



Reference Architecture 4.X

Last Updated: August 2020

Version: 2.1

Copyright © 2020 Morpheus Data, LLC. All Rights Reserved
All third-party product and company names are property of their
respective holders and use does not imply any specific endorsement

VERSION HISTORY

Description	Version	Date
Original Document	1.0	20 Nov 2019
Updates including AWS supported architecture	2.0	12 May 2020
Update to include 4.2.2 architecture changes, added options to disaster recovery section, minor updates and restructured the document for easier readability.	2.1	6 Aug 2020

TABLE OF CONTENTS

Version History.....	2
Table of Tables.....	5
Table of Figures.....	5
Deployment Requirements.....	6
Resources	6
Licenses	6
Repositories.....	7
Operating Systems.....	7
Networking Requirements	7
Connectivity.....	7
Name Resolution	8
Latency	8
Security.....	8
Authentication.....	8
Encryption	8
FIPS 140-2	8
Certificates.....	9
Compliance	9
Log Types	9
Morpheus Server Log	9
Audit Log.....	9
Agent Log.....	9
Telemetry	10

Languages	10
Components.....	10
Open Source	10
Morpheus Agent (Optional)	11
Capabilities	11
Supported Operating Systems.....	11
Morpheus Tiers	11
Application Tier.....	12
NGINX	12
Apache Tomcat.....	12
Apache Guacamole.....	12
Redis (Version 4.1.x or prior).....	12
High Availability	13
Non-Transactional Database Tier	13
High Availability	13
Messaging Tier	14
High Availability	14
Transactional Database Tier	17
High Availability	17
Morpheus Data Flow.....	18
Supported Morpheus Architectures	19
Single Server Architecture	19
Single Site Redundant Architectures	20
Redundant Combined Tiers (3-Node Architecture).....	21

Redundant Distributed Tiers (11-Node Architecture)	22
Multi-Site Architectures	23
AWS Multi-Availability Zone Architecture	23
<i>Disaster Recovery.....</i>	28
Multi-Site Active/Passive Architecture	28
<i>Appendix A: Morpheus Server Sizing</i>	32
<i>Appendix B: Ports and Protocols.....</i>	34

TABLE OF TABLES

Table 1. Minimum Server Count Per Architecture.....	19
Table 2. Morpheus Server Sizing	32
Table 3. Ports and Protocols	34

TABLE OF FIGURES

Figure 1. Message Queues.....	16
Figure 2. Data Flow Diagram.....	18
Figure 3. Single Server - Non-Redundant Combined Tiers Conceptual Diagram	20
Figure 4. Redundant Combined Tiers (3-Node Architecture) Conceptual Diagram.....	22
Figure 5. Redundant Distributed Tiers (11-Node Architecture) Conceptual Diagram	23
Figure 6. AWS Multi-Availability Zone Conceptual Diagram.....	27
Figure 7. Disaster Recovery Conceptual Diagram SAMPLES	31
Figure 8. Port Diagram.....	37

DEPLOYMENT REQUIREMENTS

Resources

Use the recommended resources details within the sections below to achieve acceptable performance for production environments and continue to monitor resource utilization on an ongoing basis increasing resources or scaling the deployment as necessary. Minimal resources are adequate for non-performance / non-production based testing.

Licenses

Community License (Morpheus Community Edition)

Available to all upon installing Morpheus 4.1.1 or greater. A Community License is a 12 months time-limited license that restricts the user to three integrated clouds and up to 25 workloads managed or discovered. Additionally, guidance (rightsizing) recommendations are read-only. This license is fully self-service and does not include any Morpheus support services. Community licenses may be requested from the Dashboard tab of Morpheus Hub (morpheushub.com) so long as the account does not currently have other licenses assigned.

PoC License

Typically limited to 30 days to allow Customers and Morpheus field teams to work together in testing suitability for specific automation use cases and environments.

Production License

Production licenses are good for the duration purchased and are based on "Workload Elements". Workload elements are defined as the granular unit of compute that is directly associated with an application service. Workload elements include both discovered and provisioned instances in any attached clouds. For a more thorough explanation of workload elements, review the knowledge base article at this URL

https://support.morpheusdata.com/s/article/What-is-a-Workload-Element-or-WE-for-purposes-of-Morpheus-licensing?language=en_US.

Component Licenses

Morpheus Data provides support services for the Morpheus platform and limited support for the underlying open source components when deployed in compliance with the supported architectures defined herein.

Refer to the Components --> Open Source section of this document for a full listing of all open source components.

If deploying a Multi-Site Active/Passive Architecture, Elasticsearch Cross Cluster Replication requires the purchase of an Elasticsearch Platinum Level subscription directly from Elasticsearch.

Repositories

Access to base “yum” and “apt” repositories (customer or publicly hosted) is required in order to deploy Morpheus.

Operating Systems

The following operating systems are supported for the latest version of Morpheus.

- Amazon Linux 2
- CentOS 7.x, 8.x
- RHEL 7.x, 8.x
- Ubuntu 16.04 or 18.04
- Debian 8, 10
- SUSE SLES 12, 15

Note: Superuser privileges via “sudo” command for the user installing the Morpheus application package is required. Configure the firewall to allow access from users on port 443.

Networking Requirements

Connectivity

Requirements for installation:

- Network connectivity from the administrator(s) to the Morpheus node(s) over TCP port 22 and 443.
- Network connectivity from the Morpheus node(s) to the yum/apt repos
- Network connectivity from the users to the appliance over TCP 443 (HTTPS).
- Virtual Machines and Docker-based hosts must be able to reach the Morpheus node(s) IP address on port 443

Note: Additional port and protocol requirements maybe found in Appendix B.

Name Resolution

Morpheus node(s) must be self-resolvable to their own hostname, FQDN, and static IP address prior to installation. Managed machines must be able to resolve the Morpheus appliance.

Latency

In a distributed architecture latency under 5ms is strongly recommended for acceptable performance.

SECURITY

Authentication

Morpheus is capable of integrating with several single sign-on solutions. These solutions require mapping security groups to user roles in Morpheus ensuring proper role assignment at first login. Review this URL https://docs.morpheusdata.com/en/latest/integration_guides/IdentityManagement/IdentityManagement.html#identity-management for details of the current compatible identity provider integrations. Morpheus is capable of supporting SAML integration which supports multi-factor authentication.

Note: For multi-factor authentication, the identity provider must have a configured and working solution and the integration must be of type SAML.

Encryption

Protected data in the database is AES-256 symmetric key encrypted. Protected data covers configuration data, passwords, and configuration metadata. Data at rest on the file system is not client-side encrypted but can exist on an encrypted file system.

FIPS 140-2

Morpheus 3.6.5 or greater supports Federal Information Processing Standard (FIPS) 140-2. If FIPS is required customers should ensure they are using the Morpheus FIPS installer.

FIPS 140-2 is a U.S. and Canadian government standard that specifies security requirements for cryptographic modules. If FIPS 140-2 is desired ensure the FIPS Morpheus installer is utilized.

Certificates

Self-signed certificates are public-key certificates that are signed and validated by the same person. Self-signed certificates are ideal only for testing purposes. Certificate authority signed certificates are preferred and recommended for all other purposes. Morpheus supports using either type of certificate.

Note: If integrating with Active Directory a valid trusted CA-signed domain certificate is required.

Compliance

Each minor version (4.1, 4.2) is scanned for Common Vulnerabilities and Exposures (CVEs) and remediated prior to release. When necessary, patch versions (4.1.1, 4.1.2) are released to address immediate CVEs. CVEs addressed in each version are documented in the release notes.

Log Types

Morpheus is capable of servicing large amounts of log traffic by utilizing Elasticsearch and buffered log transmission protocols. Morpheus provides a highly efficient and highly scalable solution for capturing log data. Logs can also be forwarded to external third-party log services.

Morpheus Server Log

Morpheus Server Logs are rotated every 24 hours with 30-day retention. These logs include check server, guacd, elasticsearch, mysql, nginx, rabbitmq, and redis (if applicable).

Audit Log

The audit log documents system changes made by users. For example, create and delete instances. Audit logs are stored on the file system with the same retention as the Morpheus Server Logs. The Activity feed, visible through the UI, is a subset of the Audit log which is stored indefinitely in the database.

Agent Log

Application logs are sent to the Morpheus Server from managed machines with the Morpheus Agent installed. ElasticSearch is used for this purpose. The Morpheus Agent forwards syslog messages to Morpheus with default retention of 7 days.

Telemetry

Morpheus Data has the ability to receive customer's telemetry data for analysis purposes. Transmission of telemetry data can be disabled as a license feature if desired.

Languages

Morpheus' default language is English, changing it requires altering web browser language settings.

Morpheus supports other user interface languages including those found at https://docs.morpheusdata.com/en/latest/getting_started/requirements/requirements.html?highlight=languages#supported-languages

COMPONENTS

Open Source

Morpheus utilizes open source components that are reviewed for updates at every major release (4.1, 4.2, etc.). In between these major releases, if a Common Vulnerability and Exposure (CVE) necessitates an update, it will be done in a minor release (4.1.1, 4.1.2, etc.).

Version 4.x uses the following Open Source software:

- Apache Tomcat
- Nginx
- Elasticsearch
- MySQL
- Erlang
- RabbitMQ
- Redis (Morpheus version 4.1 or prior)
- Apache Guacamole
- Java Open JDK JRE

Compatible versions may be found in the release notes under Service Version Compatibility https://docs.morpheusdata.com/en/latest/release_notes/compatibility.html#morpheus-service-version-compatibility. Care should be taken to ensure component version compatibility is maintained during administrative operations including upgrades and updates.

Additional information regarding Morpheus Data's use of open-source software and licenses can be found at <https://www.morpheusdata.com/licensing>.

Morpheus Agent (Optional)

The Morpheus Agent is lightweight, secure, and exists for both Linux and Windows-based platforms. It provides greater visibility into logs and stats and is capable of processing instructions instigated in the Application Tier. The Morpheus Agent opens an outbound connection from the managed machine back to the Morpheus Application Tier over port 443 (HTTPS or WSS protocol). This enables a bi-directional command bus enabling the orchestration of workloads without needing credentials or access to protocols like SSH or WinRM. The Morpheus Agent may be installed over Cloud-init, Windows unattend.xml, VMware Tools, SSH, WinRM, Cloudbase-init, or manually.

Note: The Morpheus Agent is not required by Morpheus to manage an instance however, the accuracy of stats will vary based on the integrated cloud's capability, SSH or WinRM and credentials will be required for machines.

Capabilities

The Morpheus Agent does the following:

- Provides a command bus which allows Morpheus to orchestrate automation on managed machines without credentials
- Accepts and executes commands and scripts
- Provides connection persistence over HTTPS web socket and runs as a service
- Buffers and compresses logs and sends them in chunks to minimize packet transfers

Supported Operating Systems

Morpheus Agent supported Operating Systems may be found at https://docs.morpheusdata.com/en/latest/getting_started/agent/osSupport.html

MORPHEUS TIERS

There are four tiers of services associated with the Morpheus Application. They are the Application Tier, Transactional Database Tier, Non-Transactional Database Tier, and Messaging Tier. Each of these tiers serve to enable specific Morpheus functionality. Morpheus supports various architectures in which these tiers can run

including a single machine, distributed across machines or clustered for redundancy. An explanation of each tier is as follows:

Application Tier

The application tier runs stateless services including NGINX, Tomcat, and Guacamole. To provide redundancy Application Tier servers must be placed behind a load-balancer. Additionally, the Application Tier requires shared storage at `/var/opt/morpheus/morpheus-ui/*` when more than one server is deployed in order to share uploaded virtual images, deployment archives, logos, Ansible, Terraform, and Morpheus backups.

NGINX

The NGINX component of Morpheus provides SSL termination and cache proxy for the Tomcat container. NGINX is open-source software built for web serving and is designed for maximum performance and stability. NGINX uses an asynchronous event-driven approach, rather than threads, to handle requests. NGINX's modular event-driven architecture provides for more predictable performance under high loads.

Apache Tomcat

The Tomcat component of Morpheus hosts the Morpheus Application. Apache Tomcat is an open source implementation of the Java Servlet, JavaServer Pages, Java Expression Language, and WebSocket technologies.

Apache Guacamole

The Guacamole component of Morpheus provides a client-less remote console for instances, hosts, virtual machines, and bare metal. Platform type and cloud settings determine the protocol and port used for remote console connections. Using Morpheus, Guacamole is capable of providing a remote console via SSH, RDP, and VNC over associated standard ports.

Redis (Version 4.1.x or prior)

The Redis component of Morpheus provides a log sequence counter. Redis is an in-memory data structure store, used as a database, cache, and message broker. It supports data structures such as strings, hashes, lists, sets, sorted sets with range queries, bitmaps, and geospatial indexes with radius queries and streams. Redis was removed from Morpheus beginning with Version 4.2.

High Availability

In order to provide high availability for the Application Tier, minimally two web servers should be deployed behind a load balancer. These servers require access to shared storage. This tier can scale vertically or horizontally assuming all servers would have access to the shared storage. The Application Tier runs only “stateless” services. All communications between tiers and workloads go through the Application Tier with the exception of each tiers inter-cluster communications.

Shared Storage

Redundant configurations on the Application Tier require a shared file system so that all nodes within the Morpheus cluster are able to connect to necessary files such as white label images, uploaded virtual images, deploy uploads, Ansible plays, Terraform, and Morpheus backup. This storage can be externalized to an object storage service or a simple NFS cluster.

Load Balancer

For the Application Tier, load balancing for the Morpheus URL over port 443 (layer 7) is required for highly available solutions. Sticky Sessions should be enabled. The Least Connection load-balancing algorithm is recommended for use with the Application Tier as it is typically used when session persistence is enabled since traffic can become unevenly distributed if other algorithms such as Round Robin are used. Health Checks should be enabled in order to redirect customers to active servers. SSL Pass-Through is supported for use with Morpheus however, SSL offloading is not.

Non-Transactional Database Tier

The Non-Transactional Database Tier consists of Elasticsearch. The Elasticsearch component of Morpheus enables logs and metrics from managed and discovered machines. Elasticsearch is a distributed, RESTful search and analytics engine that is capable of high write throughput at scale. If a distributed architecture is desired, an Elasticsearch cluster must be created.

High Availability

Minimally, a three-node cluster connected over transport is required for Elasticsearch high availability. Elasticsearch nodes can be added to the cluster to increase its capacity and reliability. To preserve performance the nodes of an Elasticsearch cluster should be on the same network within the same datacenter. Elasticsearch should have multiple master nodes to avoid split-brain scenarios. By default, a node is both a data node and eligible to be elected as the master node that controls the cluster. Master nodes store

detailed cluster state and data nodes are responsible for storing and querying the actual index data.

Elasticsearch efficiently stores and indexes all types of data in a way that enables fast searches. When multiple Elasticsearch nodes are in a cluster, stored documents are distributed across the cluster and can be accessed immediately from any node. Elasticsearch balances multi-node clusters to provide scale and high availability. A network load balancer should not be used with this cluster, Morpheus manages and distributes Elasticsearch requests across the cluster.

An Elasticsearch index is a logical grouping of one or more physical shards, where each shard is an index. Each index belongs to one primary shard and one or many replica (redundant copies) shards. By distributing the documents in an index across multiple shards and distributing those shards across multiple nodes Elasticsearch provides redundancy. By default, Morpheus creates 1 index per day for the activity feed, 1 index per day for logs, 1 index per day for monitor check results, and 1 index per day for stats. By default 1 replica is configured per index.

Note: When deploying a distributed architecture, Morpheus recommends customers utilize the Elasticsearch installer with the embedded Java option.

[Messaging Tier](#)

The Messaging tier is an AMQP based tier where Morpheus uses RabbitMQ for queue services. The RabbitMQ component of Morpheus provides messaging capabilities within the application. RabbitMQ can run as a single instance however, for high availability a cluster of at least 3 nodes is required.

Note: Morpheus Version 4.2.1 and prior require the use of the STOMP Protocol (for agent communication) and a network load balancer for clustered implementations. As of Morpheus Version 4.2.2 these are no longer required.

High Availability

Minimally, a three-node RabbitMQ cluster should be established to accomplish single site high availability. Clustering and Queue mirroring are meant to be used across LAN, WAN use is not recommended. Each RabbitMQ node should be configured in a similar manner to include compatible versions of RabbitMQ and Erlang as well as similar installation locations, firewall settings, hosts file, erlang cookie file, and start-on-boot configurations. Care should be taken to ensure version consistency with the respective Morpheus release. When upgrading Morpheus, if applicable the RabbitMQ cluster must be upgraded

prior to performing the Morpheus upgrade. A network load balancer should not be used with this cluster, Morpheus manages and distributes RabbitMQ requests across the cluster as of Morpheus Version 4.2.2.

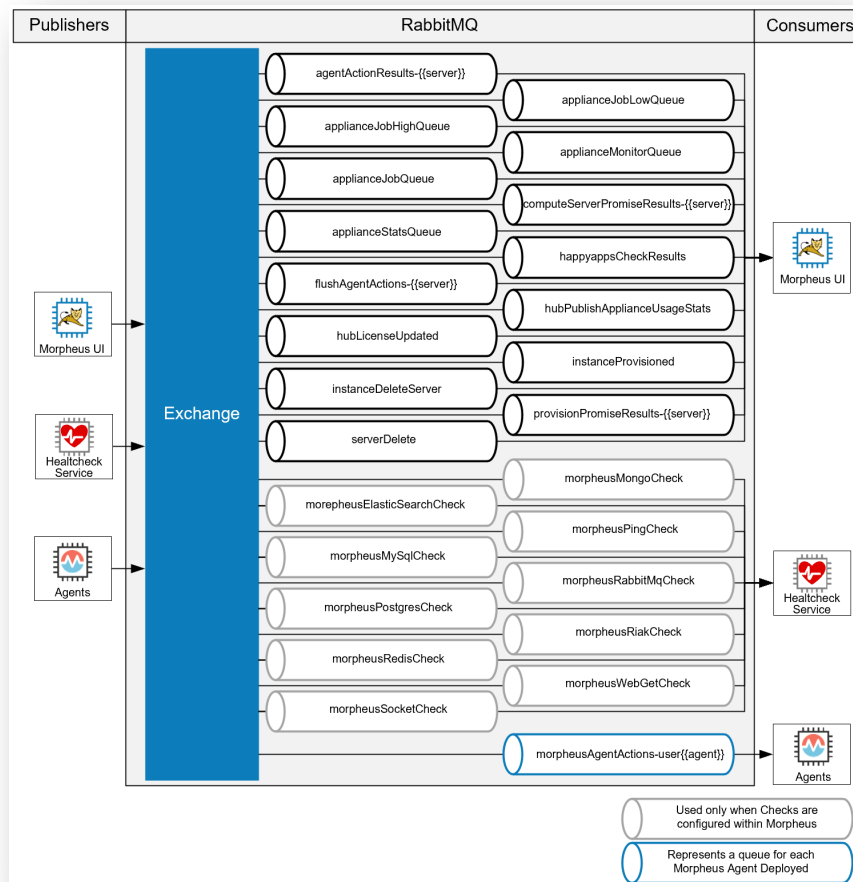
RabbitMQ clusters tolerate the failure or shutdown of individual nodes which can be restarted and rejoin the cluster as long as they can contact a cluster node within five minutes of boot. Nodes in a RabbitMQ cluster are equal peers. RabbitMQ nodes are identified by unique node names that are appended to hostnames which must be resolvable. RabbitMQ nodes must share a secret called the erlang cookie in order to communicate. A copy of the erlang cookie is needed at \home of root and for each non-privileged user that intends to utilize CLI tools.

By default, message queues are reachable from all nodes. However, queue mirroring should be enabled to allow queue contents to be replicated across nodes. Each mirrored queue consists of one queue master, operations are applied to the master and then propagated to mirrors. Queue mirroring enhances availability but not load distribution since each node must perform the same operations. If the queue master fails, that oldest synchronized mirror is promoted to master.

Note: Morpheus recommends the use of only disk nodes.

Queue mirroring is configured by setting policies. Policies match queues by name using regular expressions. Details about these policies and how to configure them can be found within the [How to set recommended 3-node RabbitMQ policies](#) Knowledge Base article. The image below shows the publishers and consumers of Morpheus queues.

Figure 1. Message Queues



Note: Messaging Tier (Morpheus Version 4.2.1 or prior):

STOMP PROTOCOL

The STOMP plugin and associated ports are required for Morpheus Versions 4.2.1 and earlier.

LOAD BALANCER

For the Messaging Tier for Morpheus Versions 4.2.1 and earlier load balancing should be configured for AMQP and STOMP (Layer 4) for highly available solutions. Sticky sessions should be enabled. The Least Connection load-balancing algorithm is recommended for use with the Messaging Tier as it is typically used when session persistence is enabled since traffic can become unevenly distributed if other algorithms such as Round Robin are used. Both SSL Offloading and SSL Pass-Through are supported for use with RabbitMQ.

Transactional Database Tier

The Transactional Database Tier consists of a MySQL compatible database. The MySQL component of Morpheus provides a logistical data store. If redundancy is desired, it is recommended that a lockable clustered configuration be used.

High Availability

When high availability is required, Morpheus recommends deploying a Percona XtraDB cluster for the Transactional Database Tier. A Percona XtraDB cluster provides a highly available and scalable, tightly coupled database cluster. It consists of a recommended three nodes where each contains the same set of data synchronized across all nodes. These nodes must physically reside close to each other and cannot be geographically diverse. A network load balancer should not be used with this cluster. Morpheus manages the load balancing of the Transactional Database Tier and can be configured for either failover or load balancing of the cluster's nodes.

Percona XtraDB Cluster integrates Percona Server for MySQL running with the XtraDB storage engine, and Percona XtraBackup with the Galera library to enable multi-master replication. Multi-master replication means that any node can be written to and then the write will be applied to all nodes in the cluster. This type of replication is called virtually synchronous replication because the primary may apply events faster than the secondary nodes. This cluster is capable of scaling read workload as any node can provide read data however, each node must write all the same data. With a Percona XtraDB Cluster, nodes can be lost without losing data as long as at least one node remains.

The size of the cluster is used to determine the required votes to achieve quorum and a quorum vote is done when nodes are suspected of being down. Write operations are blocked for a configurable no response timeout setting for the duration slightly longer than the configured timeout value when a node does not go down gracefully. To add split-brain protection in cases where the addition of a third node is not possible an arbitrator node may be deployed. An arbitrator does not store any data or run mysqld but is a voting member of the cluster and replicates data.

The Percona XtraDB Cluster uses Solid State Snapshot Transfer (SST) methods of mysqldump, rsync, or extrabackup when all data is copied from one node to another. Using mysqldump or rsync requires a read-only lock on the database. When only incremental changes are copied from one node to another the Percona XtraDB Cluster uses Incremental State Transfer (IST). IST

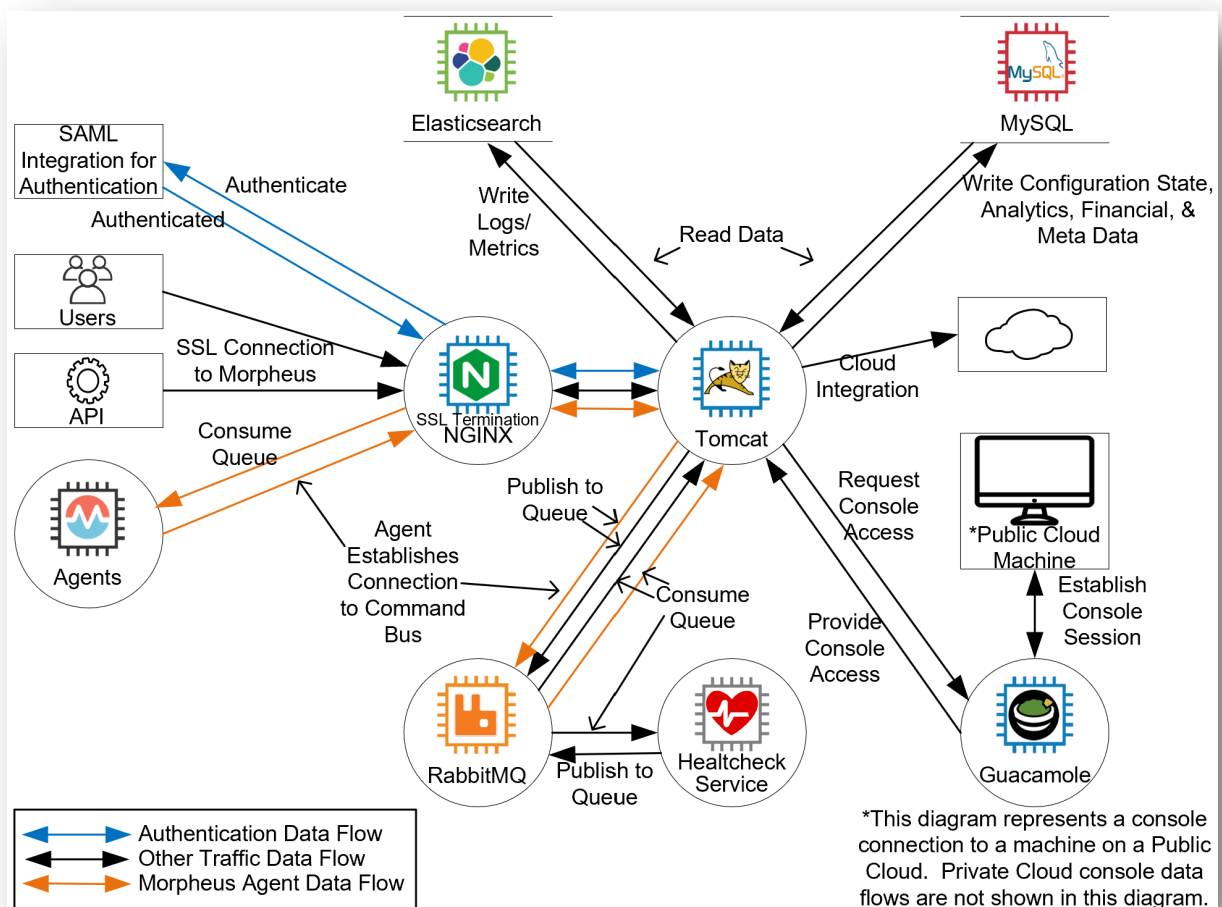
is implemented using a caching mechanism on each of the nodes. A ring-buffer of last changes is configured and the node is able to transfer this cache. If the amount of changes exceeds the ring-buffer the joining node must perform SST.

Note: Avoid creating a cluster of an even number of nodes as this can lead to a split-brain situation.

Note: The choice to utilize database software other than that shipped with Morpheus (MySQL) is the customers. It is the customer's responsibility to maintain their own database software regardless of the recommendations for clusters contained within this document.

Morpheus Data Flow

Figure 2. Data Flow Diagram



SUPPORTED MORPHEUS ARCHITECTURES

The architectures described below are supported by Morpheus Data, deviation from these architectures may result in unsupported configurations or unexpected performance. Morpheus professional services should be consulted prior to deviating from any of the below architectures.

Table 1. Minimum Server Count Per Architecture

Architecture	Combined Tiers	Distributed Tiers
Single Server Architecture	1 Server	
Single Site Redundant Architecture	3 Servers	11 Servers
Multi-Site AWS Multi-AZ Architecture		3 to 6 Servers plus Amazon Services
Disaster Recovery Active/Passive Architecture	2 to 22 Servers depending on selected options	

Note: Please refer to Appendix A for Morpheus Server resource sizing recommendations.

Single Server Architecture

Morpheus' Single Server (non-redundant architecture) is supported and recommended in all environments where application-level redundancy is not a requirement. In production environments infrastructure-level high availability may be provided in order to mitigate some risks associated running without application level redundancy. This architecture has been tested and proven to support up to 5,000 managed machines with the Morpheus Agent installed. Although, the maximum achievable number of managed machines may vary based on specific environment configurations and integrated services.

This architecture consists of hosting all Morpheus Tiers on a single server. It allows for a simple installation method as well as simplified ongoing maintenance and troubleshooting. Connectivity between vital services is not dependent on the underlying network allowing for rapid application processes independent of network speed or availability. Morpheus provides updates to all components via the Morpheus Installer however, downtime during upgrades is required. This architecture may be scaled vertically by adding additional CPU and/or RAM or horizontally by moving to one of the redundant architectures described in the sections below.


Figure 3. Single Server - Non-Redundant Combined Tiers Conceptual Diagram



Single Site Redundant Architectures

Morpheus redundant architectures are supported in all environments where single site single points of failure are to be avoided. These architectures are suitable for production deployments and can be scaled both vertically and horizontally. They allow customers to achieve single site high availability assuming all underlying hardware and dependencies are free of single points of failure. If deployed on resource managed infrastructure such as that provided by the vSphere Distributed Resource Scheduler (DRS), it is recommended that anti-affinity rules be created to ensure redundant servers are not allowed to run on the same host.

Redundant architectures require both shared storage and load balancing. Connectivity between vital services is dependent on the underlying network which could potentially cause application instability or performance issues if it is unreliable. Redundant architectures require the establishment of clusters for the Non-Transactional Database, Messaging, and Transactional Database tiers. These clusters are sensitive to network interruptions and should be maintained with an odd number of nodes to ensure split-brain scenarios are avoided. When performing upgrades or maintenance activities, care should be taken to ensure all servers are running the same version of Morpheus as well as the same and compatible versions of the underlying software and operating system patches. These redundant architectures do not require downtime during upgrades



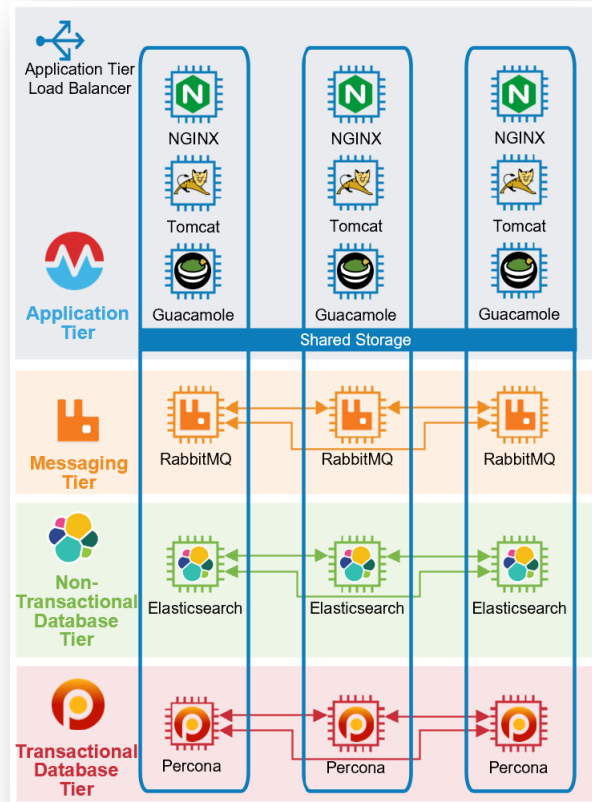
however, the Morpheus installer will only upgrade software components that reside on the same server as the Application Tier. Upgrades of distributed components are not supported by the Morpheus Installer and are the customer's responsibility.

Redundant Combined Tiers (3-Node Architecture)

The Morpheus Redundant Combined Tiers architecture (aka 3-Node Architecture) consists of hosting all Morpheus Tiers on each of three servers configured with a cluster for each tier and a network load balancer configured for the Application Tier. Given the need to establish clusters for each of the Non-Transactional Database, Messaging and Transactional Database tiers across a single set of three servers, ongoing maintenance and troubleshooting may become complex. With this architecture, Morpheus should be configured for failover of the Transactional Database Tier. This enables pointing to a single database server and failing over if it becomes unavailable. This architecture scales both vertically and horizontally. Scaling horizontally would require moving to the distributed tiers architecture described in the section below.

Option: The Transactional Database Tier may be configured on separate servers if desired.

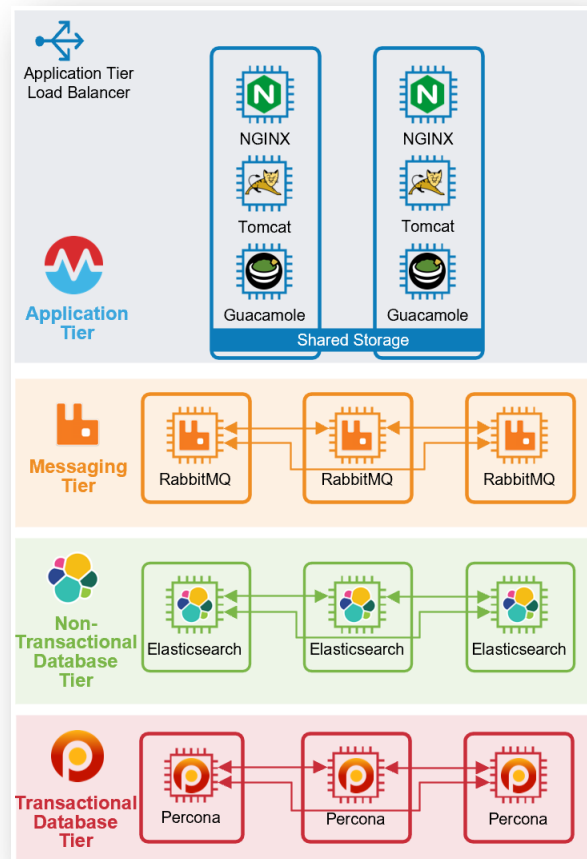
Figure 4. Redundant Combined Tiers (3-Node Architecture) Conceptual Diagram



Redundant Distributed Tiers (11-Node Architecture)

The Morpheus Redundant Distributed Tiers Architecture (11-Node Architecture) consists of distributing each tier onto a different server where at least three like servers are dedicated to each of the Non-Transactional Database Tier, Messaging Tier and Transactional Database Tier clusters and at least two servers are dedicated to the Application Tier. The distribution of tiers allows customers the ability to accommodate traditional data center network segmentation. As compared to the architectures defined in the sections above this architecture consumes significantly more resources. This architecture is typically only deployed in production environments and can be scaled both vertically and horizontally at a single site. If scaling horizontally an odd number of cluster nodes should be maintained to protect against split-brain situations. Given the requirement to establish distributed services and clustered tiers, ongoing maintenance and troubleshooting may become more complex in this architecture versus that of the architectures detailed in the sections above.

Figure 5. Redundant Distributed Tiers (11-Node Architecture) Conceptual Diagram



Multi-Site Architectures

AWS Multi-Availability Zone Architecture

Morpheus AWS Multi-Availability Zone (AZ) Architecture is supported in all environments where the use of AWS Services is available. By using this architecture customers reduce the amount of resources and the operational overhead associated with deploying Morpheus to span multiple sites. Each AWS Region is a separate geographic area with multiple, isolated locations known as Availability Zones (AZ). To prevent data loss and minimize downtime in the event of a service disruption it is recommended to distribute cluster nodes across AWS AZs in the same region. This active/active architecture is viable due to the LAN like connectivity between AZs provided by AWS. Because this architecture spans three AWS AZs customers have the ability to lose up to two AWS AZs while still providing Morpheus services.

Note: Spanning AWS Regions is not supported.

This architecture is essentially the Single Site Redundant Distributed Tiers architecture deployed in a manner that spans three AWS AZs for the Application and Messaging Tiers and utilizes Amazon Elasticsearch and Amazon Aurora for the Non-Transactional and Transactional Database Tiers spanning the same three AWS AZs. Utilizing this architecture increases Morpheus's availability above that of other reference architectures while also reducing the resource requirements below that of all but the single server reference architecture. This architecture reduces the operational overhead incurred by taking advantage of Amazon services.

This architecture can be scaled both vertically and horizontally. The use of Amazon Elastic File System (EFS) to provide shared storage and an Amazon Elastic Application Load Balancer is required.

Note: DNS name resolution must be available for all nodes and managed machines.

Application Tier

Three Application Tier (web) servers should be deployed one in each AZ. An AWS Elastic Application Load Balancer and Amazon Elastic File System (EFS) shared storage should be configured to support each Application Tier server. Highly available configurations within the Application Tier require a shared file system so that all nodes within the Morpheus cluster are able to connect to necessary files such as white label images, uploaded virtual images, deploy uploads, Ansible plays, Terraform, and Morpheus backup. Additionally, name resolution, NTP, and authentication solution connectivity should be available for all systems across AZs.

LOAD BALANCER

An AWS Elastic Application (Layer 7) load balancer should be configured for the Morpheus URL over port 443 to balance the load for across each Application Tier server. The load balancing algorithm should be set to Least Outstanding Requests with load balancer stickiness enabled.

Non-Transactional Database Tier

By utilizing Amazon Elasticsearch (ES) 7.x for the Morpheus deployment customers reduce their operational overhead while increasing availability. Amazon ES is an AWS service that should be configured to span the same three AZs as the Application Tier. With this configuration of nodes and availability zones if an availability zone experiences an outage it will result

in no downtime as two-thirds of the data nodes would still be available to elect a new master. By default, a node is both a data node and eligible to be elected as the master node that controls the cluster. Master nodes store detailed cluster state and data nodes are responsible for storing and querying the actual index data. Morpheus recommends the utilization of node-to-node encryption which enables TLS 1.2 encryption for Elasticsearch communications within the VPC.

Note: Node failures can cause the cluster's remaining data nodes to experience a period of increased load while Amazon ES automatically configures new nodes to replace the missing ones.

Note: Node-to-node encryption must be configured at the time of ES domain creation, it is not possible to configure node-to-node encryption on existing domains. Once configured node-to-node encryption cannot be disabled.

Note: It is the customer's responsibility to ensure version compatibility is maintained between Morpheus and Amazon ES.

Messaging Tier

Minimally, a three-node (maintaining an odd number of nodes) RabbitMQ cluster should be established to provide RabbitMQ high availability. Deploying these three nodes in each of the same three AWS AZs as the Application Tier is recommended. Clustering and Queue mirroring are meant to be used across LAN like connectivity (provided by AWS between AZs), WAN use is not recommended.

Option: The Messaging and Application Tiers may be combined on the same servers.

Note: Morpheus Version 4.2.1 or prior:

The STOMP protocol and an AWS Elastic Network (Layer 4) load balancer should be configured for AMQP and STOMP to balance the load across the RabbitMQ cluster. The load balancing algorithm should be set to Least Outstanding Requests with load balancer stickiness enabled.

Transactional Database Tier

Amazon Aurora is a fully managed relational database engine that's compatible with MySQL. Aurora is part of Amazon's managed database service Amazon Relational Database Service (Amazon RDS). Amazon

RDS is a web service used to set up, operate, and scale a relational database in the cloud.

With Amazon Aurora MySQL version 2.x single-master clusters (only one DB instance has write capability) are considered highly available. Amazon recommends distributing the Primary DB Instance and Aurora Replicas in the DB cluster over multiple Availability Zones to improve the availability of the DB cluster. Aurora replicas are read-only and primary database instances are read-write capable. In order to increase availability, Amazon recommends using Aurora Replicas as failover targets. If the primary instance fails, an Aurora Replica is promoted to the primary instance. There is a brief interruption during which read and write requests made to the primary instance fail with an exception, and the Aurora Replicas are rebooted. Promoting an Aurora Replica is much faster than recreating the primary instance.

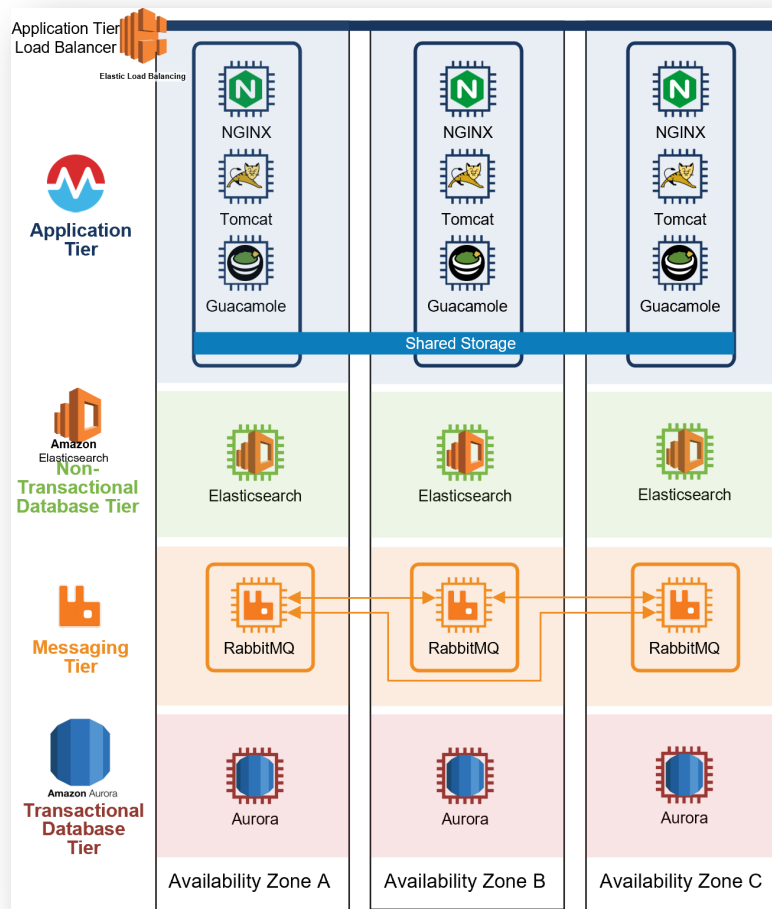
Note: Aurora MySQL version 1 is not compatible with Morpheus as of the writing of this document. With Aurora MySQL version 1, multi-master clusters (all DB instances have read-write capability) are considered continuously available. Aurora MySQL version 1 is compatible with MySQL 5.6 however, Morpheus requires MySQL 5.7. Morpheus is compatible with Aurora MySQL Version 2, should Amazon add multi-master capability to Aurora MySQL Version 2 in the future then it may be used with Morpheus.

Aurora's MySQL compatible database engine is customized to take advantage of a high-performance distributed storage subsystem that automatically grows as needed. An Amazon Aurora DB cluster consists of one or more DB instances and a cluster volume that manages the data (a separation of storage and compute). An Aurora cluster volume is a virtual database storage volume that spans multiple Availability Zones, with each Availability Zone having a copy of the DB cluster data. The data remains safe even if all of the DB instances become unavailable. Aurora synchronously replicates data across Availability Zones in the same region to six storage nodes associated with the database cluster volume.

Note: It is the customer's responsibility to ensure version compatibility is maintained between Morpheus and Amazon Aurora.

Note: Follow Amazon Aurora's recommendations if replication to a different region is required for disaster recovery use cases.
<https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/AuroraMySQL.Replication.CrossRegion.html>

Figure 6. AWS Multi-Availability Zone Conceptual Diagram



DISASTER RECOVERY

There are several options available regarding disaster recovery and Morpheus. Customers may choose to simply restore a database backup to a warm standby architecture at a secondary site or they may choose more complex options based on their RPO, RTO, and that of the underlying technologies and dependent services. The architecture below shows several options for customers interested in disaster recovery planning.

Multi-Site Active/Passive Architecture

Morpheus Multi-Site Active/Passive Architecture is supported in all environments where a secondary failover site is required. This architecture provides the capability to failover and is intended only for disaster recovery use cases. With this architecture, users must never point to both locations at the same time. Failover and failback are not automated, given the disaster recovery use case it is expected that a specific decision must be made to execute a manual failover, and configuring failback would be a manual process as well that would occur only after the threat has been neutralized.

Note: Automated failover and failback is not a capability currently on the Morpheus roadmap.

This architecture allows customers to achieve only failover capability of Morpheus, access to underlying technologies and dependent services is required for Morpheus to function properly. This architecture requires both cross-region shared storage and cross-region load balancing, as well as, access to underlying and dependent data center services. With this architecture consistently stable WAN connectivity is important to enable a viable failover solution. When performing upgrades and/or maintenance care should be taken to ensure all servers in all regions are running the same version of Morpheus as well as the same version of component software and operating system patches.

Option: Customers may choose different or similar single site architectures for both the primary and failover sites. They may even choose to accept a reduction in availability and capability at the failover site as compared to those offered at the primary site. Operational and availability requirements during a disaster event should be fully understood when choosing the architecture and defining the capabilities of the failover site. In any case, care should be taken to ensure the failover site maintains enough resources for the replicated Morpheus data.

Note: Follow all recommendations in the selected single site architecture sections above in addition to the recommendations within this section.

Application Tier

Morpheus recommends two web servers deployed at each location minimally in order to establish high availability for this tier at each location. Cross-region load balancers and cross-region shared storage must be properly configured to ensure each region's Application Tier operates as expected. Additionally, name resolution, NTP, authentication solution, and integration connectivity should be available for all systems across regions.

LOAD BALANCER

A cross-region load balancer should be configured for active/passive failover between sites. The load balancer should direct users to a single site at a time. During a failover event, the load balancer should redirect users to the failover site after replication from the primary site has been halted, never allowing customers to access both sites at the same time.

Non-Transactional Database Tier

Elasticsearch Cross-Cluster Replication (CCR) available in version 6.7 or higher enables pull-based (driven by the follower) replication of indices from one Elasticsearch cluster to another. Morpheus supports Elasticsearch 7.x in Morpheus 4.2 or later releases. This technology replicates data between two distinct clusters, in order to provide high availability at both locations follow the recommendations in the Redundant Architectures section to establish a cluster at each location prior to configuring CCR. Cross-cluster replication works by replaying the history of individual write operations that were performed on the shards of the leader index. Soft deletes occur whenever an existing document is deleted or updated, by retaining these soft deletes on leader shards they are made available for replay. CCR is active-passive since the leader index can be written directly and the follower index cannot. The leader is capable of accepting index writes and the followers have read-only copies of the index. When a leader index is not available, another index must be explicitly chosen for writes by the cluster administrator.

Note: CCR requires the purchase of an Elasticsearch Platinum Level subscription.

Option: Customers may choose not to replicate Elasticsearch data to the failover site. This will result in the loss of historical metrics, stats, and logs collected from machines (logs would still exist on the individual machines). Cost information would not be lost as it is stored in the SQL database.

Messaging Tier

Of the two replication plugins available the Federation plugin is the only one supported by Morpheus. Do not use the Shovel plugin with Morpheus.

The Federation plugin allows an exchange or queue in one cluster to receive messages published to an exchange or queue of another. Exchange federation links will start on any node in the downstream cluster and will failover to other nodes in case of an outage. The Federation plugin is designed to tolerate intermittent communication and aims to provide opinionated distribution of exchanges and queues using the erlang client. The Federation plugin communicates via the Erlang AMQP client and is included with the RabbitMQ distribution. Because this technology connects two distinct clusters, in order to provide high availability at both locations follow the recommendations in the Redundant Architectures section to establish a cluster at each location prior to configuring the Federation plugin.

Option: Customers may choose not to replicate RabbitMQ data to the failover site. This will result in the loss of any queued actions. The effects of this would be minimal and would typically go unnoticed in most environments.

Transactional Database Tier

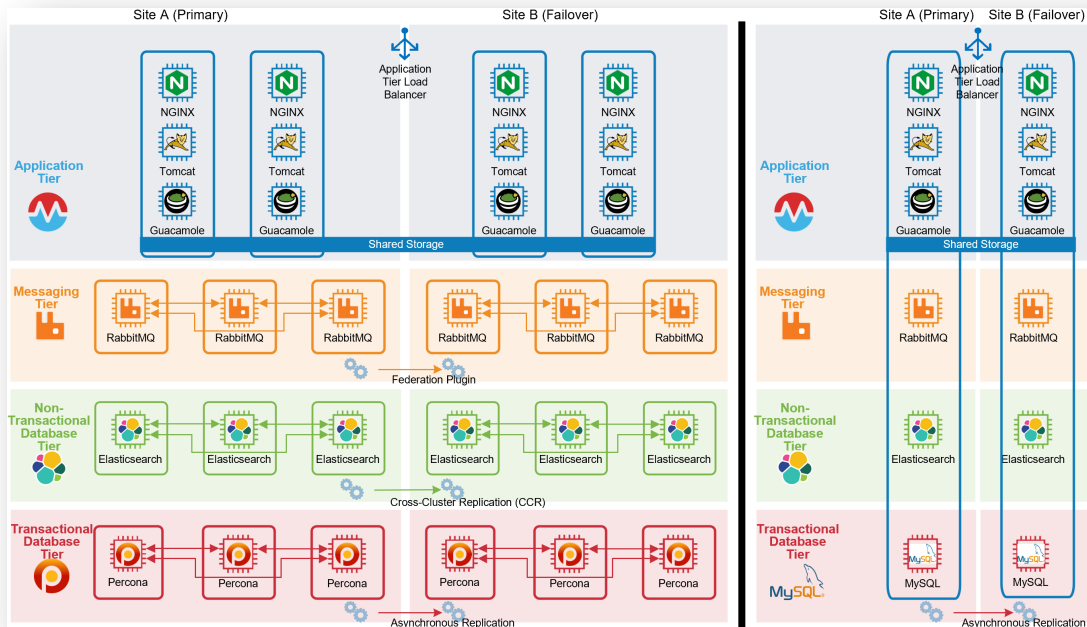
Morpheus recommends using standard MySQL asynchronous replication to span geographic locations establishing loosely coupled database clusters. Within the Transactional Database Tier section of this document, a tightly coupled database cluster utilizing the Galera library to configure multi-master replication is described for single-site high availability. Tightly coupled database clusters and/or clusters using multi-master replication are not recommended to span geographic locations. Given that the failover site is geographically separated by a presumed significant distance with WAN connectivity the only viable solution is to use asynchronous replication. Asynchronous replication allows for independence in processing and applying each transaction.

Because this technology connects two distinct clusters, in order to provide high availability at each location, follow the recommendations in

the Redundant Architectures section to establish tightly coupled database clusters at each location prior to configuring asynchronous replication. It is possible to shift from one node to another inside either of the two clusters, while the asynchronous data continues passing from one source to the other. If using Percona ExtraDB, use the Replication Manager to ease configuration and management.

The image below shows a couple of sample conceptual diagrams. Depending on the options single site architectures chosen for each location the diagram may look quite different.

Figure 7. Disaster Recovery Conceptual Diagram SAMPLES



APPENDIX A: MORPHEUS SERVER SIZING

Table 2. Morpheus Server Sizing

Combined Tiers	
Memory	Recommended 16 GB (Minimum 8 GB)
CPU	4 Core 1.4 GHz (or better) CPU
Non-Redundant Architectures Storage	<p>Minimum 200 GB (Local Storage) - Morpheus binaries, virtual images, backups, logs, stats, user uploaded, and user imported data require adequate space on the Morpheus Server.</p> <p>Recommended Swap Disk - .5 times RAM if 16 GB or 1 times RAM if 8 GB.</p>
Redundant Architectures Storage	<p>Minimum 50 GB (Local Storage) - Morpheus binaries</p> <p>Recommended Swap Disk - .5 times RAM if 16 GB or 1 times RAM if 8 GB.</p> <p>Minimum 200 GB (Shared Storage) - Virtual images, backups, logs, stats, user uploaded, and user imported data require adequate space on the shared file system.</p>
Distributed Tiers	
Application Tier	
Memory	Recommended 16 GB RAM
CPU	2 Core 1.4 GHz (or better) CPU
Non-Redundant Architectures Storage	<p>Minimum 200 GB (Local Storage) - Morpheus binaries, virtual images, backups, logs, stats, user uploaded, and user imported data require adequate space on the Morpheus Server.</p> <p>Recommended Swap Disk - .5 times RAM</p>
Redundant Architectures Storage	<p>Minimum 50 GB (Local Storage) - Morpheus binaries</p> <p>Recommended Swap Disk - .5 times RAM</p> <p>Minimum 200 GB (Shared Storage) - Virtual images, backups, user uploaded, and user imported data require adequate space on the shared file system.</p>
Non-Transactional Database Tier	
Memory	Recommended 8 GB RAM
CPU	2 Core 1.4 GHz (or Better) CPU
Storage	<p>Recommended 150 GB (Local Storage) - Elasticsearch binaries for Elasticsearch servers.</p> <p>Recommended Swap Disk - 1 times RAM</p>
Messaging Tier	

Memory	Recommended 8 GB RAM
CPU	2 Core 1.4 GHz (or Better) CPU
Storage	Recommended 150 GB (Local Storage) - RabbitMQ and Erlang binaries
	Recommended Swap Disk - 1 times RAM
Transactional Database Tier	
Memory	Recommended 8 GB RAM
CPU	2 Core 1.4 GHz (or Better) CPU
Storage	Recommended 200 GB - Morpheus database, Percona binaries
	Recommended Swap Disk - 1 times RAM

Note: The swap size recommendation is .5 times the amount of RAM for systems with greater than 8 GB of RAM or 1 times the amount of RAM for a system with less than or equal to 8 GB of RAM. Customers may choose to follow their standard enterprise swap sizing best practices.

APPENDIX B: PORTS AND PROTOCOLS

Table 3. Ports and Protocols

Source	Destination	Port	Protocol	Description
User	Application Tier	443	TCP	User Access
Morpheus Servers	DNS Servers	53	TCP	Domain Name Resolution
Morpheus Servers	Time Source	123	TCP	Time Resolution
Morpheus Servers	Web or Offline Installer	80, 443	TCP	Download repos and Morpheus packages (yum/apt repos)
Managed Machine	Application Tier	443	TCP	Morpheus Agent Communications
Managed Machine	Application Tier	80, 443	TCP	Agent Installation. (Requires port 80 only for Ubuntu 14.04)
Managed Machine	Application Tier	N/A	N/A	Agent Installation Clout-init (Linux)
Managed Machine	Application Tier	N/A	N/A	Agent Installation Cloudbase-init (Windows)
Managed Machine	Application Tier	N/A	N/A	Agent Installation VMtools
Managed Machine	Application Tier	N/A	N/A	Static IP Assignment & IP Pools (Cloud-init or VMware Tools)
Managed Machine	Docker Image Repo	443	TCP	Applicable if using docker
Managed Machine	Application Tier	69	TCP/UDP	PXE Boot (Forwarded to internal PXE port 6969)
Application Tier	Managed Machine	5985	TCP	Agent Installation WinRM (Windows)
Application Tier	Managed Machine	22	TCP	Agent Installation SSH (Linux)

Morpheus Application Tier	Managed Machine	22, 3389, 443	TCP	Remote Console (SSH, RDP, Hypervisor Console)
Application Tier	AWS S3	443	TCP	Morpheus Catalog Image Download
Application Tier	Hypervisor	443	TCP	Hypervisor hostname resolvable by Morpheus Application Tier
Application Tier	Non-Transactional Database Tier	443	TCP	Applicable if using Amazon Elasticsearch Service
Application Tier	Docker CE Repo	443	TCP	Applicable only when integrated with Docker
Application Tier	Rubygems	443	TCP	
Application Tier	Morpheus Hub	443	TCP	(Optional) Telemetry data (Disabled only via license feature)
Application Tier	Mail Server	25 or 465	SMTP	Send email from Morpheus
Application Tier	Messaging Tier	5672	TCP	AMQP non-TLS connections
Application Tier	Messaging Tier	5671	TCP	AMQPS TLS enabled connections
Application Tier	Messaging Tier	61613	TCP	STOMP Plugin connections (Required only for Morpheus versions 4.2.1 or prior)
Application Tier	Messaging Tier	61614	TCP	STOMP Plugin TLS enabled connections (Required only for Morpheus versions 4.2.1 or prior)
Messaging Tier	Messaging Tier	25672	TCP	Inter-node and CLI tool communication
Administrator Web Browser	RabbitMQ Server Management	15672	TCP	Management plugin

Administrator Web Browser	RabbitMQ Server Management	15671	TCP	Management plugin SSL
Messaging Tier Cluster Node	Messaging Tier Cluster Node	4369	TCP	erlang (epmd) peer discovery service used by RabbitMQ nodes and CLI tools
Application Tier	Non-Transactional Database Tier	9200	TCP	Elasticsearch requests (Used in all cases except when utilizing AWS ES service)
Non-Transactional Database Tier	Non-Transactional Database Tier	9300	TCP	Elasticsearch Cluster
Transactional Database Tier	Transactional Database Tier	4567	TCP/UDP	Write-set replication traffic (over TCP) and multicast replication (over TCP and UDP).
Transactional Database Tier	Transactional Database Tier	4568	TCP	Incremental State Transfer (IST)
Application Tier	Transactional Database Tier	3306	TCP	MySQL client connections
Backup Solution	Transactional Database Tier	4444	TCP	State Snapshot Transfer (SST)
Application Tier	Integrated Technology	Varies	TCP	Integrations (Uses the port of the 3rd party systems API)

Figure 8. Port Diagram

