DriveScale

DriveScale Architecture for Data Center Scale Composability



by Brian Pawlowski,
Chris Unkel, Jim Hanko,
Jean-François Remy

Overview of talk

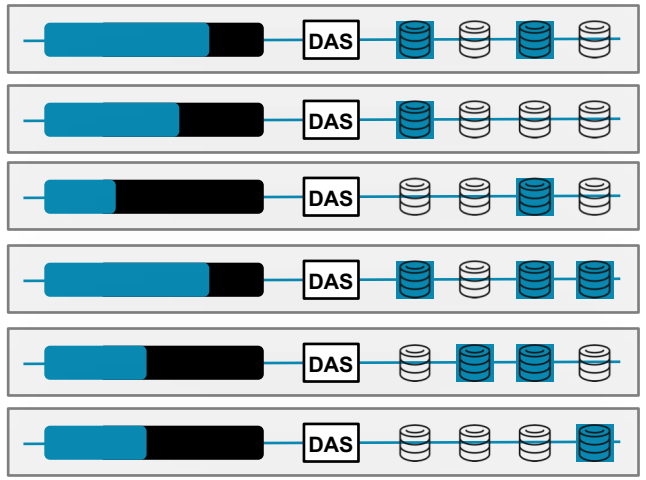
- Defining composable infrastructure
- What the administrator sees
- DriveScale design and architecture
- Future directions
- Recapitulation
- Questions

DriveScale

Defining composable infrastructure

DriveScale – Disaggregate then Compose

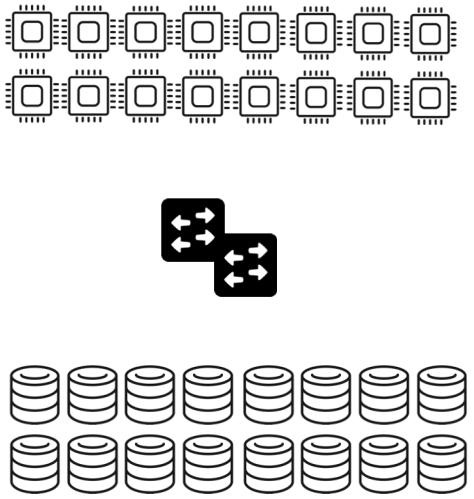
Captive, fixed DAS



Purchase Time
Defined Infrastructure

Disaggregate

Composable



Right sized

Software Defined Infrastructure

Composable Infrastructure – blade chassis scale

- Single vendor hardware
- Low scalability

10 Servers

100 Drives

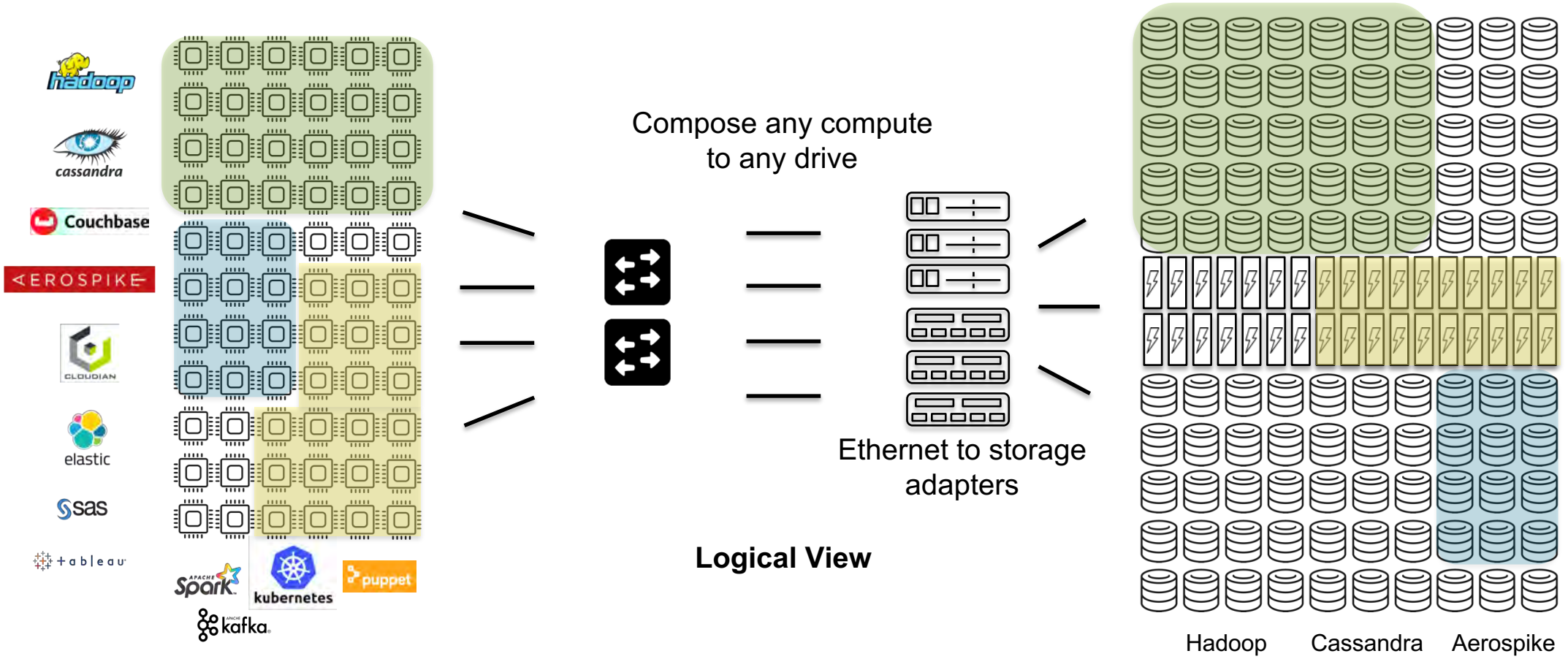
Composable Infrastructure – data center scale

10,000 Servers

100,000 Drives

- Choose optimized dense diskless servers, optimized storage platforms for HDD and SSD
- Multivendor – mix and match
- Multiprotocol Ethernet (iSCSI, ROCEv2, NVME over TCP) for wide variety of use cases
- Composer platform server(s) configuration: dual socket x86, 128GB DRAM, SSD

Composable Infrastructure at scale



DriveScale

Think Bare Metal Virtualized Infrastructure for Linux, at scale – but without the VMs.

The devil is in the details

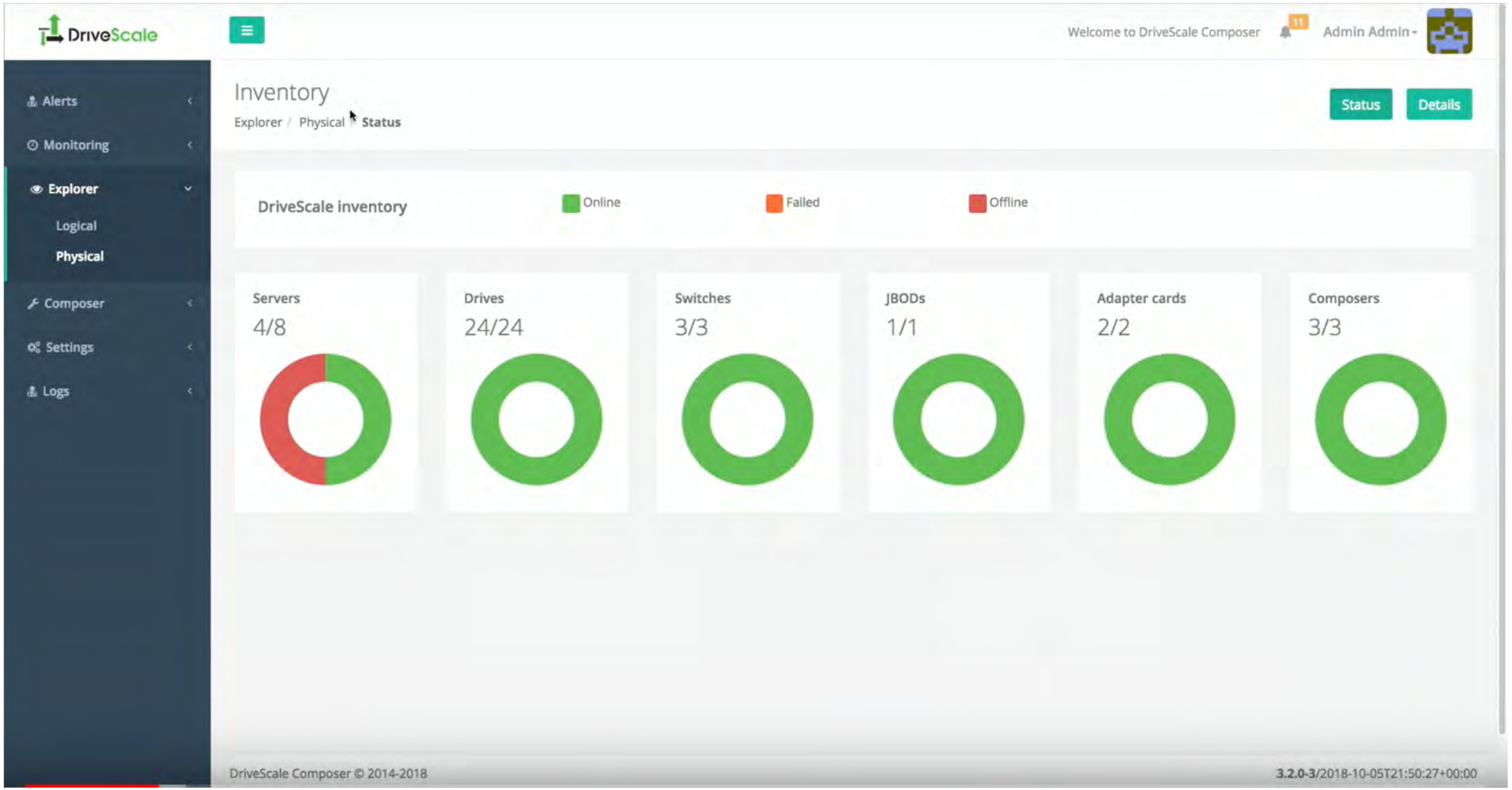
- Breaking up (disaggregation) is easy (in my pocket)
- Composable Infrastructure – at scale – is hard
  - Create *secure* durable bindings between servers and drives
  - Plumb end-to-end Linux storage stack over multiple protocols, multipathing, file systems, RAID configurations, hardware vendors and storage interconnect technologies, and encryption/key mgmt.
  - Persist server/drive bindings through rolling upgrades, multi-version software support, and failed data path component replacement
  - Non-disruptive cluster expansion
  - Retrying configuration changes in face of transient network errors
- Manage and automate complexity at scale

DriveScale

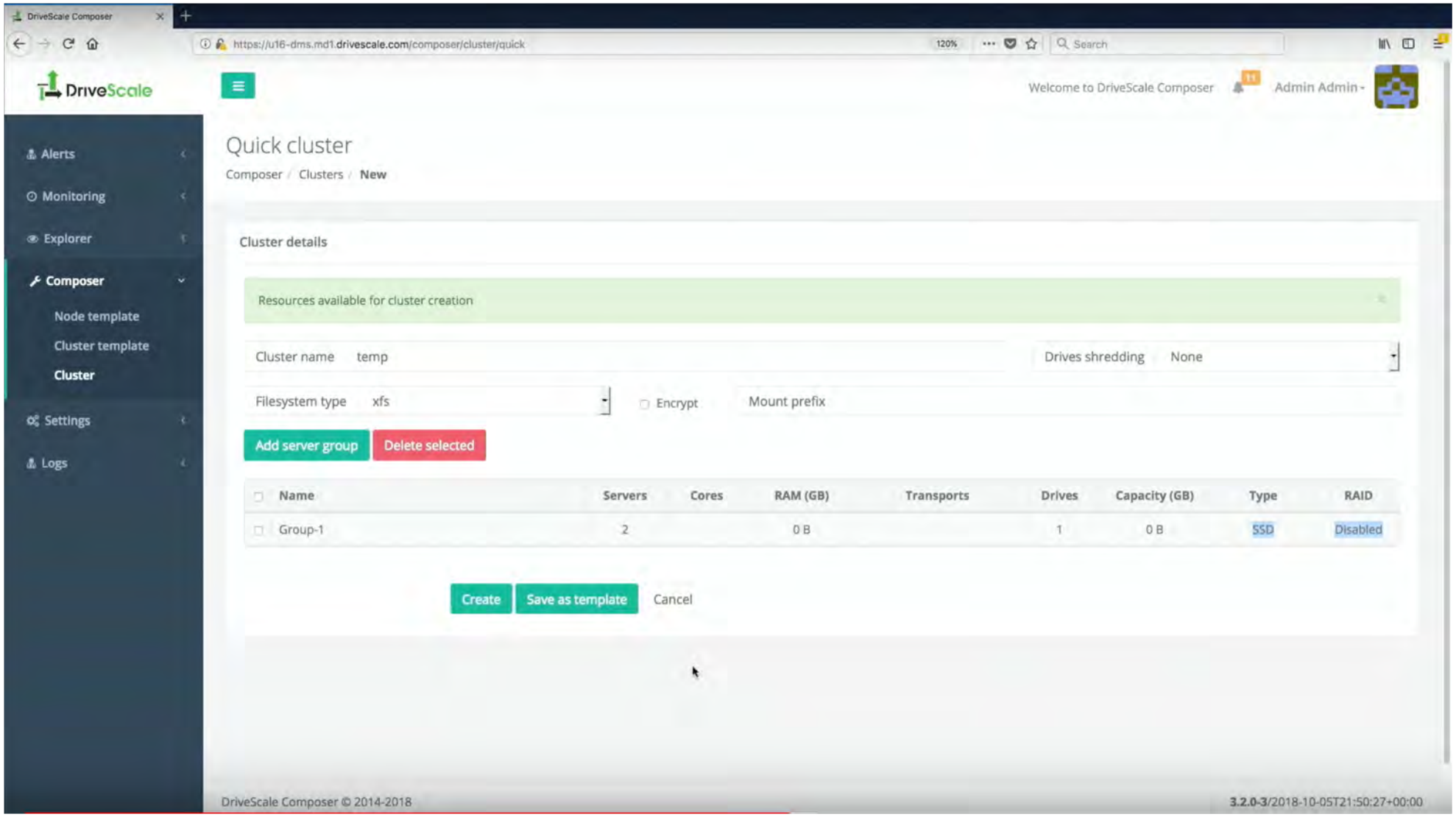
What the administrator sees

See demo at https://www.youtube.com/watch?v=_PPxyM-liQo

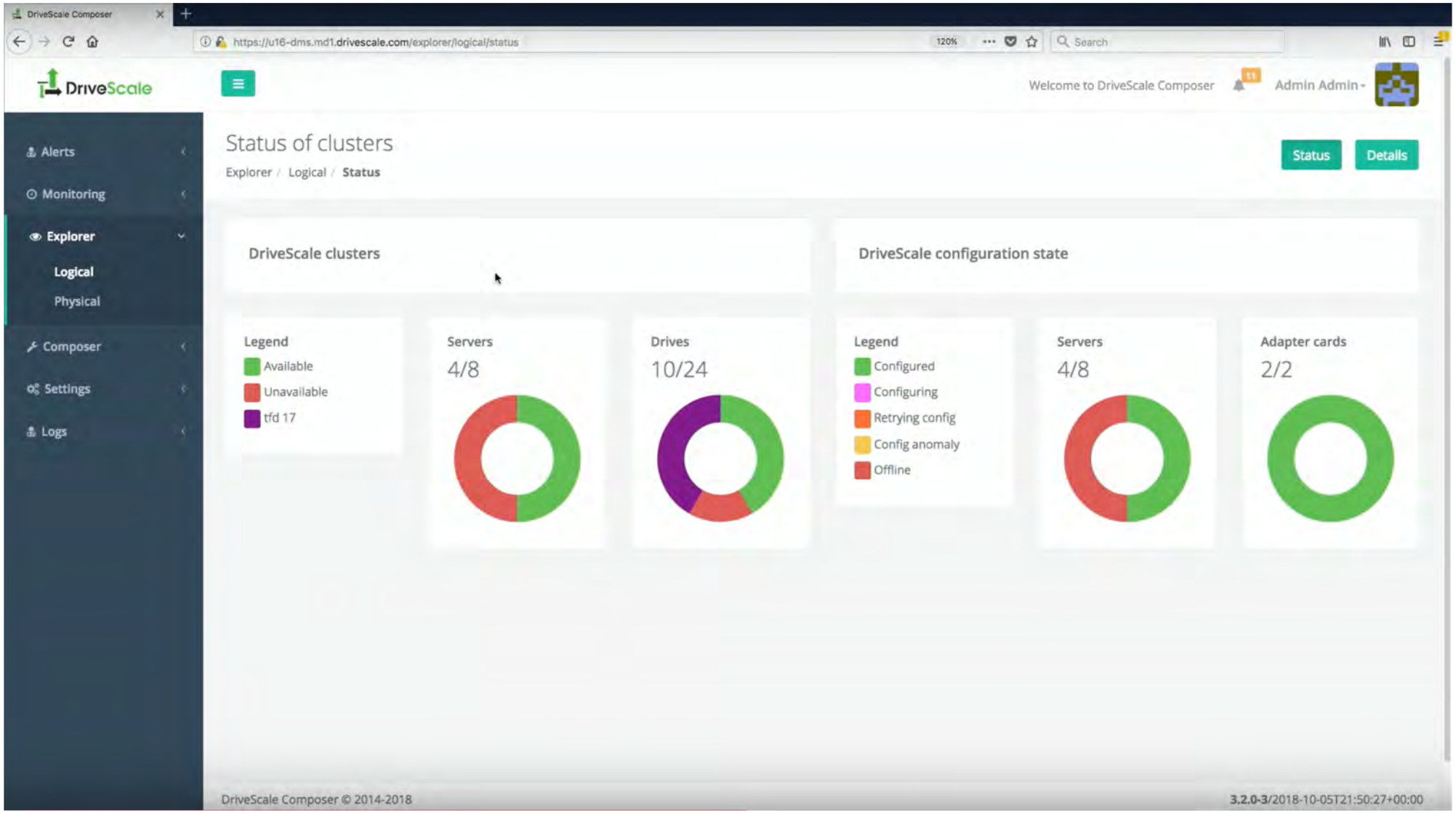
DriveScale Composer – Inventory



DriveScale Composer – Quick Cluster



DriveScale Composer – Logical Cluster



DriveScale

DriveScale design and architecture

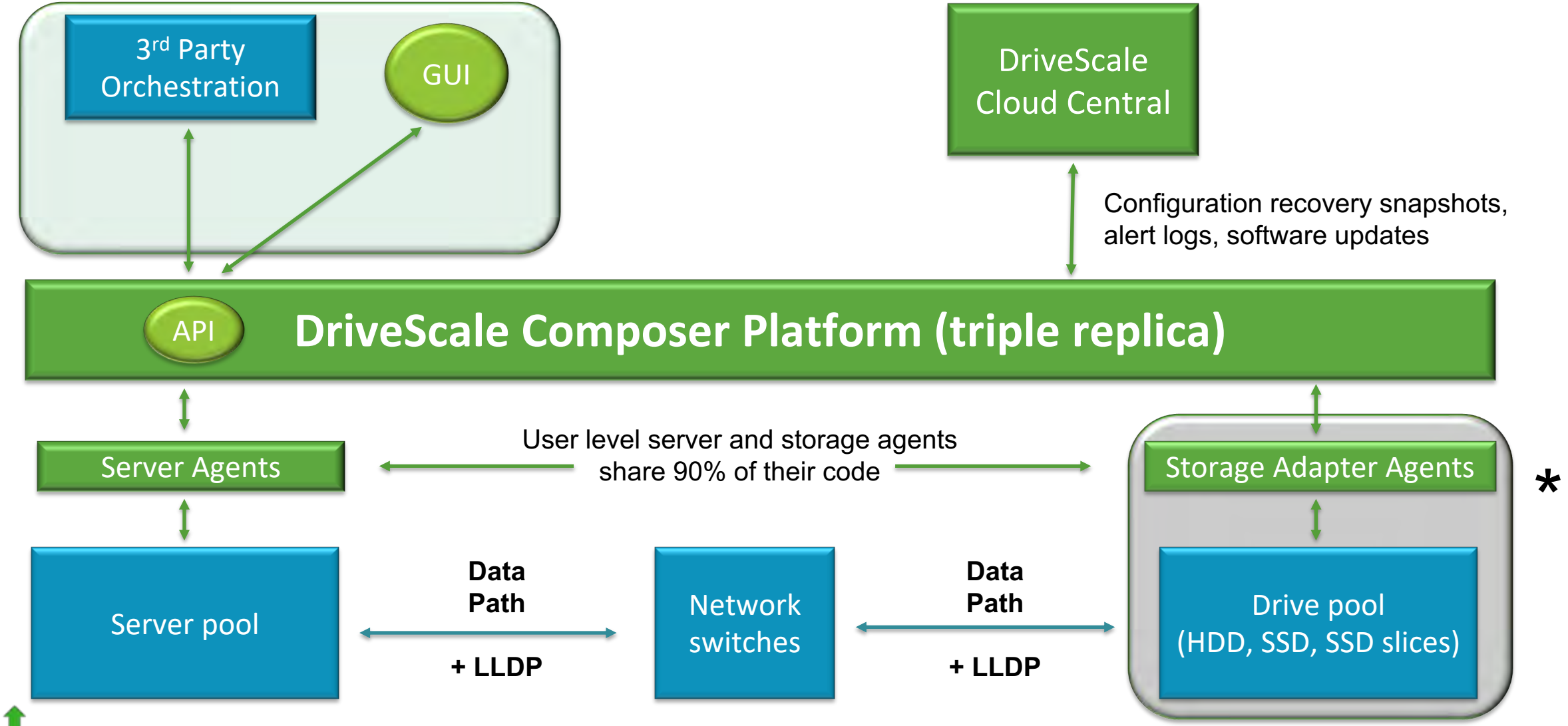
Terminology for this talk

Disaggregation	Breaking up of compute and storage into discrete elements on an Ethernet network
Composition	Or composability – the dynamic secure binding of servers to drives
Logical Cluster	*Secure* closed (non-overlapping) set of durable server and drive bindings used to host a single scale-out application
Configuration	The set of logical cluster and element configurations written by the DriveScale composer, read by agents
Inventory	The set of servers, drives, and storage adapters written by DriveScale agents. Ephemeral status znodes.
server	In composable deployment, a thin (diskless) compute node or element
drive	A whole HDD, SSD or an (SSD) slice – also called a *disk*
HDD	Hard Disk Drive – standard form factors are 2.5" and 3.5"
SSD	Solid State Drive (form factor irrelevant)
slice	A slice is a discrete persistent portion of a single SSD that is presented to a server as a single drive
storage adapter	An Ethernet (iSCSI or NVME) to JBOx adapter – presents drives (12, 24, 60, or 80 drives are typical sizes) – *simply* provides multipath access to drives for high availability
JBOx	Just a Bunch Of Disks (HDD) or Flash (SSD)
EBOx	Ethernet-attached Bunch Of Disks (HDD) or Flash (SSD)
ZooKeeper	A centralized service for maintaining configuration information, naming, providing distributed synchronization

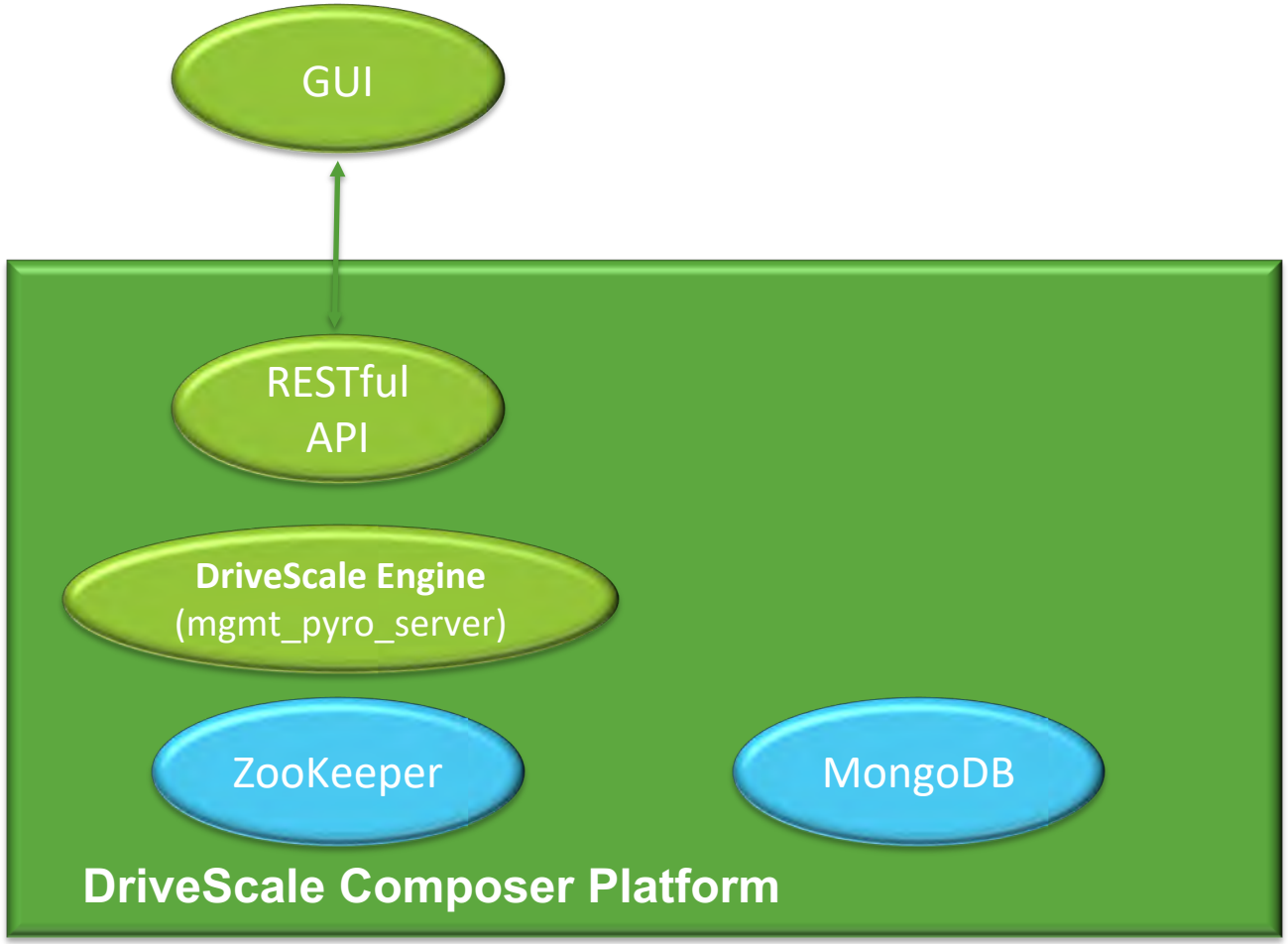
Design goals

- Scalable (key goal from which much followed)
- Durable
- Highly available
- Correct
- Performant
- Recoverable

DriveScale Composer Platform (control and data plane)

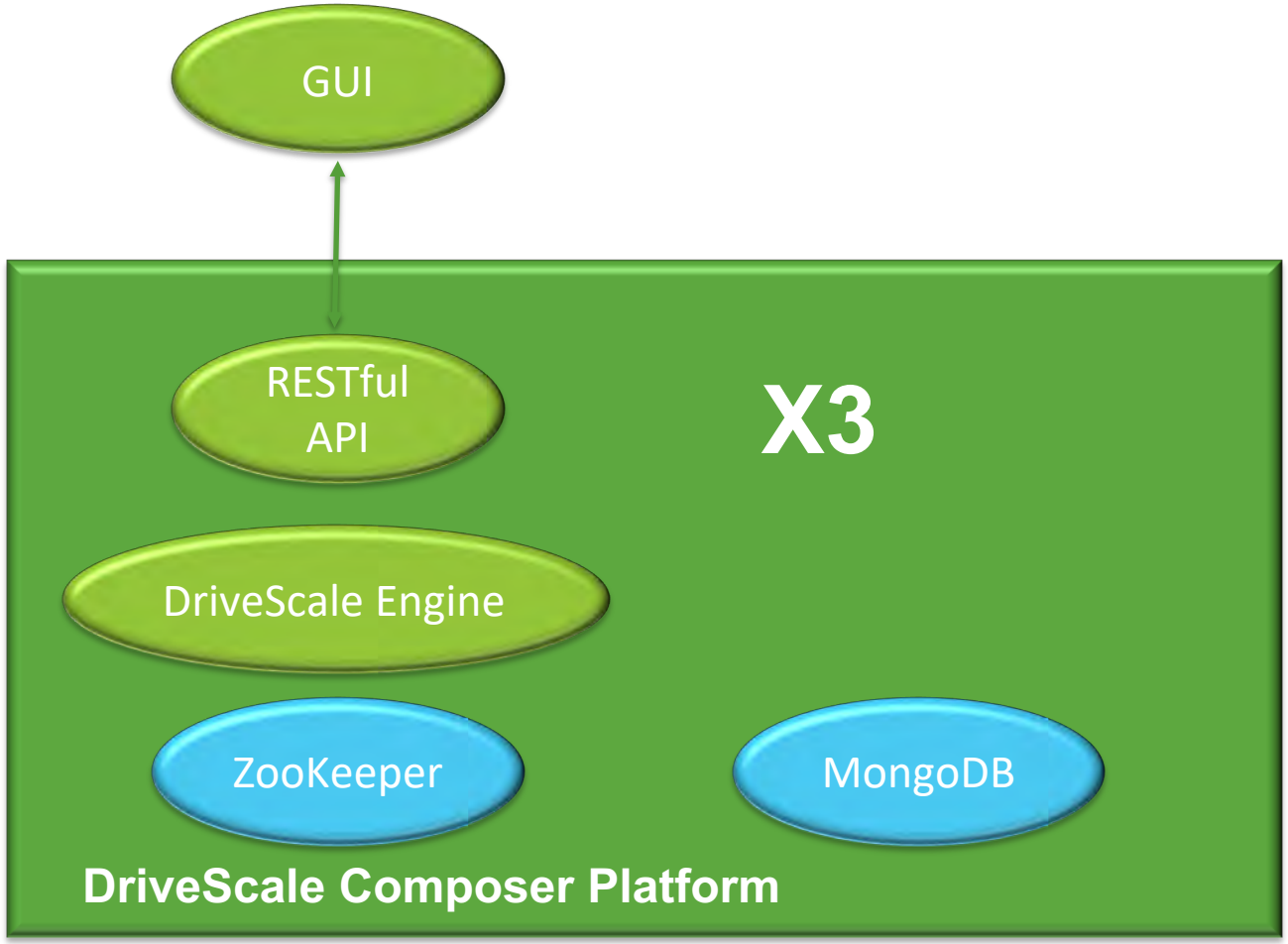


DriveScale Composer platform functions



- Provides durable bindings of servers to drives (HDD, SSD, SSD slices)
- Stores the auto-discovered inventory from disaggregated server and storage adapters
- Create, expand, manage and report on Logical Clusters
- Health monitoring and alarm notification
  - Notification management of all agents
- Template driven (includes user defined)
  - Templates as expression of each app reference architecture
- Manage role-based administrators
  - Read-only/modify/administer users/groups
  - Fine-grained, design for customer multitenancy (future)

DriveScale Composer platform structure

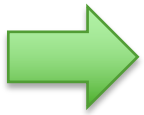


- DriveScale GUI uses same RESTful APIs as provided for 3rd party integration (see backup)
- Uses open source components
  - Apache Zookeeper is the durable configuration store
  - MongoDB stores auditing (daily push) and monitor logs (hourly push)
- DriveScale Engine implements control logic
- Triple replicas
  - Separate fault domains outside disaggregated pools
  - Provides increased read bandwidth
- Configuration snapshots are pushed daily to DriveScale Cloud Central for disaster recovery
- Monitoring and alert logs are pushed hourly to DriveScale Cloud Central (predictive use)

DriveScale Start Up

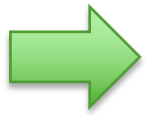
Installation

- Install DriveScale Composer software on triple replica Linux hosts (can be VMs)
- Install DriveScale agents on servers and storage adapters (/etc/drivescale/conf)



Discovery

- DriveScale agents automatically inventory the disaggregated nodes
- Initialize the DriveScale Composer with inventory of available compute and storage



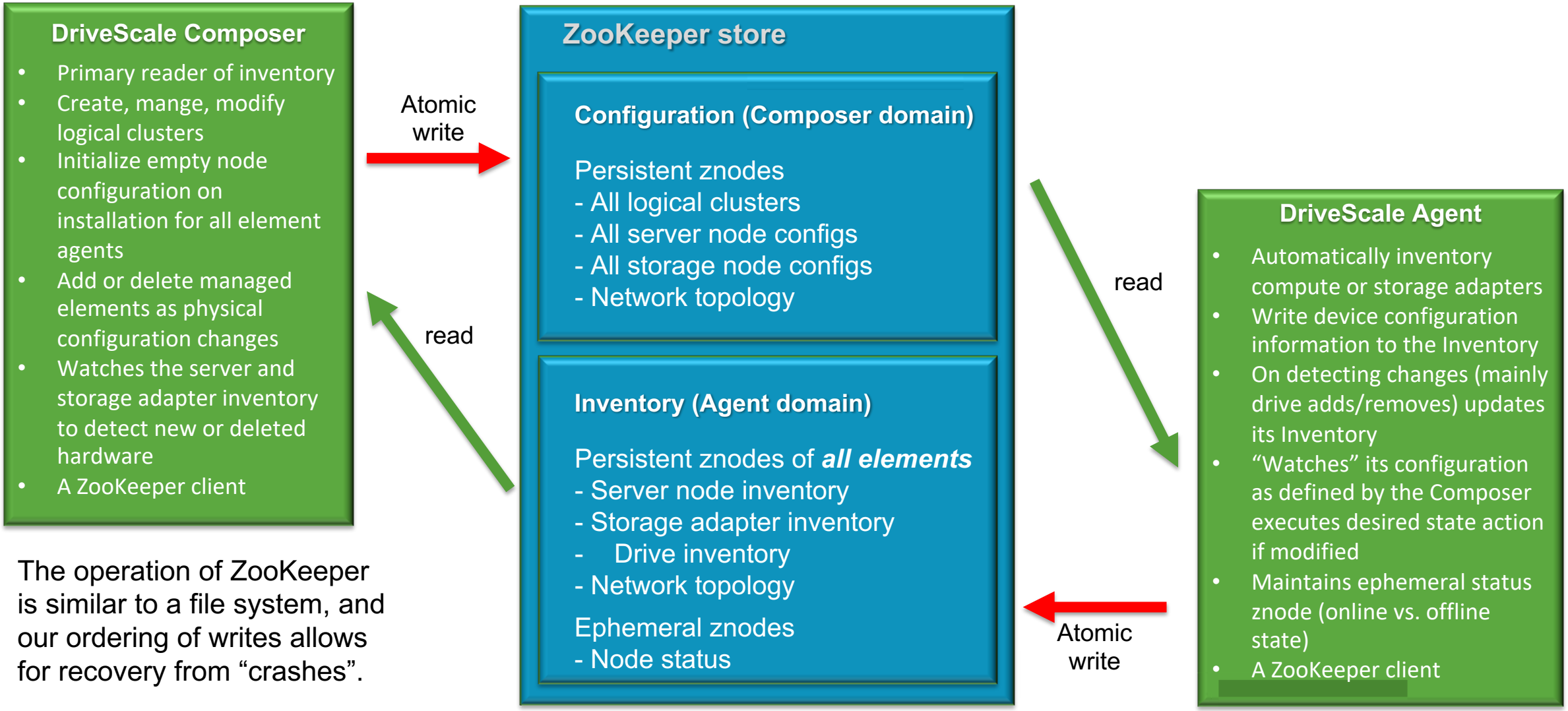
Operation

- User creates an *application (logical) cluster* from available servers and storage elements using the GUI or programmatically
- User monitors, modifies, expands, destroys clusters

DriveScale platform structure

- The early choice of the ZooKeeper framework reinforced and influenced the product design
  - Fully enabled the “current to desired state” execution model
  - (needed) Alert mechanism came along for free
  - Data structures separated in schema to single writers (owners)/multiple readers
  - We think in terms of *consistency points* when updating configurations
- User level resource agents and Composer
  - The DriveScale Composer manager contains the DriveScale Engine, ZooKeeper and Mongo DB instances – all deployed as 3-way replicas
  - Server and storage adapter agents are symmetrical (90% of code is common)
  - *Agents seek to achieve the **desired state** defined by the Composer in ZooKeeper*
- The *DriveScale Engine recovers incomplete server and storage adapter configuration updates after crash*
  - End point configuration writes are replayed to ZooKeeper complete transactions before starting service

DriveScale Data Design



The operation of ZooKeeper is similar to a file system, and our ordering of writes allows for recovery from "crashes".

DriveScale Inventory – agent write domain (all elements)

Server Inventory
- UUID
- network: addresses, speeds, LLDP
- supported transports (iSCSI, RoCE, NVMEoTCP)
- system memory
- # cores
- bogomips per core
- OS

Storage adapter inventory
- UUID
- network: addresses, speeds, LLDP
- supported transports

Drive Inventory
- WWN or NQN (persistent)
- size
- SSD or spinning RPM
- vendor/model/revision/serial
- ID of enclosing JBOD

Persistent znodes

Adapter Status
- status existence = online
- goal config version #
- latest achieved config version #
- status of goal config (accomplished, trying, retrying)
- errors trying goal config

Server Status
- same as adapter, plus
- is it safe to remove drives?

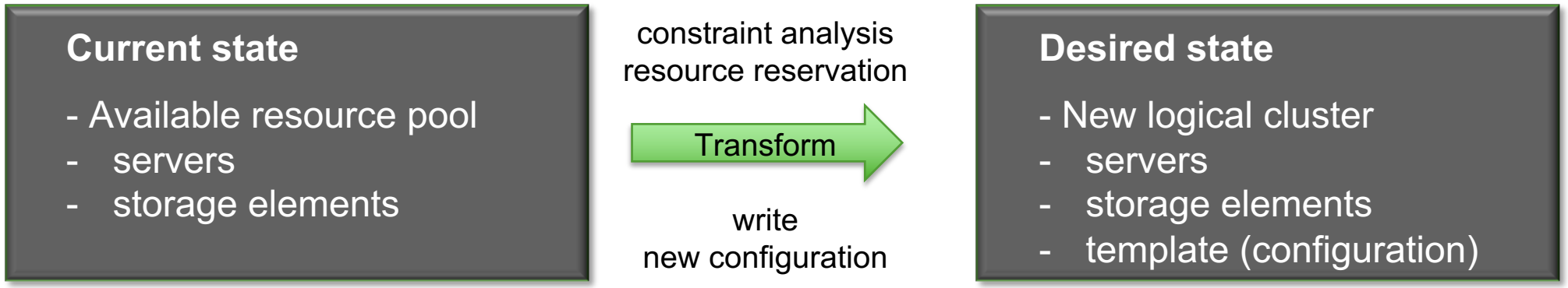
Ephemeral znodes

Ephemeral znodes deleted on heartbeat failure, indicating to the DriveScale Engine that the element has gone offline.

The above data is what we use in composition. Below-the-line data for both servers and adapters includes:
- HW vendor/model/serial
- installed version of our software
- complete SAS topology, etc.

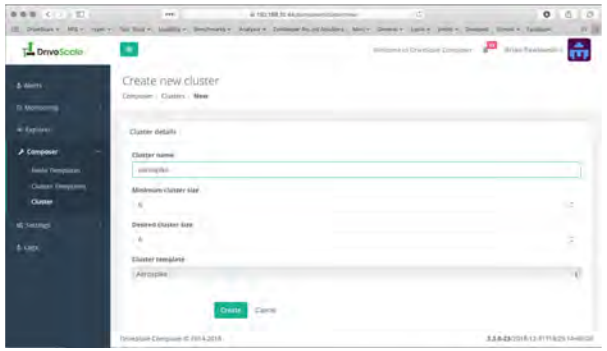
```
/DriveScale/server/config/*
/DriveScale/dengine/config/*
```

States – create logical cluster example



- Multiple distinct logical clusters co-exist in a DriveScale data center deployment
  - alongside global available (free) servers and storage elements
- ***Drives,*** not storage adapters, are composed with ***servers***
- DriveScale Composer configurations move from *current state* to *desired state*
  - *Only affected elements are acted on by respective agents, unused inventory unaffected*
- Creating or modifying a logical cluster configuration is non-disruptive to *unchanged* elements
  - Elements are added or deleted to a logical cluster as defined by the *desired state*
- *The DriveScale Composer defines the "what", as a desired state, not the "how", a set of commands. The DriveScale agents determine the steps to achieve the new configuration*

DriveScale Logical Cluster etc. – Composer write domain



new logical cluster configuration

DriveScale Engine

lcluster atomic write

/DriveScale/lcluster/config/*

Logical cluster configuration

Servers
- Targets (storage elements)
- Transports

Drives
- Addresses (network)
- Drive template

Template - policies
- File system
- Media type
- Encryption
- mount point, Transports

Server/Drive association/linkage

Version (config)

The Truth

Server configuration
- File system
- Drives
- Encryption
- mount prefix, etc
- Version

Storage adapter configuration
- Drives
- Server endpoint
- (not visible in lcluster, but necessary to push config)
- Version

The Derived Truth
restartable
recoverable

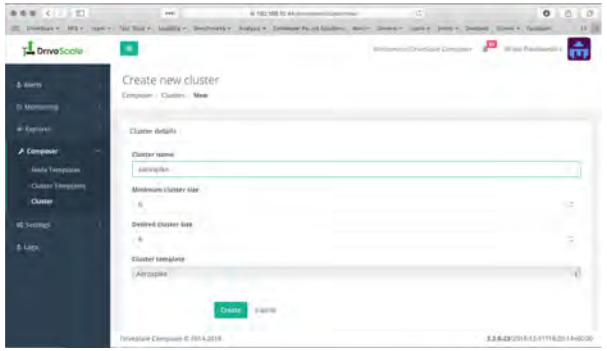
DriveScale Logical Cluster (JSON export)

```
root@u39:/opt/drivescale/tools/zookeeper# ./zk_cat.py /DriveScale/lcluster/config/75119458-dfb0-4a65-a994-7a66c74531e8
{
    "annotationsForLogicalNode": {
        "raid-encrypt-xfs-4-00-0000": {
            "ds:raidId": "0OqFrgzQIs"
        }
    },
    "driveShredMethod": "",
    "enonce": "JAv6J5ZdCqWy9usEApiinDU0z2n-n2muCX9j2YQw1O6PYte4h4osrgzAQKGLPDH3vViwKy3ohSfFx8HU5-O9MTo9Raqdjdu9Ek3dSP0JKEsnILktvrJU3wx5p-KOLjXPQ",
    "logicalNodeForDrive": {
        "iqn.2013-04.com.drivescale:wwn:0x5000c50058d47e87": "raid-encrypt-xfs-4-00-0000",
        "iqn.2013-04.com.drivescale:wwn:0x5000c50058d47ebb": "raid-encrypt-xfs-4-00-0000",
        "iqn.2013-04.com.drivescale:wwn:0x5000c50058d47edb": "raid-encrypt-xfs-4-00-0000",
        "iqn.2013-04.com.drivescale:wwn:0x5000c50058d47f3f": "raid-encrypt-xfs-4-00-0000"
    },
    "logicalNodeForServer": {
        "4c4c4544-0039-3010-8030-c4c04f313437": "raid-encrypt-xfs-4-00-0000"
    },
    "name": "jgh-raid",
    "options": [],
    "templateForLogicalNode": {
        "raid-encrypt-xfs-4-00-0000": "c207f56c-042f-4723-8fa5-47c885ce5d86"
    },
    "templates": {
        "c207f56c-042f-4723-8fa5-47c885ce5d86": {
            "driveTagsExcluded": [],
            "driveTagsRequired": [],
            "drives": 4,
            "encrypt": false,
            "fsType": "",
            "maxInstances": 100000,
            "mediaTransportsAllowed": [],
            "minDriveRpm": 0,
            "minDriveSize": 0,
            "minInstances": 0,
            "minServerBogoMips": 0,
            "minServerCpus": 0,
            "minServerMem": 0,
            "mountPrefix": "",
            "name": "raid-encrypt-xfs-4",
            "networkTransportsAllowed": [],
            "raidConf": "level=10,layout=n2,chunk=64,assumeclean=true,fstype=xfs,encrypt=true",
            "ratio": 1,
            "rotating": true,
            "serverTagsExcluded": [],
            "serverTagsRequired": [],
            "uuid": "c207f56c-042f-4723-8fa5-47c885ce5d86"
        }
    },
    "version": 51
}
root@u39:/opt/drivescale/tools/zookeeper#
```

DriveScale Composer in operation

- An administrator creates a *logical cluster* from available server and drive elements
  - Rule checked by the Composer and a configuration is created and atomically written to Composer's Zookeeper key-value domain
- The configuration for each compute server and storage adapter is derived from the cluster configuration and ***a new element state*** is atomically written to each element Zookeeper instance
  - Small optimization, storage configuration pushed first as servers tend to aggressively retry – the storage must reach the new state for compute to bind

Achieving *desired state* – storage adapter agents



Atomic Write

new configuration

ZooKeeper store

Configuration (Composer domain)

Persistent znodes
- All logical clusters
- All server node configs
- All storage node configs
- Network topology

Inventory (Agent domain)

Persistent znodes of ***all elements***
- Server node inventory
- Storage adapter inventory
- Drive inventory
- Network topology

Ephemeral znodes
- Node status

watch

read

new Configuration

the “what”

Storage Adapter Agents

Agents attempt to achieve new desired state – the ”how”

Retry forever

. . .

All agents are independent and execute in parallel

DriveScale storage adapter move to desired state (storage adapter logic)

A. What targets am I offering?

```
currentlyOffering = set(activeTargets().get('targets', []))
```

B. Is there anything I'm offering that isn't in my inventory?

```
good = devlinks.validTargetIqns(includeSlices=True)
bad = currentlyOffering - good
```

C. Is there anything I'm offering that isn't in my target configuration?

```
toOffer = set(newConfig.get('targets', []))
withdraw = currentlyOffering - toOffer
```

D. Stop offering everything from B and C:

```
withdraw.update(bad)
failures = stopTargets(withdraw)
```

E. Is there anything I'm not offering that is in my target configuration and inventory? Offer it.

```
toStart = set()
for targetName, config in newConfig['targets'].iteritems():
    # SNIP QUARANTINE LOGIC
    if targetName not in good:
        noDev += 1
        failures.add(targetName)
        continue
    changed = updateTargetConfig(targetName, config, queueDepth)
    # None if failure, true if updated, false if no change
    if changed is None:
        failures.add(targetName)
    elif changed or targetName not in currentlyOffering:
        toStart.add(targetName)

failures.update(startTargets(newConfig['targets'], toStart))
```

Performance – Aerospike ACT 4 benchmark

Flash Device	*Speed (tps)*	*>1ms*	*>8ms*	*>64ms*	*Endurance*	*ACT*	*Source*
SmartIOPS DataEngine 3.2T *	555,000 *	1.8%	0.01%	0.00%	3 DWPD	3	Aerospike
***Micron 9200 Max 1.6 TB**	**316.500**	**4.64%**	**0.13%**	**0.00%**	**3 DWPD**	**3**	**Aerospike**
HGST UltraStar SN200 RoCEv2	**300,000**	**2.60%**	**0.06%**	**0.00%**	**3 DWPD**	**4**	**DriveScale**
HGST UltraStar SN200 iSCSI	**300,000**	**4.14%**	**0.06%**	**0.00%**	**3 DWPD**	**4**	**DriveScale**
Intel P4610 1.6 TB	300,000	4.56%	0.85%	0.00%	3 DWPD	4	Aerospike
Intel P4610 6.4 TB	300,000	4.98%	0.14%	0.00%	3 DWPD	4	Aerospike
ScaleFlux CSS1000 3.2T	300,000	4.59%	0.00%	0.00%	5 DWPD	4	ScaleFlux
Micron 9200 Pro 3.84 TB	283,500	4.54%	0.13%	0.00%	1 DWPD	3	Micron
Huawei ES3600P V3 1.6T	153,000	4.47%	0.00%	0.00%	3 DWPD	4	Aerospike
Toshiba PX04PMB320 3.2T	135,500	4.73%	0.00%	0.00%	10 DWPD	3	Toshiba
Intel P4510 1 TB	120,000	4.31%	0.00%	0.00%	1 DWPD	4	Aerospike
Samsung PM983 1.92 TB	108,000	0.09%	0.00%	0.00%	1.3 DWPD	3.1	Aerospike

DriveScale

Wrap

Future directions

- We understand the network topology, and constrain cluster to avoid obvious configurations that will violate application replica fault separation – really want to do more here
- Additional capabilities on predictive analytics
  - Exploit long term monitoring to provide trending and failure predictions
  - End-to-end data monitoring provide rich data source
- ZooKeeper is not really a database – may restructure non-configuration information out of its data store for more powerful query capabilities
- Composing disaggregated GPUs
- Container framework integration – building on current Kubernetes plug-in
- Application centric analytics

Recapitulation

- Manage and automate complexity of composable infrastructure at scale
  - Including fault domains for rack failure semantic support
  - Distributed key distribution for authentication and date encryption over-the-wire and at rest
- Laser focus on scalability, durability, correctness goals
- Provide a simple, intuitive user experience

DriveScale

Questions

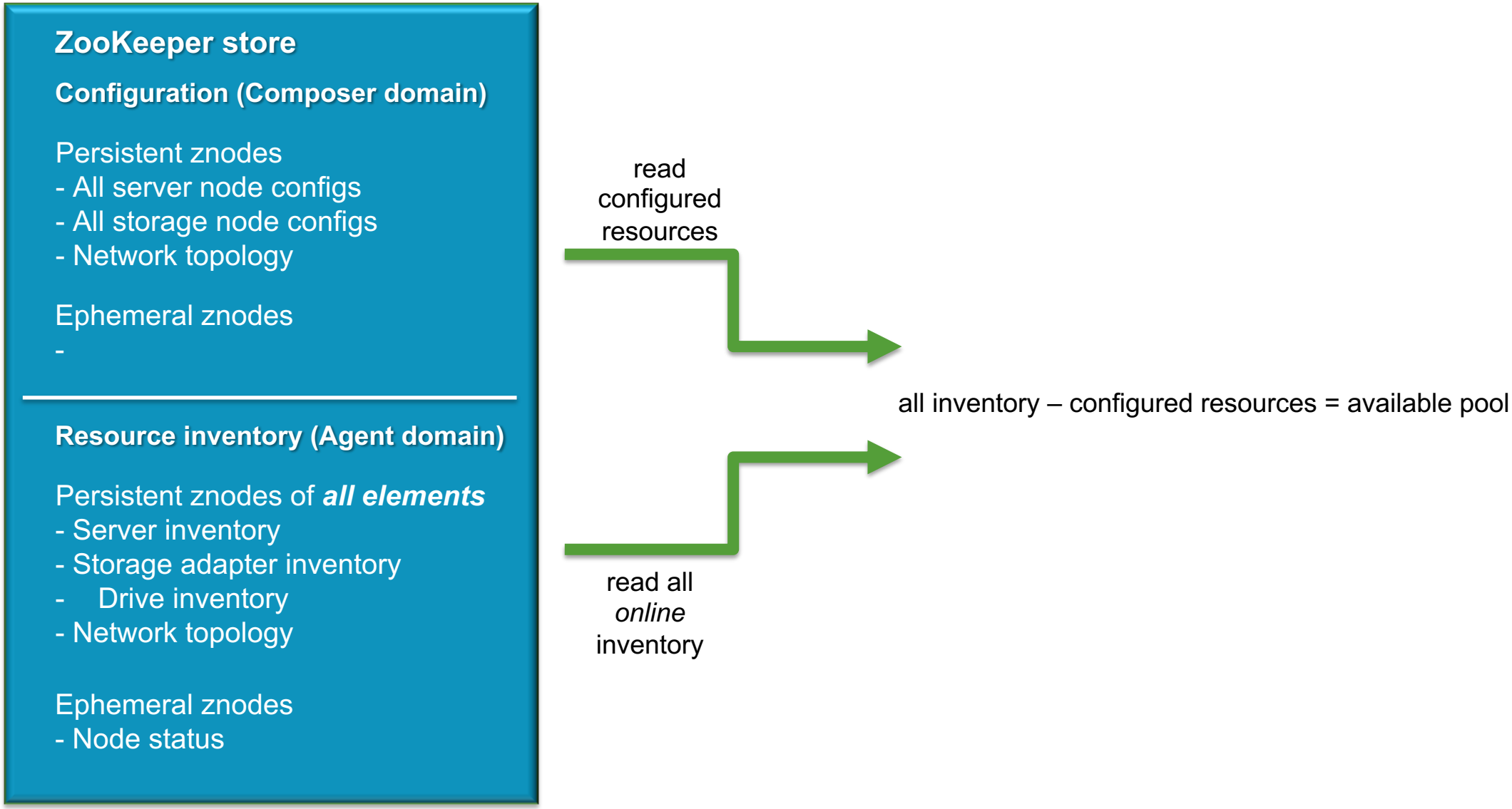
DriveScale

Backup

References

- ZooKeeper watches https://zookeeper.apache.org/doc/r3.1.2/zookeeperProgrammers.html#ch_zkWatches
- ZooKeeper design https://zookeeper.apache.org/doc/r3.1.2/zookeeperProgrammers.html
- ZooKeeper overview https://zookeeper.apache.org
- DriveScale Drive Encryption Overview - http://go.drivescale.com/rs/451-ESR-800/images/DriveScale%20Drive%20Encryption-WP-final-v2.pdf
- ZooKeeper use at Twitter https://blog.twitter.com/engineering/en_us/topics/infrastructure/2018/zookeeper-at-twitter.html
- Aerospike benchmark information https://www.aerospike.com/docs/operations/plan/ssd/ssd_certification.html

Determining the available resource pool



DriveScale

"**Carol Danvers (aka Captain Marvel) is very *durable***, able to survive attacks from the likes of the Hulk and has actually been able to lend her hand in physically fighting Thanos without getting one shotted."

DriveScale

ZooKeeper description

DriveScale

"**ZooKeeper** is used at Twitter as the source of truth for storing critical metadata. It serves as a coordination kernel to provide distributed coordination services, such as leader election and distributed locking."

Apache ZooKeeper: a durable data repository

- A widely used service for maintaining configuration information, naming, providing synchronization in a distributed system
  - Data is stored in a hierarchical key-value store (similar to a file system)
  - Multiple replicas provide availability and increased aggregate read performance
- ZooKeeper implements a *leader/follower model*
  - A single leader is elected on start up, the remaining nodes becomes followers
  - Only the leader can receives writes (guaranteeing atomicity)
  - Writes are propagated to followers before being acknowledged
  - All ZooKeeper instances can serve reads (design is suited for read mostly deployment)
  - Failure of a leader results in election of a new leader from the remaining followers (quorum on partition defines election group)
- The ZooKeeper data repository resides in memory during operation
  - Fits easily in 128GB server or VM
  - Initialized from persistent storage (SSD) on start up
  - Compressed data znodes (includes directories) are limited to maximum 1MB

DriveScale

Technology trends

Technology Trends – towards composable infrastructure

- Ubiquitous high performance, low latency fabric (Ethernet)
- Massively parallel Big Data and NoSQL etc. applications
  - Hadoop and other apps
- Affordable Solid State Storage enabling web scale real time applications
  - And increases in reliability of both HDD and SSD
- NVME everywhere

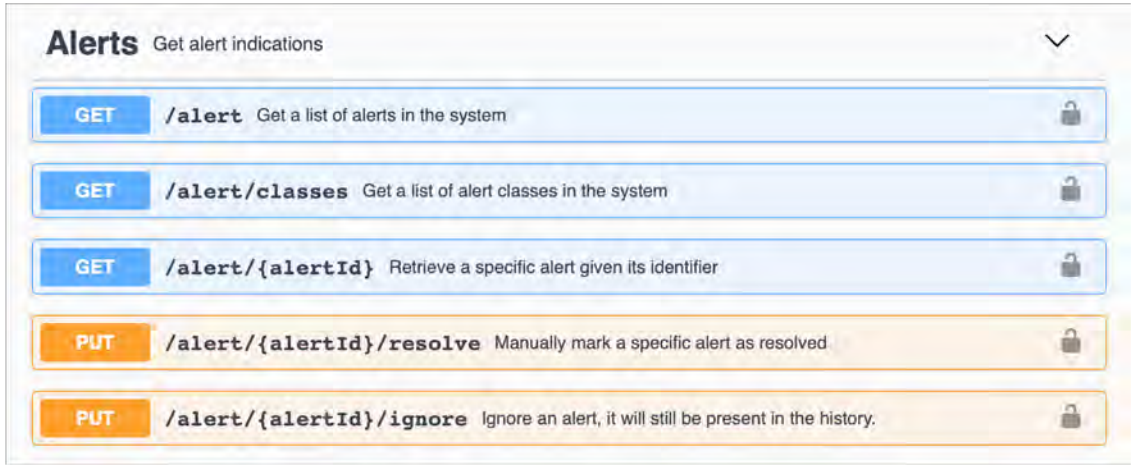
DriveScale's design exploits ubiquitous low-latency high-speed Ethernet at scale

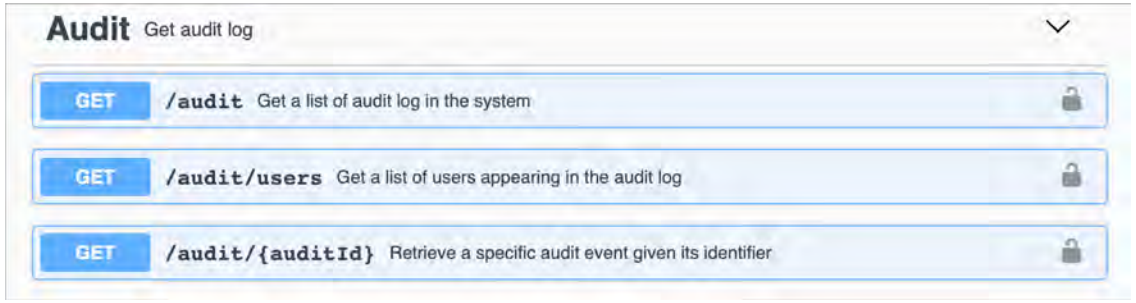
DriveScale

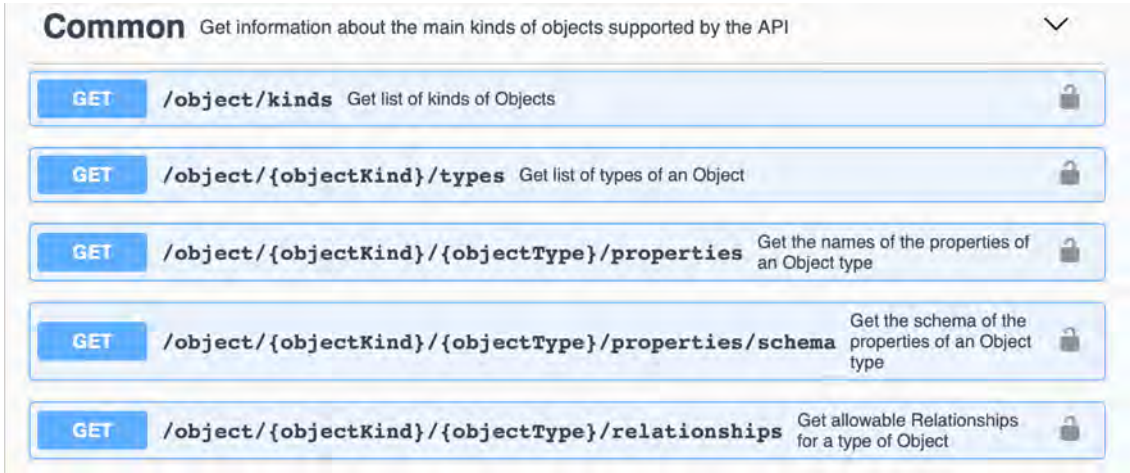
DriveScale Composable Infrastructure API Summary

REST API structure pt 1

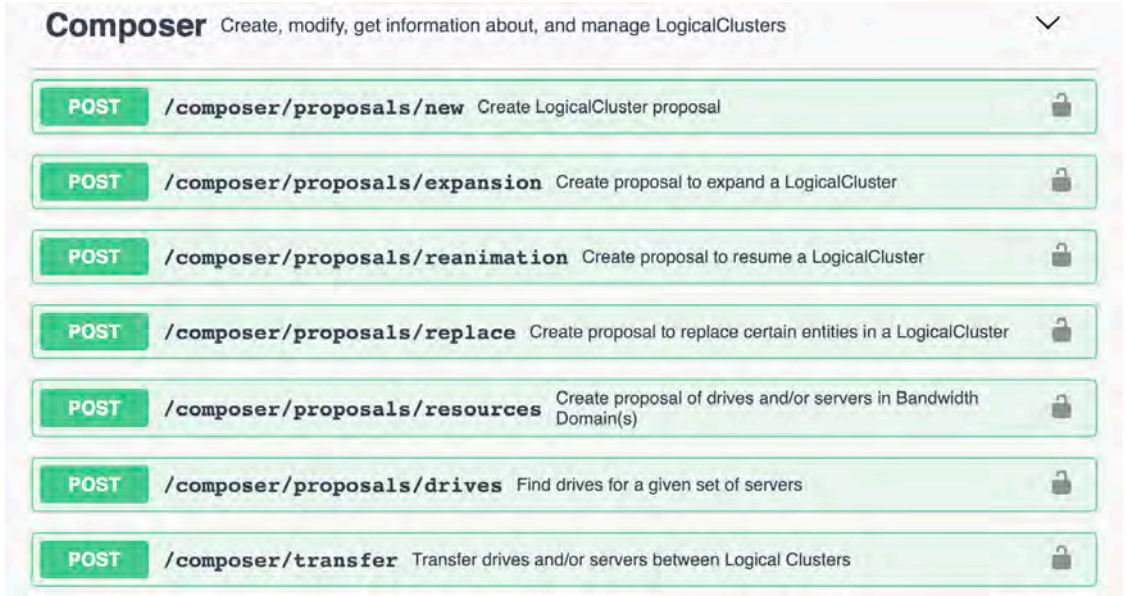








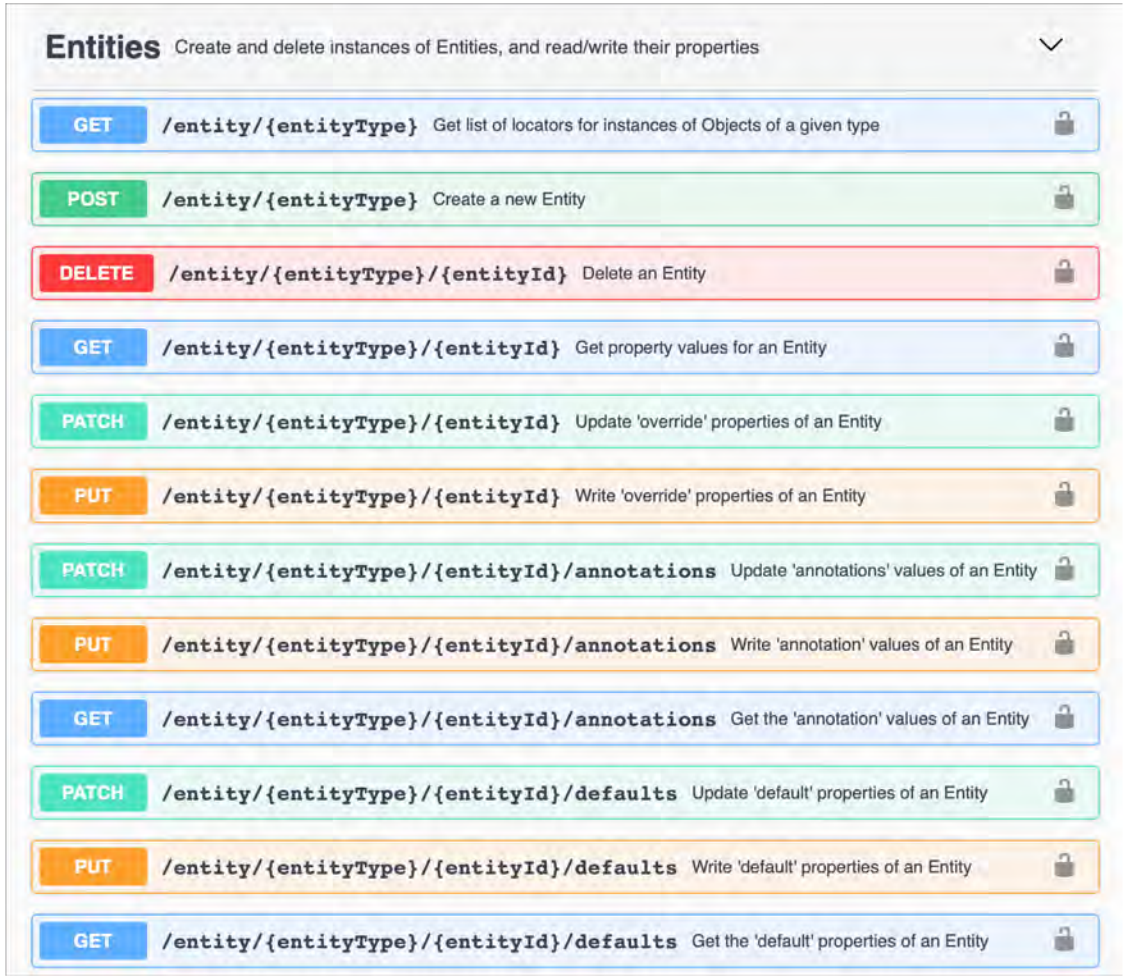
REST API structure pt 2







REST API structure pt 3



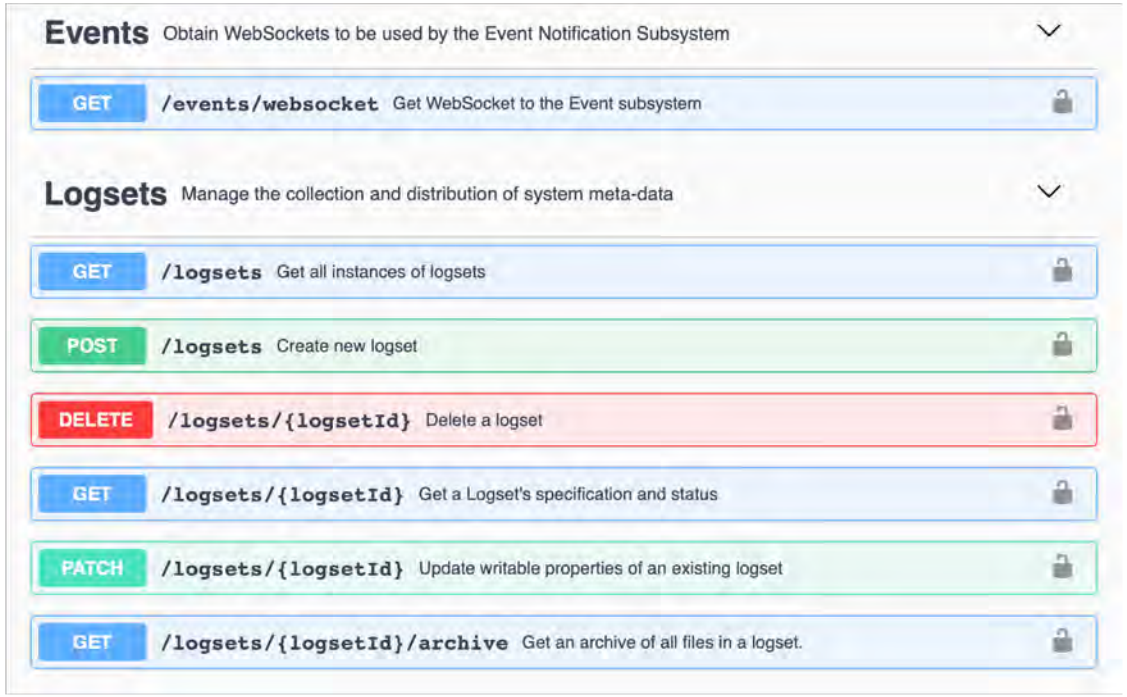
Entities Create and delete instances of Entities, and read/write their properties

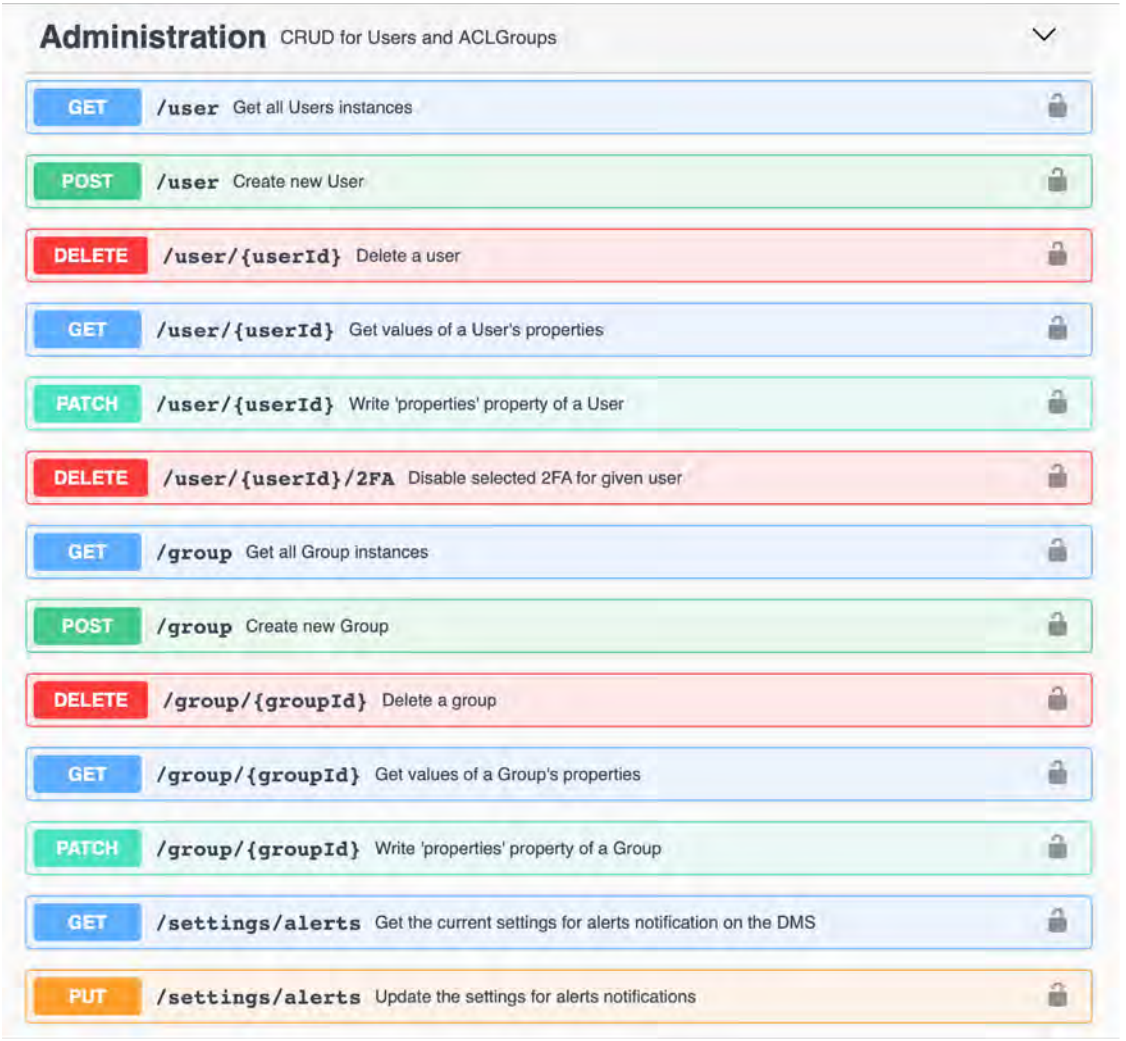
GET	/entity/{entityType}	Get list of locators for instances of Objects of a given type
POST	/entity/{entityType}	Create a new Entity
DELETE	/entity/{entityType}/{entityId}	Delete an Entity
GET	/entity/{entityType}/{entityId}	Get property values for an Entity
PATCH	/entity/{entityType}/{entityId}	Update 'override' properties of an Entity
PUT	/entity/{entityType}/{entityId}	Write 'override' properties of an Entity
PATCH	/entity/{entityType}/{entityId}/annotations	Update 'annotations' values of an Entity
PUT	/entity/{entityType}/{entityId}/annotations	Write 'annotation' values of an Entity
GET	/entity/{entityType}/{entityId}/annotations	Get the 'annotation' values of an Entity
PATCH	/entity/{entityType}/{entityId}/defaults	Update 'default' properties of an Entity
PUT	/entity/{entityType}/{entityId}/defaults	Write 'default' properties of an Entity
GET	/entity/{entityType}/{entityId}/defaults	Get the 'default' properties of an Entity



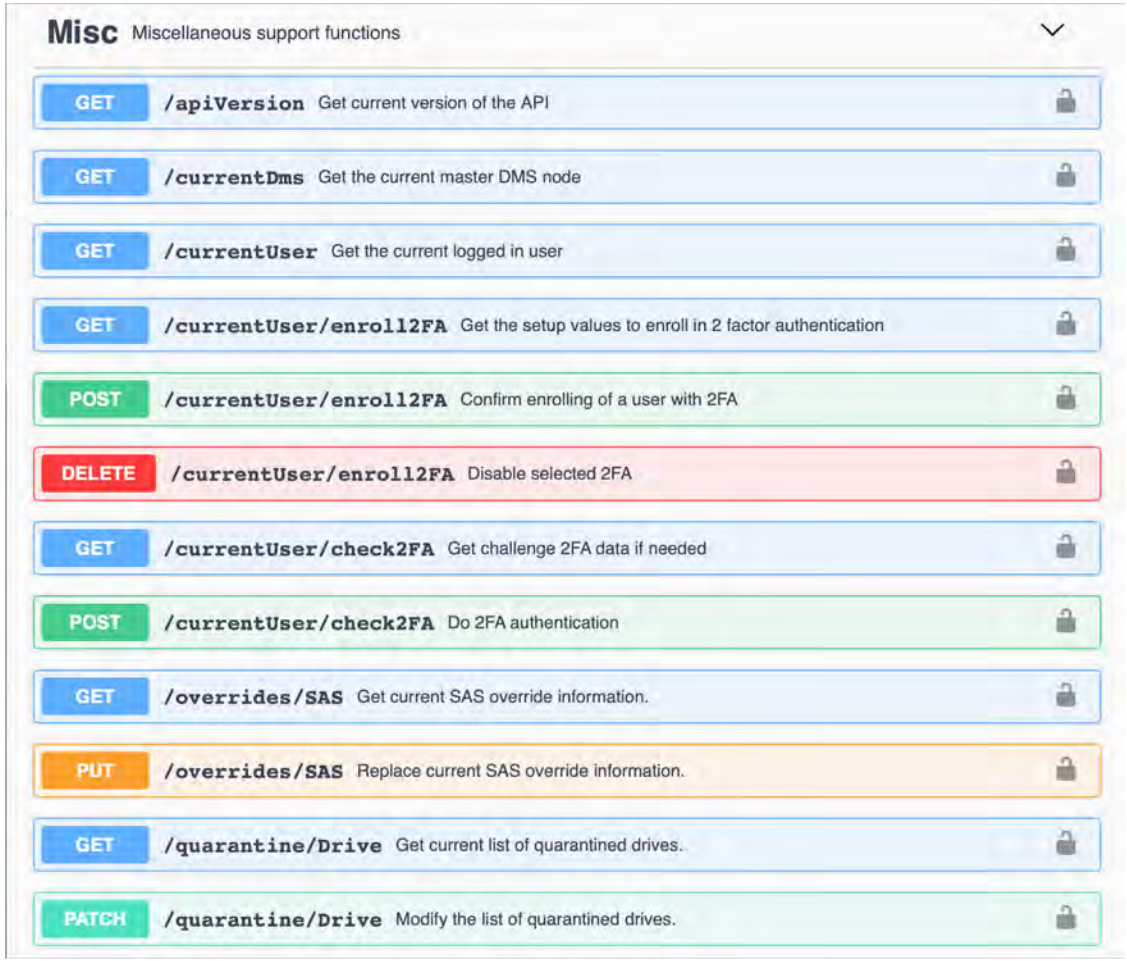
GET	/entity/{entityType}/{entityId}/graph	Get graph of an Entity's relatives
POST	/entity/{entityType}/{entityId}/unlock	Unlock ManagementDomain access for the entity
GET	/entity/LogicalCluster/{logicalClusterId}/config	Get the configuration properties of a LogicalCluster
PATCH	/entity/LogicalCluster/{logicalClusterId}/config	Update configuration properties of a LogicalCluster
PUT	/entity/LogicalCluster/{logicalClusterId}/config	Configure a LogicalCluster
POST	/entity/LogicalCluster/{logicalClusterId}/reconnect	Reconfigure a LogicalCluster
POST	/entity/LogicalCluster/{logicalClusterId}/reconnectAll	Reconfigure a LogicalCluster completely
POST	/entity/Drive/{entityId}/enableCarving	Enable a free SSD Drive to be carved into Drive-Slices
POST	/entity/Drive/{entityId}/disableCarving	Stop an SSD Drive from being carved into Drive-Slices
POST	/entity/Drive/{entityId}/shred	Request shredding of a free drive
POST	/entity/Drive/{entityId}/abortShredding	Abort shredding of a drive

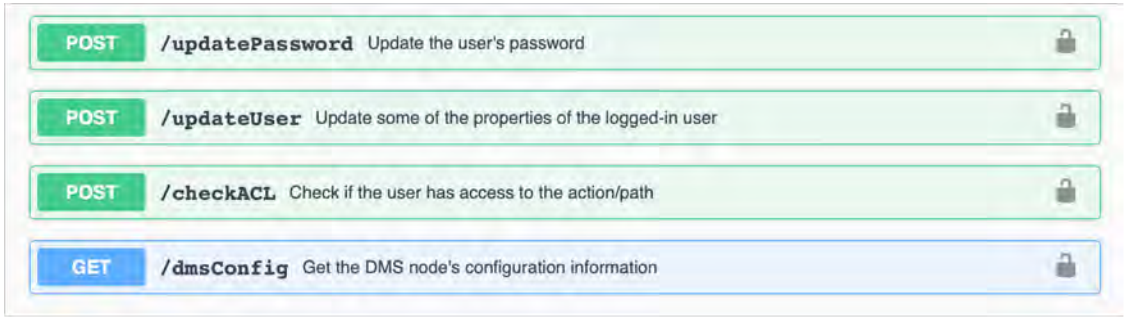
REST API structure pt 4





REST API structure pt 5





DriveScale Composer Platform (control and data plane)

