

High Availability

This document provides an overview of the components and suggested configuration to deploy Ubersmith in a way that ensures the availability of the system when a hardware or network failure occurs.

The use of physical or virtual machines for the deployment of Ubersmith are at the discretion of the administrator. In the past, we have advocated for the use of physical hosts for the Ubersmith appliance environment. With the improved performance of virtualized environments and the ease of deploying to the cloud, this is now less of a concern.

While a default deployment of Ubersmith's web frontend is straightforward enough that it can be contained on a single host, a high availability (HA) deployment can be achieved with the use of several open source / free components. This is the route many Ubersmith clients have chosen, rather than implementing proprietary third party systems, potentially at greater cost and complexity.

Load Balancing and Failover

Incoming connections from the Internet at large, an internal network, or VPNs should be handled by a redundant load balancer. This load balancer can be hardware or software based. We do not currently have any guidance on particular vendors with respect to hardware load balancing solutions. Typically, organizations large enough to be considering an HA configuration already have this hardware on hand.

An example of a software load balancer in use in a production capacity by many Ubersmith clients is HAProxy. While deploying a redundant HAProxy load balancer configuration is outside the scope of this document, a common method is to use the `keepalived` daemon to provide failover of virtual IP addresses between the two HAProxy instances. HAProxy also provides TLS termination for inbound connections.

HAProxy can determine which of our web frontend nodes are available to handle connections, and can then route them appropriately. This only scratches the surface in what the HAProxy software is capable of, but for our purposes, its role is fairly straightforward: handle incoming TLS HTTP connections, and pass them to a web frontend node.

Some DNS providers also provide monitoring that can allow for a change in the way traffic is routed, should an Ubersmith endpoint become unavailable. This can potentially remove the need for a load balancer entirely, with the caveat that the transition from primary to replica may take some time to occur, possibly resulting in downtime.

Further to this, many cloud providers support virtual IPs and/or automatic failover as part of their service offerings.

Web Frontend Nodes

While a stock deployment of the Ubersmith frontend (either via the Ubersmith Installer, or provided by our Support team) can be used as a starting point, several changes are required for Ubersmith to run correctly in a high availability configuration. As an example, let us consider two hosts with a stock deploy of the Ubersmith software that we would like to load balance.

By default, both hosts will be running containers for Solr, Cron tasks, Redis, and Percona Server 5.6 (MySQL). These services provide resources that both hosts must share, which we will discuss shortly. The simplest solution to remedy this need for shared resources is to deploy Ubersmith on a third (or more) host that will run **only** these services, which both web frontend hosts can then communicate with. The hosts acting as web frontend nodes can have these services disabled by removing them from the `ubersmith_start.sh` script, found in the Ubersmith home directory (usually `/usr/local/ubersmith`).

One caveat here is that the addition of this third host has the possible side effect of introducing a single point of failure into the architecture of the system.

Solr

Solr, which provides Search Services for Ubersmith, can further be configured in a cluster to provide failover for Ubersmith's search services. In practice, however, we are not aware of any clients that have chosen to implement such a configuration, and internally at Ubersmith we do not run Solr in a cluster configuration. All hosts acting as Ubersmith web frontends will need to be able to communicate with Solr to provide search services to users.

Cron

Ubersmith's Cron container executes polling and daily invoicing tasks. As such, only one instance of this container should be running at any given time. While there are safeguards in place for multiple concurrent invoicing processes running that should prevent the system from double-billing clients, it's best to avoid the situation in the first place. We do not currently have any guidance on how to provide high availability / failover for Ubersmith's cron tasks.

Redis

On this page:

On this page:

- [Load Balancing and Failover](#)
 - [Web Frontend Nodes](#)
 - [Solr](#)
 - [Redis](#)
 - [Percona Server and Percona XtraDB Cluster](#)
 - [Appliance Nodes](#)
 - [Database Backend](#)
- [Monitoring](#)
- [Kubernetes](#)
- [References](#)
- [Related Topics](#)

While Redis can be run in a cluster, Ubersmith does not leverage this functionality, and in fact, for Ubersmith to function properly, Redis should **not** be configured as a clustered service. When using a single database server, Ubersmith can leverage Percona Server's `GET_LOCK()` functionality to handle advisory locking within the software. This prevents situations where two users may be trying to act on the same unique item in Ubersmith - as an example, an IP address or an invoice - at the same time.

When using a clustered database solution like Percona XtraDB Cluster, the `GET_LOCK()` functionality is not available, and we rely on three distinct instances of Redis to determine if we have successfully locked an object in order to take action on it. See [Clustering Databases](#) for configuration guidance.

Percona Server and Percona XtraDB Cluster

Percona XtraDB Cluster 5.6 is the preferred database backend for high availability configurations. This fork of MySQL allows for any 'node' within the cluster to accept `INSERT` or `UPDATE` SQL commands. Additionally, a copy of all of Ubersmith's data is stored across each cluster node, providing redundancy. Support for PXC 5.7 is currently in testing, and is expected to be available soon.

HAProxy can again be pressed into service to determine the availability of nodes and to direct queries from the web frontend hosts. The cluster will remain available as long as 1 node is online; a minimum of 3 (or an odd number of) nodes is highly suggested to avoid a 'split brain' condition where none of the database nodes have an authoritative copy of the data. Unless required, it is usually safest to use a 'one writer' configuration where one of the database nodes is written to, and the other nodes are read-only. Percona provides documentation on a reference implementation using HAProxy.

Also worth considering is ProxySQL, which provides a more SQL-centric approach to proxying database connections. Here at Ubersmith, we are currently running ProxySQL with the hope of being able to advocate its use for our customers.

Appliance Nodes

Incoming connections from Ubersmith web frontend hosts are handled by a load balancing technology (software or hardware) deployed at the discretion of the administrator, just as for the Ubersmith frontend hosts. The load balancer can tell which of the appliance nodes are available to handle connections, and can then route them appropriately in the event the primary appliance fails.

Unlike the Ubersmith web frontend nodes, Ubersmith appliances are standalone devices, and do not share data or services between each other. There is no need to modify the configuration or services running on the appliance hosts. That being said, having multiple appliances poll the same devices could impact the performance of those devices. It may be wise to leave polling tasks disabled on the replica appliance until it is needed.

As the appliance hosts store significant amounts of data in RRD (round robin database) files, it may be more efficient to replicate this data between the primary and secondary appliance hosts, rather than have both performing polling tasks. There are many methods of keeping data synchronized between multiple hosts. DRBD or a similar network filesystem/mirroring technology can be used to meet this requirement. NFS is not recommended due to performance concerns.

Should the primary appliance host fail, the polling cron tasks on the secondary appliance need to be activated. Since the RRD data is already in place thanks to DRBD or other filesystem monitoring, metric collection can resume immediately.

Database Backend

Presently the Ubersmith appliance database requires the use of the MyISAM table engine, which precludes the use of most clustering solutions. Percona Server 5.6 is the supported RDBMS. For redundancy, replication must be enabled.

Configuration of automatic 'replica' promotion must be considered for a high availability configuration.

Monitoring

An external monitoring solution is suggested to continuously verify that all components of the cluster are functioning properly. Some DNS and cloud providers can provide external monitoring. While Ubersmith may be able to monitor some of its own cluster components, in the event of a database or cron failure, alerts for failed monitors may not be triggered or delivered. It is best to not rely on Ubersmith to monitor itself.

Kubernetes

Since Ubersmith is a containerized application using Docker based containers, Kubernetes (or similar products, such as RedHat's OpenShift) may be an option for deploying and orchestrating high availability for Ubersmith. Unfortunately, we do not have any guidance on deploying Ubersmith on Kubernetes at this time, but know of no technical reason why it would not be possible.?

References

[DRBD](#) allows mirroring of filesystem block devices over the network.

[HAProxy](#) offers high availability, load balancing, and proxying for TCP and HTTP-based applications.

[keepalived](#) provides facilities for loadbalancing and high-availability to Linux based infrastructures.

[Kubernetes](#) is an open-source system for automating deployment, scaling, and management of containerized applications.

[RRDtool](#) is an open source data logging and graphing system for time series data.

[Percona XtraDB Cluster](#) is an active/active high availability open source solution for MySQL clustering.

[ProxySQL](#) is a high-performance MySQL proxy.

Related Topics

[Clustering Databases](#)