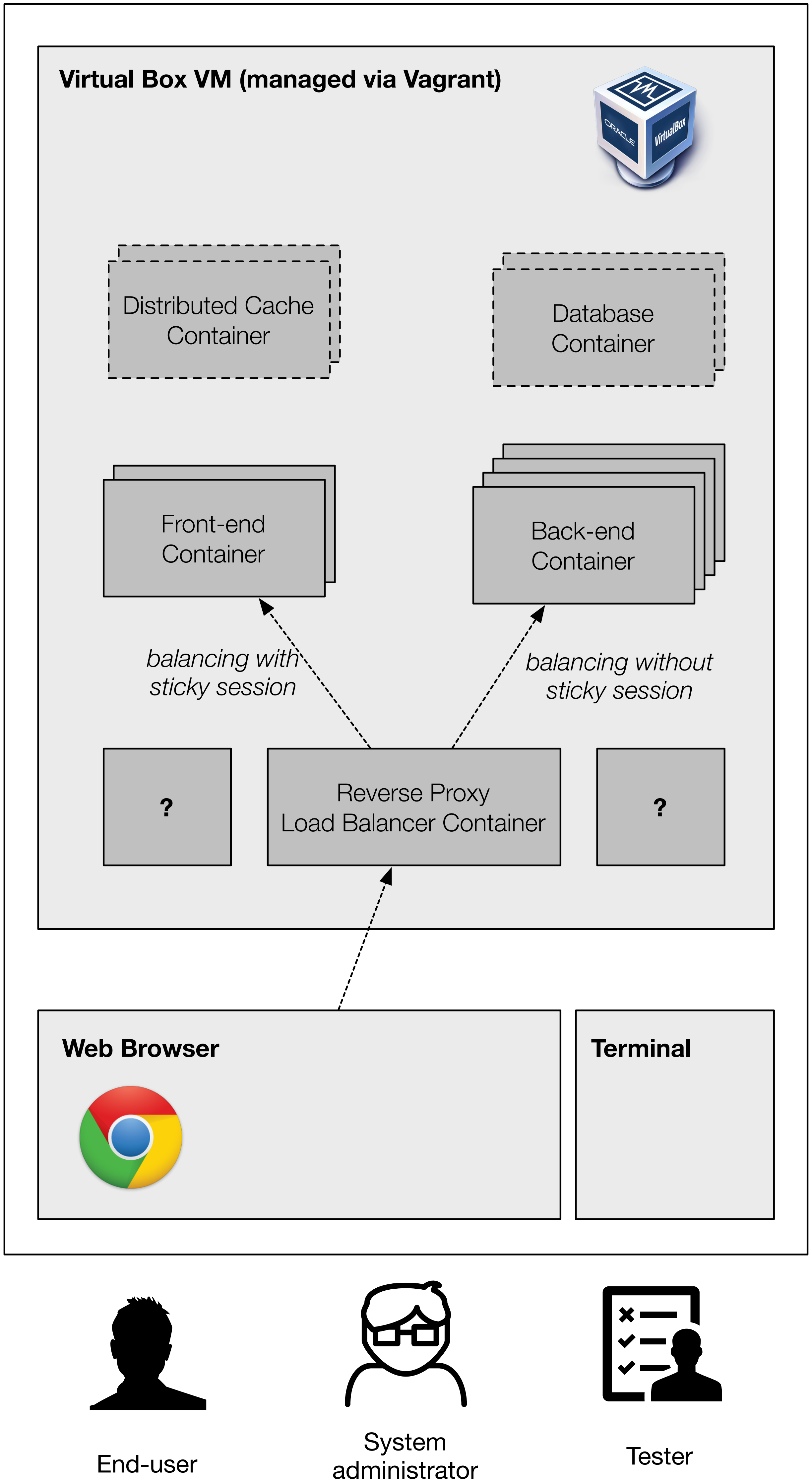
Virtual Box VM (managed via Vagrant)
Distributed Cache Container
Database Container
Front-end Container
Back-end Container
balancing with sticky session
balancing without sticky session
?
Reverse Proxy Load Balancer Container
?
Web Browser
Terminal
End-user
System administrator
Tester

Requirements

As a user, I can access a Web UI. The static content is served by one of the front-end containers. A Javascript script running in the browser makes AJAX calls to fetch dynamic data served by one of the back-end containers.

As a sysadmin, I can use a web browser to see the list of running Docker containers and see from which image they were created.

As a sysadmin, I can use a web browser to start and stop front-end and back-end containers.

As a tester, I can prove that HTTP requests are load-balanced between the back-end containers and that all containers receive requests to process.

As a tester, I can prove that load balancing is configured in a way that sticky sessions are enabled on the front-end containers.

As a tester, I can prove that load balancing is configured in a way that sticky sessions are disabled on the back-end containers.

As a tester, I can prove that when a new container (front-end or back-end) is started, the load balancer is reconfigured and starts sending some of the requests to it.

As a tester, I can prove that when a container is stopped (front-end or back-end), the load balancer is reconfigured and stops sending requests to it.

As a tester, I can prove that the dynamic reconfiguration of the load balancer uses a custom protocol based on UDP. I can provide the documentation of this protocol.

Constraints

The front-end containers MUST use the apache httpd server and PHP.

The back-end containers MUST serve JSON resource representations under the /api/ end-point.

The distributed cache and database containers are not mandatory. They MAY use technologies such as redis, memcached, mysql or mongodb.

The resources managed by the back-end containers must have a dynamic state, so that successive requests produce different representations.