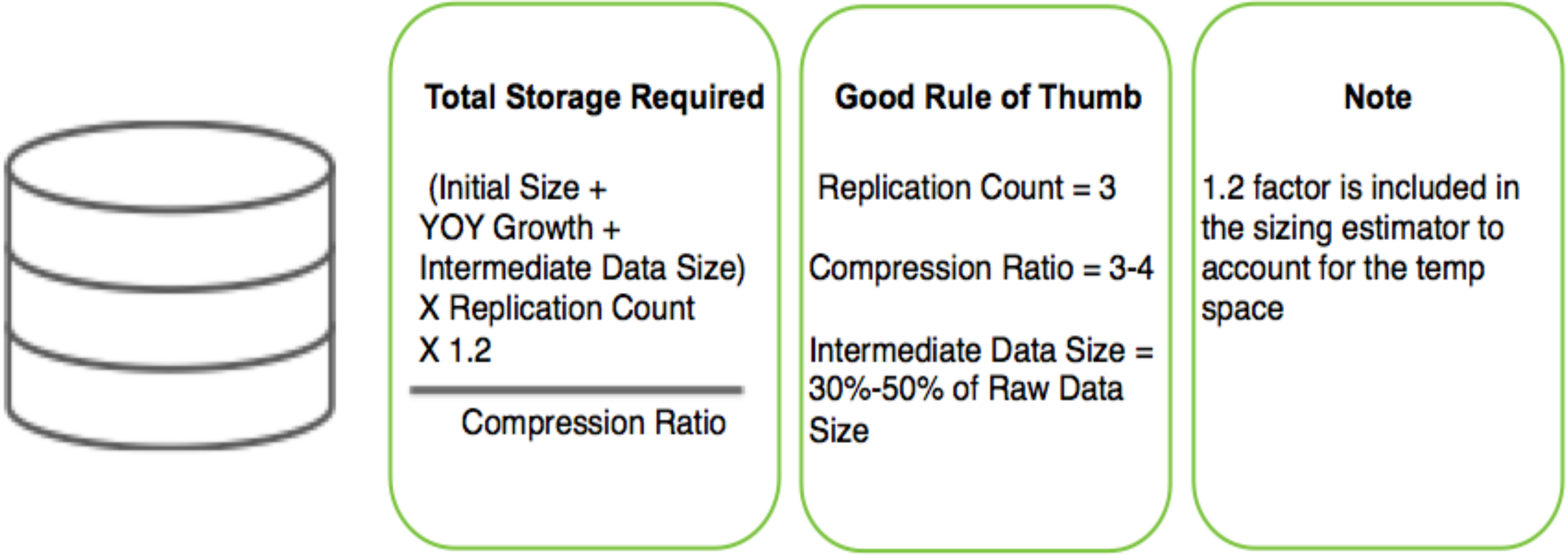
Materialized Views

Master Data

Work In Process Data

Raw Data

Storage Calculator



Total Storage Required

$$\frac{(\text{Initial Size} + \text{YOY Growth} + \text{Intermediate Data Size}) \text{ X Replication Count X } 1.2}{\text{Compression Ratio}}$$

Good Rule of Thumb

Replication Count = 3

Compression Ratio = 3-4

Intermediate Data Size = 30%-50% of Raw Data Size

Note

1.2 factor is included in the sizing estimator to account for the temp space

Network design

- Be prepared for overhead from node failure
    - Example: a single data node with 10 TB fails
        - The cluster will produce ~10 TB of network traffic to recover

- Typically
    - Data nodes: Bonded 1 GB or Single 10GB
        - *Might want to go with 10GB to future proof*
    - *Switches dedicated to the cluster only!*
    - Rack Interconnect: 2 x 10 GB
    - 2 TOR *(top of rack)* switches, or multiple spines so there is no SPOF

Provision & Deploy

Unix-y

Remember: Much of Hadoop follows “Unix-like” semantics

- Keep it simple. Often you simply do things the Unix way.
- That’s *a good thing!*

e.g. HDFS depends on system users & groups

Automate!

- **You must automate!**
  - Successful Hadoop clusters quickly reach to 100s or 1000s of nodes.
    - Data nodes are largely identical from an OS point of view
    - 10s of nodes are added at a time
    - Problem nodes are rarely fixed live, but instead decommissioned

- Use the OS automation you already have

  puppet, chef, ansible, cfengine, kickstart, gold images, just a bunch of scripts, ...

Provisioning workflow

- Infrastructure
- Base node OS & software configuration
- Automate installation & registration of Ambari
- Deploy with Ambari
- Choose what else your cluster needs
- Validate the cluster
- Use the cluster

Prepare your infrastructure

- Packages available locally
  - See our docs for mirroring our repos
  - Red Hat Channel, Spacewalk, apt mirror, …
    - *Automate the process of Red Hat registration!*
- Host name resolution
  - or DNS that is reliably maintained for all hosts *(forward & reverse!)*
  - /etc/hosts distributed & updated through automation *(don't do it manually)*
- Time server *(never virtualize this)*
  - On an edge node if you don't already have one

Prepare your nodes

Burn in!

- Have your provider do this if possible.
- Many drives fail: http://goo.gl/lCvkej
- fio, dd *(with direct & non-direct io)*

Don’t skip this

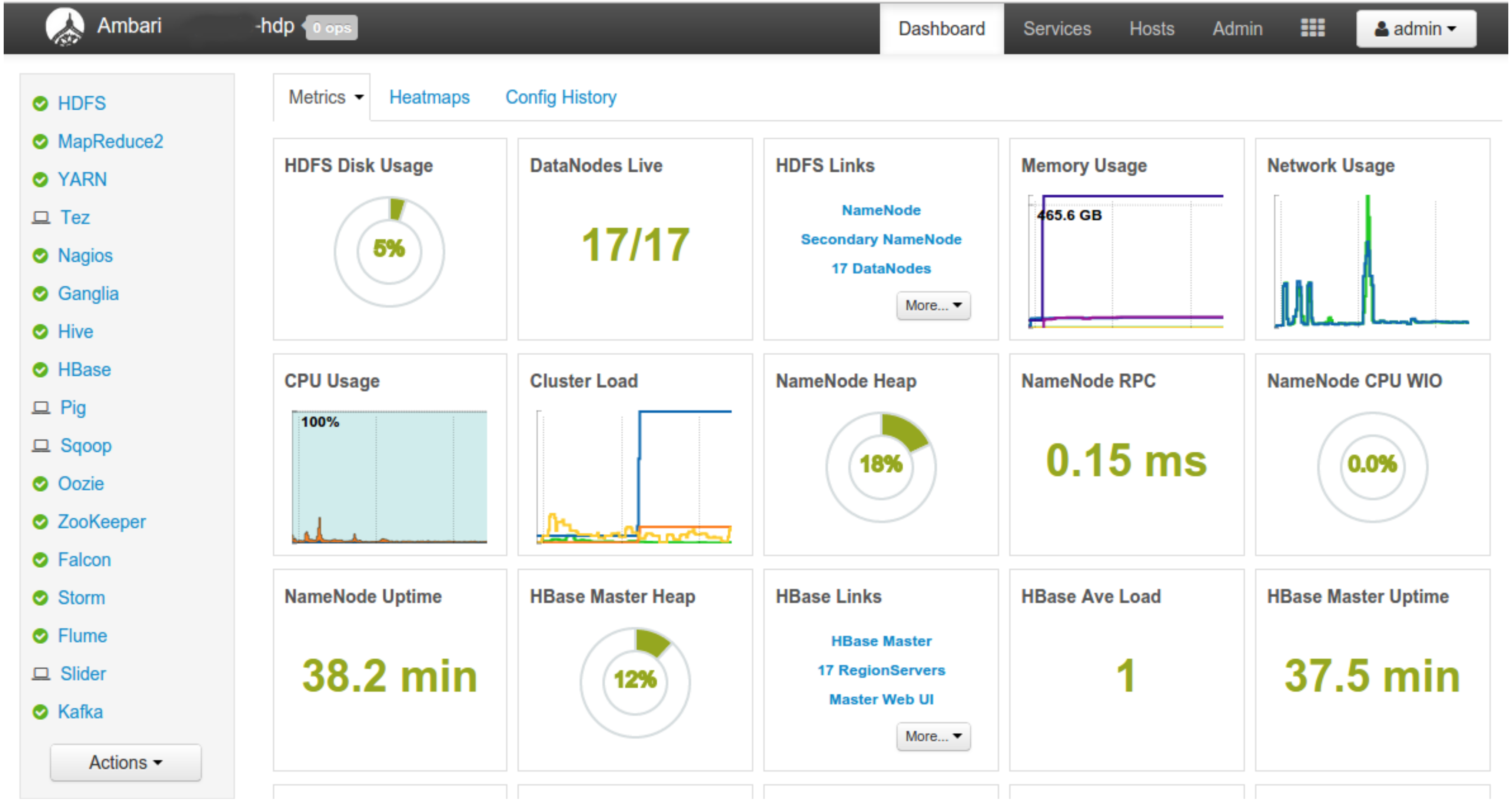
Prepare your nodes

- CPUs & Drive cache at same time: `hdparm -T /dev/sda`
- RAM
- Network negotiation:
  - Negotiation: `ethtool eth0 | grep Speed, mii-tool, dmesg`
  - Errors: `ifconfig |grep errors`
  - Performance: `iperf`
- Disk formatting & mounts
  - Choose filesystem: http://hortonworks.com/kb/linux-file-systems-for-hdfs/
  - Format: By default the file system reserves 5% of space for root!
  - Disable: *mkfs.ext4 -m 0* or *tune2fs -m 0*
  - Mount filesystem with ‘noatime’ to stop writing of access times
- Remove unneeded services *(cups, postfix, ...)*
- Names: /etc/hosts or DNS *(hostname -f & hostname -i): http://goo.gl/vRIOAZ*

Prepare your nodes - OS tuning

- *File Handler Limits:*
  - *raise nofile & nproc in /etc/limits.conf or /etc/security/limits.conf*
- *Disable transparent huge pages*
- *Disable IPv6*
- Name service caching (e.g. ncsd)
- Check BIOS power management settings
- Research IO scheduler *(deadline, cfq, noop)*
  - Choice varies on: SSDs, RAID card, virtualized, ...
- No swapping here: vm.swappiness=0 *(debated in newer kernels)*
- TCP Stack tuning: such as jumbo frame *(MTU 9k)*

Ambari



HDFS Disk Usage 5%

DataNodes Live 17/17

HDFS Links
NameNode
Secondary NameNode
17 DataNodes

Memory Usage 465.6 GB

Network Usage

CPU Usage 100%

Cluster Load

NameNode Heap 18%

NameNode RPC 0.15 ms

NameNode CPU WIO 0.0%

NameNode Uptime 38.2 min

HBase Master Heap 12%

HBase Links
HBase Master
17 RegionServers
Master Web UI

HBase Ave Load 1

HBase Master Uptime 37.5 min

Bootstrap Ambari

https://raw.github.com/seanorama/ambari-bootstrap

```
## install the ambari-server
pdsh -w server_public_hostname "curl -sSL ${bootstrap_url} | install_ambari_server=true
sh"

## install to all other nodes.

pdsh -w cluster0[2-3].hortonworks.com "curl -sSL ${bootstrap_url} |
ambari_server=server_private_hostname"
```

Ambari Considerations

Security

- Change password
- Adding Ambari users/groups
- Configure HTTPS
- Run as non-root

Ambari Agent communication

- Configure agents to self register *(during automated agent installation)*
  - or distribute SSH keys

Consider the database

Deployment: Ambari Blueprints

Definition of a cluster which enables:

- Repeatable model for cluster provisioning
  - allows for consistency
- Method to automate cluster provisioning
  - Enables ad hoc cluster creation for both bare metal and cloud
- Portable and cohesive definition of a cluster
  - Enables sharing best practices on component layout and configuration.

Hands On Demo: Deploy a cluster using a blueprint

Post deployment considerations

- NameNode HA
  - Additional dfs.namenode.data.dirs, possibly on NFS
- ResourceManager HA
- AD/LDAP integration
  - will increase adoption and ease user management
- Security integrations
- Capacity scheduler, multi-tenancy, ...
- Document how users access & get access to the system
  - create a mail list for your users

Cluster Validation

- Ambari Smoke Tests are run automatically
  - Run them manually
- Validation in the docs
  - HDP Install Documentation: http://goo.gl/USjn1g
- terasort
- Other common tests
  - DFSIO
  - HiBench
  - If using Cloud Object Storage connectors, check them *(gs, s3, swift, wasb)*

- Run a representative workload/job/...

Manage

and ongoing Operations

Tuning: Name node tuning

- Heap size needs to be tuned as the cluster grows
  - thumb rule: 200 bytes per object, i.e. file, block, dir
    - another rule: another 1.3GB of heap per 1PB of data
  - Young generation space ~⅛ of total heap *(maxed of 5GB)*
  - HDP Documentation: http://goo.gl/pA9wKC

- Use parallel GC *(garbage collection)*

Configuration Management

- Base OS *(network configs, DNS, system level configuration)*
  - Whatever you already use: Puppet, Chef, Ansible, cfengine, just a bunch of scripts, enterprise CM solutions, ...
- Hadoop
  - Ambari management configuration and also includes
    - versioning, history, compare, revert, recommendations

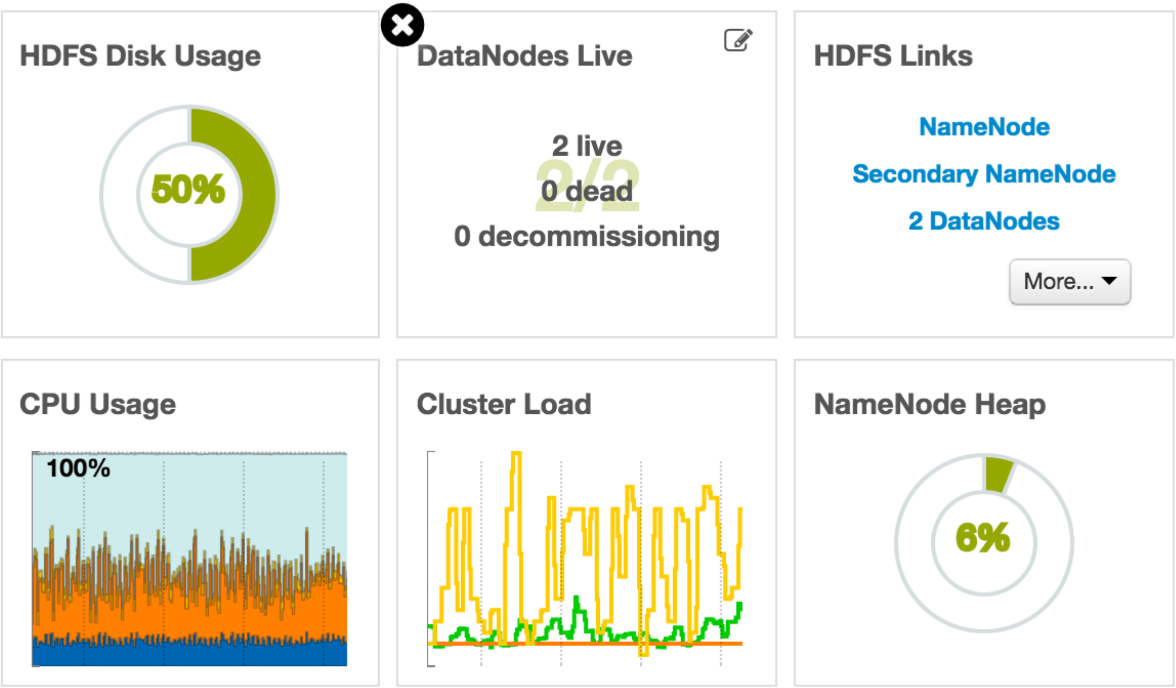
Monitor: What to monitor?

Integrate with existing monitoring toolsets

- System Center, Teradata ViewPoint,
- HP BSM, BMC & CA

Review Ambari Dasbboard

- HDFS Disk Usage
- DataNodes Health
- ...
- Review Alert & Metrics configuration



Some monitoring is application/database specific

- HBase, Storm

Hortonworks®

Monitor: HBase

Top 10 things to monitor per HBase table:

- callQueueLength
- compaction queue size
- memstore size
- slowHLogAppendCount
- get, mutation ops/sec
- mean, 95pct, 99pct latency
- GCTime
- CPU load *(proc.loadaverage.1min)*
- CPU allocation *(system, user, iowait)*
- blockcache size vs page cache size
- IO request counts *(IOPS)*

Monitor: Storm

Storm: Cluster wide metrics

- Nimbus availability
- Total slots available

For each Storm application

- Capacity: Alarm at >80%
- Latency: Alarm at deviation from expected
- Failed event count: Alarm with increasing number

Monitor: Kafka

- Disk space available
- Lag between reads and writes

Logs

Most troubleshooting is of job/task failure

- Typically drilling down to the machine and then the daemon log
- Most troubleshooting is specific to your application

Most used

- Audit: HDFS audit log - often forgotten about
- Ops: Component logs: /var/log/hadoop*
- Apps: Application logs: Land in HDFS /app-logs/

Often forgotten

- If you use Hive CLI instead of beeline: /tmp/<userid>/hive.log

Backup & HA

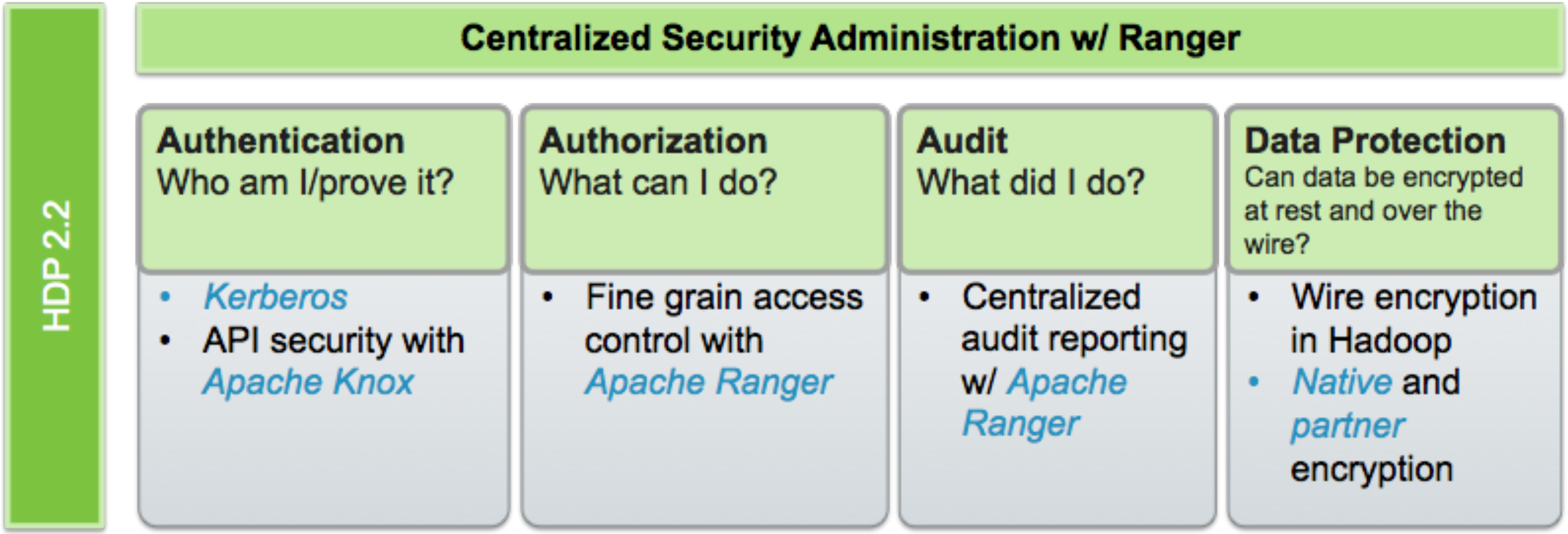
- On masters: Use your current backup method/service
  - All configuration *(/etc/, /usr/hdp, …)*
  - NameNode: Additional metadata mounts, some can be on remote NFS
    - dfs.namenode.data.dir
  - Databases: Ambari, Hive Metastore, Oozie, Ranger
- HDFS
  - Don’t disable Trash!
    - Configure the default expunge period (can be set per user)
  - Use Snapshots
  - All of HDFS: It depends.
    - Replicate to other clusters *(Falcon, distcp, …)*
- High Availability for Hadoop: http://goo.gl/BLXykB

Adding new nodes

- Do the same node checks, burn-in and prep
- Add in phases/groups
  - e.g. 3+ nodes at a time, half rack, or full rack
- Don’t forget to re-balance HDFS
  - Balance means:
    - % difference between storage utilisation of node and cluster
    - lower % threshold the more balanced the cluster
  - Can tune bandwidth allocation for rebalancing

Secure

Secure



HDP 2.2

Centralized Security Administration w/ Ranger

Authentication
Who am I/prove it?

- *Kerberos*
- API security with *Apache Knox*

Authorization
What can I do?

- Fine grain access control with *Apache Ranger*

Audit
What did I do?

- Centralized audit reporting w/ *Apache Ranger*

Data Protection
Can data be encrypted at rest and over the wire?

- Wire encryption in Hadoop
- *Native* and *partner* encryption

Kerberos in the field

Kerberos no longer *"too complex"*. Adoption growing.

- Ambari helps automate and manage kerberos integration with cluster

Use: Active directory or a combine Kerberos/Active Directory

- Active Directory is seen most commonly in the field
- Many start with separate MIT KDC and then later grow into the AD KDC

Knox should be considered for API/Perimeter security

- Removes need for Kerberos for end users
- Enables integration with different authentication standards
- Single location to manage security for REST APIs & HTTP based services
- *Tip: In DMZ*

Authorization/Audit/Protection

Ranger provides fined grain access control and auditing

- Provides policies for HDFS, Hive, HBase, Storm and Knox
- Integrates with LDAP/AD for users/groups
- Ranger installation is automated in Ambari 2.0
- *Tip: Consider using HDFS as audit store if volume is high*

Encryption/Data Protection

- Several options for encryption at REST, from disk to application level
- Fine grain encryption recommended, encrypt only sensitive data
- If enterprise-wide strategy is needed, use a Partner solution
    - (Voltage, Protegrity, Vormetric, DataGuise)

Security Workshop

Secure HDP 2.2/2.1 using FreeIPA LDAP

Recording & documented process for

- Authentication: Configure kerberos with LDAP on sandbox
- Authorization & Audit: access policies and audit around Hadoop from central Ranger UI, integrated with LDAP
- Perimeter Security: Configure Knox for kerberized cluster to enable perimeter security. Integrated with LDAP/Ranger
- Protection at rest: Setup Transport Data Encryption (TDE)

- http://hortonworks.com/partners/learn

Tenant onboarding

Provision, onboard and secure new tenants *(user groups, …)*

Multi-Tenancy & Tenant Onboarding

The request

- *“As an administrator, I want to quickly provision access to new and separate tenants across HDFS, YARN, Hive, HBase, Storm, Accumulo, Knox, ...”*

For many IT organizations, provisioning access is the most time consuming process *(not just for Hadoop).*

Multi-Tenancy & Tenant Onboarding

The request:

- *“As head of the Awesome Department, I must ensure my applications get appropriate resources and meet SLAs”*
  - *“The CogApp 4.0 is lower priority than SuperApp 2”*

YARN to the rescue

Multi-tenancy

Multi-tenancy is one cluster with

- Multiple Business Units
- Multiple Application/Jobs

YARN enables multi-tenancy with

- Shared Processing Capacity
- Shared Storage Capacity
- Data Access Security

YARN Capacity Scheduler

FUNCTION	**Capacity Sharing**	Queues with priorities Job submission Access Control Lists
FUNCTION	**Capacity Enforcement**	Max capacity per queue User limits within queue
FUNCTION	**Administration**	Management Admin accesses Access Control Lists Capacity Scheduler Configuration File (via UI in 2.1+ Release)

The YARN Platform Journey

	HDP 2.0	HDP 2.1	HDP 2.2
YARN Platform	Genesis of YARN Abstract OS from App Frameworks	Hierarchal Queues Preemption	CPU Node Labels CGroup Default Q mapping RM REST API
Workloads	• MapReduce v2	• MapReduce v2 • Tez (Hive) • Partners+	• MapReduce v2 • Tez(Hive, Pig, Cascading) • HBase • Storm • Partners++

■ ■ ■

Multi-Tenancy with Capacity Scheduler

Queues

- Hierarchical

SLAs

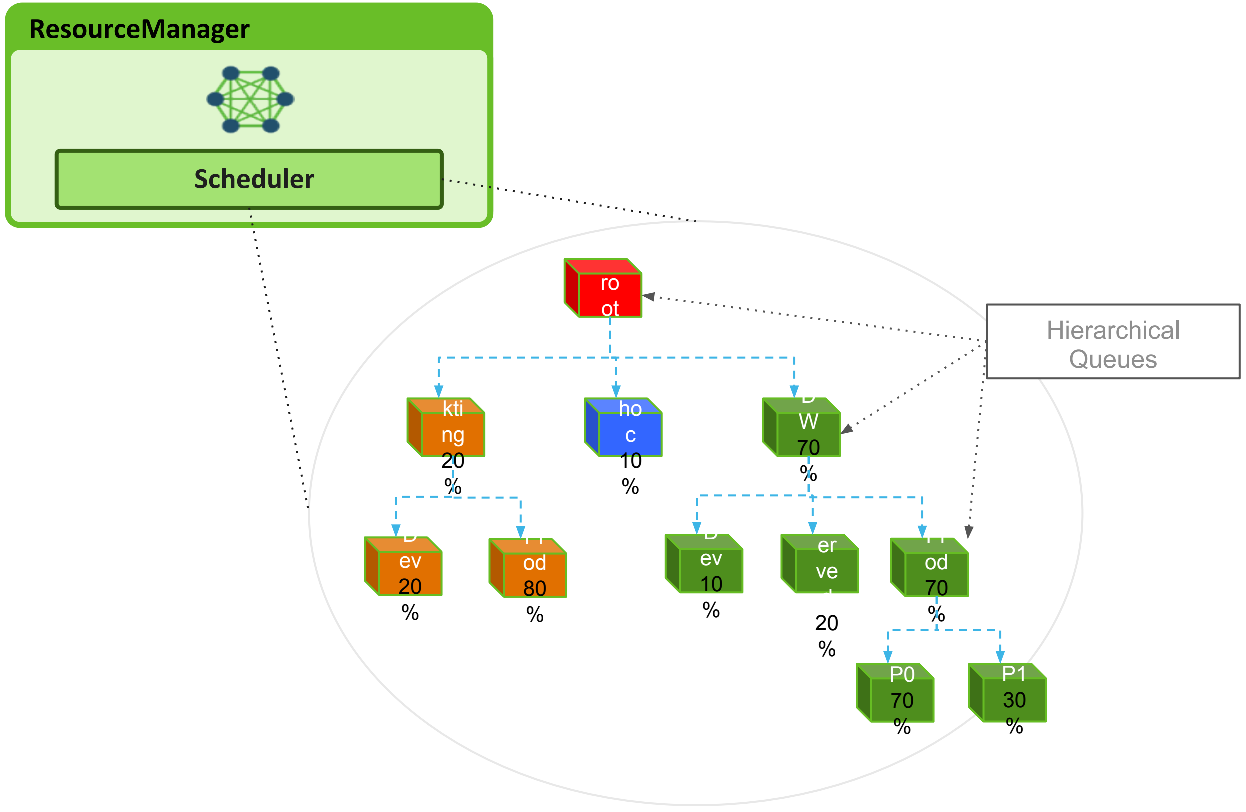
- Preemption based on priority

Resource Isolation *(Containers)*

- Linux: cgroups
- MS Windows: Job Control

Administration

- Queue ACLs
- Run-time re-configuration for queues



Hierarchical Queues

Capacity Scheduler

Default Queue Mapping

What

- Admin can define a default queue per user or group
- If no queue is specified for an application, YARN Capacity Scheduler will use the user/group's default queue for application

Why

- Queues are required for enforcing SLAs, make it easy to utilize queues
- Users and Applications want to submit Yarn apps/jobs and not have to specify queue
- Ease migration from Fair Scheduler usage

Default Queue Mapping

From Ambari, add to ‘custom yarn-site’:

- yarn.scheduler.capacity.queue-mappings

Format is: `[u|g]:[name]:[queue_name][,next mapping]`

For example: `g:mktg:marketing,u:etl:dataLoad`

Good use case is mapping users to a queue of their name/group:

u:%user:%user

Tenant Details

Tenant Details

- Tenant
  - name
  - description
  - status
  - users/groups
- Associated components: HDFS, YARN, Hive, HBase, …

Tenant Onboarding: HDFS

HDFS

- user validation: adding /user/${user} for each tenant user
- path for tenant in HDFS. e.g. /tenant/${tenant}
- default permissions and/or extended security (ACLs or Ranger)
- quotas
  - number of files
  - total allowed space (both raw & replicated)
- scheduled HDFS snapshots
- storage type(s)

Tenant Onboarding: Hive

Hive

- Create database(s)
- Database privileges of users associated with tenant
- Database visibility *(hidden or visible in catalog to other users)*
- Replication *(use of Falcon to replicate between clusters)*
- Scheduled snapshots of HDFS & Hive Metadata

Tenant Onboarding: YARN

YARN

- Top-level queue and min/max allocations
- Sub-queues, their allocations, user limits, …

Onboard a tenant

Live

Wrapping up

Partner Integration

Integrate through

- YARN & Slider: Simplified on-boarding of existing apps to Hadoop YARN
- Ambari Services/Stacks: Plugin new services that can co-exist with Hadoop
- Ambari Views: New ways to interact with Hadoop and visualize operations
- Ambari Blueprints/APIs: automate cluster setup in repeatable way
- Ranger plugins: manage authorization/audit of 3rd party s/w via Ranger UI

More details in upcoming Slider (Feb-26) and Ambari (Mar-26) webinars

Get started today using demos and code samples on Partners Learn page

Upcoming Workshops

Upcoming workshops (see Partners Learn page)

- Build YARN Ready Application with Apache Slider – Feb 26
- In Memory Processing with Apache Spark – Mar 12
- Ambari Stacks, Views and Blueprints Workshop – Mar 26

Upcoming meetup (see hortonworks.com/events) :

- Long Running Services on YARN using Slider - March 4

Support

Hortonworks Support

- Engineers with talent across the entire Hadoop ecosystem
- Escalate to subject matter experts for depth in a particular area
- Challenging cases may escalate to Apache committers at Hortonworks if additional expertise is required

Apache Community Support

- user questions and support: user@hadoop.apache.org
- reporting confirmed bugs: https://issues.apache.org/jira
- Users, contributors, committers & PMC members all participate actively in these forums to help resolve issues

Support

What we see

- Core Hadoop components *(HDFS, YARN and MapReduce)* are used across all deployments, and therefore receive proportionally more support cases than other ecosystem components.
- Misconfiguration is the dominant root cause.
- We are constantly improving the code to eliminate operational issues, help with diagnosis and provide increased visibility.

Thanks

http://hortonworks.com/partners/learn/

https://github.com/seanorama/workshop-hadoop-ops

Hortonworks