MathOps.Dev Server Design

The MathOps system is designed to be deployed on a bank of servers that provide availability in the event of any one server failing, and to support live updates of production applications without downtime. This allows individual servers to be patched and upgraded without downtime.

Servers may be deployed in virtual machines under the control of a hypervisor, but there should be at least 2 (and preferably 3) physical machines so that one can be taken offline for maintenance without a loss of availability of the production system.

This document assumes servers run Debian Linux 12.2. It will be updated from time to time if this changes.

Server OS installation and setup should follow this outline:

* Normal install from ISO image, but add “SSH Server” in software selection.
* apt update / apt upgrade (as root)
* if running in a VirtualBox environment, install “guest additions” (as root)
* apt install keepalived (as root)
* apt install docker (as root)
* apt install docker-compose (as root)
* apt install glusterfs-server (as root)

Network Design

Servers can operate with a single NIC attached to a single public network, but it will improve performance if there are two NICs, one attached to the normal, public network, and the second attached to a private network. This way, synchronization and management traffic can occur over the private nertwork, and neither network will affect the other’s traffic.

Suppose there are **M** physical servers, which we will label **Server 1** through **Server M**. These servers may be running hypervisors, in which case the number of hosts could be greater than the number of physical servers. Suppose there are **N** Debian hosts, which we will label **Host 1** through **Host N**.

Each host will have its own dedicated IP address on the public network, and (if there is a private network), a dedicated IP address on that network as well.

Load-Balanced Front-End Design

All servers will run **keepalived**, listening on a common “well-known” IP address on the public network, while interconnecting with each other on the private network. This daemon will also monitor the “httpd” process for failures and fail-over to backup servers when the primary httpd front-end goes down.

All servers will run **glusterFS** over the private network to share a directory structure that includes static web pages, images, closed-caption files, etc. One machine will be the primary server, and will distribute file updates to clients. If the primary goes down, the clients have copies to use in the interim, but could not accept automated updates until the primary is running again (or until some other machine is made primary).

All servers will run a **Docker** container with an instance of **Apache httpd** with TLS certificates and a **Shibboleth** service provider, listening on the “well-known” IP address on the primary network using a Docker user-defined bridge network that exposes only ports 80 and 443. All instances can serve static files from the glusterFS filesystem, or can forward service requests to any of the registered web services. These web server instances are stateless – any instance can serve any request. Keepalived could route all traffic to the primary and keep the others as hot backups, or could load-balance traffic to all servers.

Shared Data Services

Since any server could be asked to field any request, state must be shared across servers. This includes long-term state such as database tables, as well as short-term state like login sessions or cached query data.

All servers will run a **Docker** container with an instance of the **Apache Cassandra** replicated database, with replication factors that allow at least one (and preferably more than one) server to fail without loss of data or service. This will store long-term data. This container will only have access to the private network. Any web service may access long-term data from this data store.

All servers will run a **Docker** container with an instance of the **Redis** in-memory key-value database, with replication factors that allow at least one (and preferably more than one) server to fail without loss of data or service. This will store short-term data like login sessions, assessment sessions, and cached user-specific data queried to serve requests. This container will only have access to the private network. Any web service may access short-term data through this data store.

Web Services

Every web site, web service, or websocket service provided within the system will be deployed as a separate service within a **Docker Swarm** on all servers. Each will listen on its own TCP port, to which the Apache HTTPD front-end will forward traffic by reverse proxy to serve requests. These services will only be accessible from the private network.

The **WeBWorK** web services requires its own MySQL database. This can either be deployed on a single server, or MySQL replication can be set up to allow it to run across all servers.

In addition commercial products can also be deployed on distinct TCP ports and served through the front-end. For example:

* **Mantis** issue tracking
* **Grafana** log aggregation
* **Prometheus** metrics collection
* **Nagios** monitoring
* **TestContainers** integration test automation
* **Flow.CI** continuous integration