TECHNICAL STRATEGY DECISION PAPER

Area	Worker Node Hosting Model, Service Upgrades and Data Store
Triad	<director level triad name>
Decision Owners	
PM team	
SWE team	
Review Date	<date of director review and/or pre-review>
Document Status	Draft

<See https://osgwiki.com/wiki/Technical_Strategy_Decision_(TSD)_Process for a discussion of the TSD review process, and good examples of TSD document content. Use this template for authoring a TSD, and delete guidance text such as this sentence.>

QUESTIONS THIS DOCUMENT SHOULD ANSWER

The key questions this document aims to answer are as follows.

1. What will be the hosting model for services running on ASZ worker nodes?
2. What will be the key-value data store for services running on ASZ worker nodes to replace ReliableDictionary usage?
3. How will services running on ASZ worker nodes be upgraded?

AREA DESCRIPTION

Below are services that need to run on ASZ worker stamp.

Engineer Contact	Service	Note
Arindam Chakraborty <Arindam.Chakraborty@microsoft.com>	NC	Currently carried over NC infra VM from Hub to run with Windows built-in version of service fabric.
Benjamin Davidson <Benjamin.Davidson@microsoft.com>	Health service	To aggregate fault info etc. data from each node. There are per-node and cluster-wide use-cases.
Madhan Raj Mookandy Madhanm@microsoft.com	MOC cloud service	To work with node agent on each node. Using failover cluster registry as the data store solution for now. MOC Stack provides an Azure like API Surface to manage Virtual Machines, Network, Storage and Security constructs.
Raghu Ram Vemula <rvemula@microsoft.com>	Update service	To discover applicable updates etc.
Rahim Maknojia <Rahim.Maknojia@microsoft.com>	ECE service PMC service	To invoke action plan post deployment e.g., PnU etc. Physical machine controller which uses BMC protocol/interface to talk the power/BIOS management controller of the node
Ted Roberts <tedr@microsoft.com>	JIT/RBAC service	Query and approve user role request for just in time and role-based access control.
Yuan Zhang <yuanzh@microsoft.com>	Download service	Download OS, infra service etc. update for PnU which can be big size as 10 GB+ and want to be able to resume with fail over.
Tony Ni <tonyni@microsoft.com>	IBC Service	Backup/Restore Infrastructure meta data

Such services shall use the same hosting model and data store to save servicing costs.

Below services may be handled separately:

1. Services hosting at an infrastructure VM and not at the host, e.g., NC.
2. Services which need to store secrets, e.g., ECE which will have a dedicated data store solution.
3. Services already had their own integration with Failover Cluster, e.g., MOC.
4. Services which have append-only high-availability workloads, e.g. Health's time-series database.

Below are the requirements from services which will impact the decision of hosting model, service lifecycle support and data store. These requirements reflect upper bound assumptions that satisfy existing problem boundaries, not forward-looking ones.

	NC	MOC	Health	ECE	PMC	Update	Download	JIT/RBAC	IBC
Minimum availability (cold, warm, hot redundancy) Cold: transient unavailability during failover and update. Warm: transient unavailability when clients must connect	warm	cold	cold	cold	cold	cold	cold	cold	cold

to new servers, e.g. 1 primary and 1 standby instance. Hot: continuously available, multiple active instances.									
Outage downtime before SLA breach or data loss	5 min		2 hours	5 min	2 hours	5 min	1 hour	5 min	4 hours
Has shared state requires HA (excludes secret)	Yes		Yes	Yes	No	Yes	Yes	Yes	Yes
Can your schemas change or introduce new data types? (Service will perform data migrations *0)	Yes		Yes	Yes	N/A	Yes	Yes	Yes	Yes
Has shared secrets require HA	Yes		No	Yes	Yes	No	No	No	Yes
Has secrets shared between services	Yes		No	Yes	Yes	No	No	No	Yes
Has state to backup/restore at BCDR	Yes		No	Yes	No	No	No	Yes	Yes
Has secret to backup/restore at BCDR	No		No	Yes	Yes	No	No	No	No
Data can't lose during data migration	No		Health data for addon RPs	No concern	No	No	No	Request and usage data to audit	No
Health monitor by liveness probes (update rollback determination, failover, reboot etc.)	No		Yes	Yes	Yes	Yes	Yes	Yes	Yes
Requires internal endpoint (use within the infra only) *1	Yes		Yes	Yes	Yes	Yes	Yes	Yes	Yes
Expose external endpoint to outside of the cluster	Yes		No	No	No	Yes?	No	No	No
Scale out across nodes, partitions, and/or shards	Yes		No	No	No	No	No	No	No
Data size	10MB		200MB	100MB	0	100MB	10MB	200MB	100MB
Latency sensitive (Synchronous APIs which support the critical user experience path)	Yes?	No?	No	No?	No?	No?	No	No?	No?
Write rate (calls per second)			10 TPS	10 TPS	N/A	5 TPS	1 TPS	1 TPS	2 TPS
Read rate (calls per second)			10 TPS	10 TPS	N/A	10 TPS	3 TPS	3 TPS	3 TPS
Write request rate (request size)	5 KB/s		30 KB/s	30 KB/s	N/A	10 KB/s	5 KB/s	1 KB/s	1 KB/s

Read request rate (response size)	5 KB/s		30 KB/s		N/A	10 KB/s	5 KB/s	Export	1 KB/s
Minimum data ordering requirements? Some examples are given below. Sequential ordering implies that there is a total ordering. Read your own write ordering implies that clients can consistently read data after they've written it. Monotonic read and write ordering implies that a client can never read or write further in the past than a previous read or write operation, respectively.	Read your own writes		Read your own writes	Read your own writes	N/A	Read your own writes	Read your own writes	Read your own writes	Read your own writes
Require multikey transactions? (serializability)	Yes		No	No	No	Yes	No	No	
Require ad hoc analytical reads? I.e., do you have query requests that are compute and data intensive?	No		No	No	No	No	No	No	No

*0 Schema updates and introducing new stateful entities are not a well-supported use-case in Hub.

*1 If the workloads behind internal endpoints are coordinated by implementations of consensus algorithms, then the infrastructure footprint of many software architectures can be reduced. This pattern is called a consistent core. For example, assume you have an existing database that implements replication but does not provide coordination. Using the consistent core pattern, you could coordinate a replication scheme which would provide high availability for data. This is considerable time and effort compared to off-the-shelf distributed databases but since coordination is provided by infrastructure no individual task is insurmountable.

The common needs of the services include creation/update lifecycle support. This includes metadata that identifies or is specific to the environment including DNS prefixes, region, and identifiers for the cluster and machine.

The common needs of the services include preventing quota-based oversubscription of shared resources such as compute capacity, system memory, disk utilization and so on. Services which are infrastructure for other services could be modeled as another shared resource, providing inter-team quota management and signaling changes in demand. This is commonly referred to as limiting.

The common needs of the services include preventing rate-based oversubscription of shared resources. This includes preventing clients from performing too many requests per second against their resources. This is commonly referred to as throttling and can be solved in a local or distributed way.

DECISION SUMMARY

The hosting model decision was made previously to use Failover Cluster + ALM (Agent Lifecycle Manager) + System Agents. ALM is currently used to deploy HA services as Failover Cluster generic services. Other options were previously dismissed to save development costs and system footprint. E.g., Kubernetes is not applicable until all services migrate to use .NET core and be able to run on a Linux VM, Service Fabric was not preferred according to the goal to minimize system resource consumption specially to maintain a low hardware spec for the single node scenario.

However, we have confidence issues that built-in data store along with the failover cluster supports the requirements. Teams are exploring options between cluster DB, mainline ESENT DB on ReFS with integrity streams, ESENT with custom replication on local storage, commercial off the shelf time-series databases such as Prometheus and exposing internal datastores for new use-cases such as SDDC's time-series database.

ALM can deploy HA services but it is not cluster-wide orchestrator. ALM cannot handle all upgrade scenarios while minimizing failovers and maximizing availability. When all nodes up and HA service can be upgraded and started on all nodes ALM ensures that the update completes with an only a single failover. When one or more cluster nodes are down ALM will upgrade the service on all nodes that are up with a single failover. However, the overall update will fail at ECE action plan level since ECE waits for ALM success indicators for all nodes. ECE will not see the success indicators for nodes that are down. Once cluster nodes are brought back up and ECE's update action plan must be restarted so that ALM will update the service on nodes that were previously down.

Upgrade rollback presents some difficulties. Currently ALM cannot handle scenarios that require the HA service upgrade to be rolled back. An upgrade rollback is a cluster-wide operation that should be triggered from a single upgrade orchestration entity – while ALM is cluster-aware and can make decisions based on cluster resource as well as node state, it is not a cluster orchestrator.

If Failover Cluster remains the hosting model, options that either enhance ALM for better upgrade orchestration support or introduce that support via other components need to be explored\considered. At present, Failover Cluster does not have any built-in functionality that can be leveraged here. While CAU can allow for a rolling upgrade of HA services via CAU plug-ins, it does not have support for cluster-wide rollback based on upgrade failure on a single node or for a cluster-wide rollback if the cluster resource is in a failed state post-upgrade.

<Summarize what you're proposing we do in this area for Redstone.>

EVALUATION CRITERIA

The following is a hosting model comparison. It is not meant to be a generic comparison between FC (Failover Cluster) and SF (Service Fabric) but, rather, a comparison as it relates to service requirements outlined in the table above and other existing ASZ Worker scenarios. The comparison is meant to represent functionality that currently exists more-or-less "out of the box." It does not account for new features that could be implemented on top of either hosting model to satisfy requirements (e.g., CAU\FC enhancements for rollback, etc.)

	FC + ALM + System Agents	SF + System Agents
Active-Passive Cold Replica Support		
Meets outage downtime requirement during failover		
Shared\HA data store with support for random data access	Partial support as per the limitations with more details in the data store choice compare session	
Shared\HA data store with support for time-series data access	Can use a DB (e.g. ESENT) on top of shared storage provided by S2D	Reliable Queues do not provide complete time-series support.
Shared\HA secret store	A service-specific implementation of secret store exists in Common-Infra but no platform-level support in FC.	
BCDR support for shared data	Cluster registry can be backed up but no per-service BCDR support. Hub IBC can be leveraged, more details in the data store choice compare session.	
BCDR support for shared secrets	BCDR support can be built, more details in the data store choice compare session.	Requires some additional logic on top of SF
Health monitoring via liveness probes (update rollback determination, failover, reboot etc)	Existing heartbeat support is not sufficient for outside-in monitoring	
Support for cluster-wide service rollback in rainy day scenarios		
Internal Endpoint Support (endpoint configuration, discovery)		
External Endpoint Support (endpoint configuration, DNS)		
Cluster scale out 1 -> 2 -> 3+ nodes		Support for cluster witness is work-in-progress
Support for app provisioning (bits, etc) on scaled out nodes		
Resource Governance (Memory\CPU) for services		
Coordination between clustering technologies (resource consumption, rebalancing, etc)	Not needed	

The following evaluation criteria are applicable to the decision of data store to use together with Failover Cluster.

1. Does it full fill the services requirements?
2. Can it meet GA (general availability) timeline in 2022 Oct/Nov?

The following evaluation criteria are applicable to the decision on how highly available services will be upgraded on ASZ worker node.

1. Is there existing support for rolling upgrade that maximizes availability\minimizes failovers?
2. Support for health probes to be able to determine HA service health post-upgrade for cases where service failures do not manifest as Generic Service crashes\exits?
3. Can features be implemented to support rollback of arbitrary HA service?
4. Implementation complexity\cost

<List the evaluation criteria using the following guidelines:

1. *DO NOT use pros / cons. They sow seeds of great confusion and deep conflict.*
2. *DO list the candidate options from which a winner is to be chosen.*
3. *DO list the evaluation criteria that candidate options should satisfy, stating each criterion as a positive outcome that can be measured objectively.*
4. *DO eliminate redundant criteria. Redundant criteria are defined as criteria that the candidate options all satisfy or all fail to satisfy, i.e. they don't help tell us which option to pick.*
5. *DO include all distinct criteria. Distinct criteria are defined as the opposite of redundant criteria.*

DO rank the criteria in order of most critical to least critical. Do take as much time as required to discuss the ranking. Failure to agree on ranked evaluation criteria is the #1 reason that decision making processes fail.>

OPTIONS CONSIDERED

The following options were considered for the decision of data store to be used along with failover cluster.

- Option 1: Cluster DB.
- Option 2: ESENT DB over S2D.
- Option 3: Service Fabric Reliable Dictionary.

The following options were considered for determining how HA services will be upgraded\rolled back. This assumes that Failover Cluster remains the answer to the first question:

- Option 1: Implement a custom upgrade orchestration component that will be a part of (or work in conjunction with) ALM
- Option 2: Enhance CAU to support rollback based on upgrade failure or bad cluster resource state post-upgrade
- Option 3: Implement rollback via ECE action plans that set ALM's desired state to force a rollback (more specifically, force a downgrade to the previous good version)

<Summarize the different paths we could realistically take in this area and the implications of each. Be sure to cover desktop, WCOS, Xbox and HoloLens options. If applicable, also cover IoT and Azure Host OS.>

CHOSEN OPTION AND REASONING

The previous decision was already made to use **Failover Cluster + ALM (Agent Lifecycle Manager) + System Agents** as the hosting model.

Option TBD is the chosen option for second decision ("what will be the data store for highly available service running on ASZ worker stamp") based on the following table.

Evaluation Criteria	**Option 1 (Cluster DB)**	**Option 2 (ESENT DB over S2D)**	**Option 3 (Service Fabric Reliable Dictionary)**
Cold/Warm/Hot Redundancy	**Warm**	**Warm**	**Warm**
Support data size required by all services (750MB-1GB)	No	Yes	Yes
Has servicing support	Failover Cluster team support (Dan Upton)	ESENT team (Anil Ruia) shall support, pending confirmation	Service Fabric
Transaction support	Mitigated by using an in-memory cache	Yes	Yes
KVS (Key Value Store) and RC (Reliable Collections) support via common infra	Need to improve current implementation for (1 dev month before GA)	Need implementation, can leverage Health or Shanghai team's previous implementation (1 dev month before GA)	Yes
Secret storage	Mitigated by encrypted settings store library		n/a

Time-series data access	No	No	No
BCDR support for shared data and secret	Hub IBC already supports it and can be leveraged (cost TBD by Tony for ? Dev month before GA)	TBD by Tony (TBD by Tony for ? Dev month)	
Performance	TBD	TBD	
Tenant Isolation	Single instance available to the cluster.	Multiple instances available to the cluster	Built-in partitioning support.

Option TBD is the chosen option for third decision ("How will highly available services running on ASZ worker stamp be upgraded?") based on the following table.

Evaluation Criteria	**Option 1 (ALM + new cluster orchestration component)**	**Option 2 (CAU Enhancements)**	**Option 3 (ALM + rollback implemented via ECE)**
Existing support for rolling upgrade	Yes	Yes, via developing a custom CAU plugin to upgrade Generic Service resources	Yes, via ALM
Support for Health Probes	Already planned for non-HA agents but not implemented yet.	No	Not directly but could make use of ALM's reporting.
Can be enhanced to support rollback of arbitrary HA service	Yes	Possibly	Yes but cannot support rollback of ECE service itself

Implementation Complexity\Cost	Medium	Likely high\very high	Low
Support for resource isolation e.g. CPU/memory limit			
Topology (scale out from 1 node to multi-nodes)			

<Which path are you proposing we take and why? Use the following guidelines:

- *Do generate a two-dimensional matrix where one dimension (row / column) lists the candidate options and the other dimension (column / row) lists the evaluation criteria in order of rank.*
- *Do score the candidate options against the evaluation criteria.*
- *Do select whichever candidate option satisfies the longest contiguous run of evaluation criteria starting from the highest ranked criterion.*
- *If there is more than one winning candidate, it must mean that:*
  a. *We didn't completely / correctly list our evaluation criteria. I.e. there is some missing criterion that, if we'd included it, would have broken the tie between the multiple winners.*
- *If the winning candidate option is unacceptable in some way, it must mean that:*
  a. *We didn't completely / correctly list our evaluation criteria*
  b. *We didn't identify a complete / correct set of candidate options*
  c. *We didn't rank the evaluation criteria correctly, i.e. the ranking doesn't actually reflect what we really care about>*

REMAINING ISSUES AND QUESTIONS

1. S2D configuration (from Jessica Collins <jescollins@microsoft.com>)

S2D would be functionally correct with integrity streams enabled, which is not enabled in the current plan, but we've been told it can be enabled for individual folders, but the consequences aren't clear in terms of disk management or performance.

2. Issues of using ESENT store
    a. Azure Consistent Storage (ACS) blob/store/queue implementation took this approach where they persisted data in an ESENT store and leveraged S2D for replication. According to @Davie Chen, they ran into issues with this approach. (from Brendan Bache <bbache@microsoft.com>)
    Jessica Collins <jescollins@microsoft.com> followed up. @Davie Chen mentioned integrity streams, an SMB loopback issue for locally mounted volumes (pg 126 of the attached file), and changes that needed to be made to ESENT itself. There were concerns about the how integrity streams would handle large ESENT snapshots but in their testing this was not an issue in the hundreds of GB range. The small files were less of a concern.
    b. (from Jeremy Collette <Jeremy.Collette@microsoft.com>)
3. ALM feature asks (point of contact: Vlad Alexandrov <Vlad.Alexandrov@microsoft.com>)
    a. We need it to be FC-aware, failing over to only the latest version of code and not reverting to a previous version mid-deployment. (from Jessica Collins <jescollins@microsoft.com>)
    b. Single repo check in when bringing in a new version of service nuget together with ALM configuration changes. Currently would need to have additional check in to the ALM repo. (from Yuan Zhang <yuanzh@microsoft.com>)
4. Sharable code check in to common infra repo
    a. Least-common-denominator APIs in ALM or some common code (from Jessica Collins <jescollins@microsoft.com>)
    b. Data persist interface similar to KVS and RC to work with the data store solution, so the http service built on top of common infra shared code doesn't need to have additional changes (from Yuan Zhang <yuanzh@microsoft.com>)

<What are the big open issues in this area that we still need to resolve? These will be tracked and resolved over the course of the release. >

LINKS TO RELEVANT DOCUMENTATION FOR BACKGROUND

The following documents provide additional background information.

- Application Hosting Models TSD provides more details of different hosting models.
- ASZ Update problem statement for more details about the 3rd decision to update HA services running on ASZ-W with FC+ALM.
- Onboarding agents to ALM provides instructions to use ALM to deploy a service.
- Extensible Storage Engine Managed Reference - Win32 apps | Microsoft Docs
- Cluster-Aware Updating advanced options and updating run profiles | Microsoft Docs

- Observability Service TSD describes the model to use at section 7.2.3.2.1. It is cluster aware but not a clustered resource, to ensure independency from failover cluster while retaining ability to get information from it, thus it is not in the scope of this TSD.
- Transactions and lock modes in Azure Service Fabric Reliable Collections describes the ordering of transactions in Reliable Collections.

<Any additional documentation relevant to the area?>

LINKS TO RELEVANT WORK ITEMS

<Insert a link to a VSO query or a table of links to VSO deliverables for the decision>