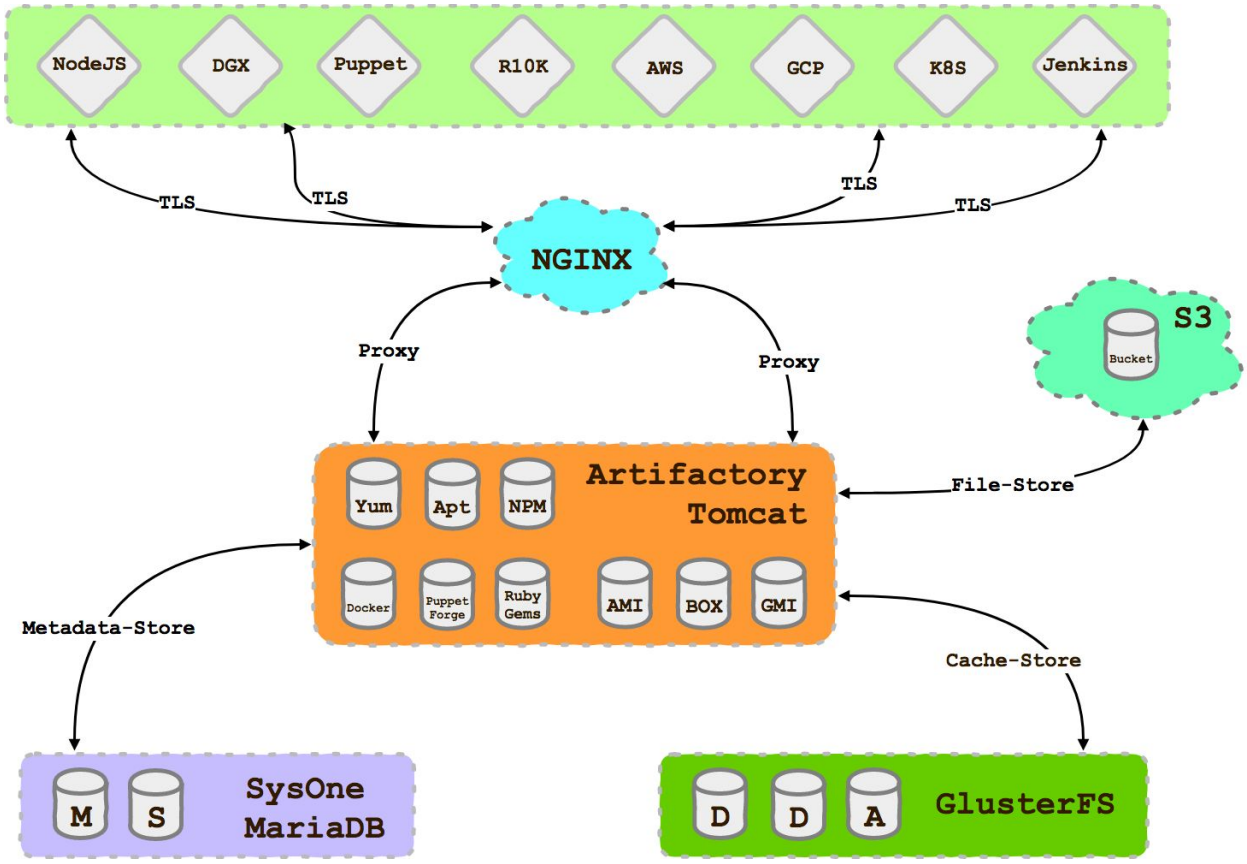
Artifactory Architecture

Overview

Artifactory architecture is rather simple because this is the first iteration and does not have HA. The Enterprise license which supports HA is cost prohibitive at this time so the best we can do it design an infrastructure that would have fast recovery times at every part of the stack.

Diagram



Stack and Recovery

Artifactory Unit: Artifactory Unit node is a single instance of artifactory tomcat service where artifacts are can be distributed to the rest of the nodes in the Infrastructure. It supports various types of packages; Yum, Apt, NPM, Docker, JARs, RubyGems, AMI, Gcloud, Puppet Forge, etc. If this node goes offline its as simple as spinning up another node in Foreman, its all Puppetized.

S3 File-Store: While the metadata is stored in SQL the actual packages are stored in AWS S3 because, well, SQL cannot handle such large objects and it will be very slow to retrieve them. The actual file contents are stored in S3 Bucket which can scale to infinity (but probably not infinity).

GlusterFS Cache-Store: A GlusterFS volume is mounted onto the Artifactory VM, this volume acts as cache for the artifacts since S3 is a remote file-store. GlusterFS itself is highly-available as it has 2 data bearing and 1 arbiter node, in case of failure the other node service the file contents.

MariaDB Metadata-Store: The backend used by Artifactory to store in metadata related to artifacts. We decided to use MariaDB because other parts of our infrastructure already uses it, we are familiar with. It's been setup external to the Artifactory Unit node in a Master/Slave HA model so when a failure does happen on the Master we can switch to the Slave in about 3-5 mins.

Nginx Reverse Proxy: Nginx is good with handling connections, caching and of course TLS termination. While Tomcat supports all this we should take this additional load off of it and let Nginx be in front of it handling HTTP requests, TLS termination and load balancing connections.