Modernizing Monolithic Applications at a Financial Services Company
Summary

A large Financial Services and Banking organization with a large legacy infrastructure engaged Shadow-Soft to aid with the transition from a monolithic code base to a microservices application architecture. Critical to the effort was to seed the next generation of applications and enable future development with lessons learned. After the customer engaged with our consulting team, defined the current state & long-term objectives, Docker was selected to abstract pieces of a Java based trade routing application. We also provided a foundational understanding of the platform for the client.

Defined Engagement Objectives:

- Prove enhanced speed of adoption and testing
- Determine the feasibility of using Docker and micro-services
- Confirm compatibility with their high-speed compute infrastructure
- Migrate an “all-in-one” application into a split infrastructure on Docker and bare-metal

Challenge

The client’s internal cloud had machines with a kernel version suitable for all of the features offered by Docker at the time. Using these nodes as a base infrastructure, network latency testing, host to host, container to host, and container to container testing were performed. These cloud-nodes also served as the core development infrastructure of a Docker Registry (w/ an NGINX reverse proxy), GOGS (a Docker based SCM), and Drone (a Docker based CI tool). Some physical hardware with AHA/Mellanox cards were also used. On these machines, we again tested network latency and developed a solution so the Docker containers would load special hardware libraries to utilize the compression cards and the high speed NIC’s effectively.
To confirm these assumptions, Shadow-Soft built a Docker image w/ the latest version of the client’s application code and linked that to two other containers running KDB, one running a live data-set and another that contained the configuration settings. Using Docker Compose, an auditing application was stood up and began running jobs in a matter of seconds. This confirmed the ability to rapidly prototype application changes against existing data.

Shadow-Soft also began prototyping using Docker as an alternative to the Vagrant/Ant based development machines currently being used. To achieve this, long bash install scripts were developed and deployed to convert these to a Docker-file that produced an image very close to what the client’s developers were using. This allowed the client to spin up new development machines w/ Docker, cutting deployment time by 50%.

To complete phase 1, the components of the application, which weren’t designed to run standalone (i.e. not crash when they can’t find some other piece) were successfully migrated. To achieve this complex task, the team had to design entry point scripts that performed checks, before allowing the application inside the container to start. They also had to be careful with managing the state of the process in the container, and how processes were handled when using these entry point scripts, to avoid issues.

Based on the success and lessons learned during the initial phase of the project, it was decided to base the second phase of the POC on Swarm and Compose. Docker Swarm allowed us to create and access to a pool of Docker hosts using the full suite of Docker tools. Because Docker Swarm follows the “swap, plug, and play” principle, experience using this mechanism will easily translate to a large scale production deployment using more powerful back-ends like Mesos & Kubernetes. Similarly, using Docker Compose allowed us to orchestrate the deployment of the application across our Swarm Cluster.

In addition, during the phase 2, we dealt with orchestration & networking. Also, breaking out more components of the application and performed best-practices: specifically around security/patching, logging, and monitoring.
Shadow-Soft additionally built a Swarm cluster using internal cloud nodes as well as the physical hardware nodes. The application depended heavily on the exact location of certain components, so labels were used to restrict where containers ran to ensure they started in the proper order. In order to confirm success, we used Docker Compose (with a highly customized docker-compose.yml) and bash scripts to bring up and down the application. This exercise demonstrated that the application could be brought up, shut down, and restarted on different nodes very quickly.

Conclusion

At close of the engagement, the application was successfully split into 7 separate pieces and deployed on Docker. Using a “real” application, coupled with extensive knowledge transfer to the client, allowed the client a new way to view their legacy code and start breaking up their monolithic applications into smaller (but maybe not micro) services. Finally, a white paper was produced detailing the engagement & discoveries made for the purpose of seeding the next generation of their application.

Call our consultants and discover the right solution for your business:

770-546-0077

shadow-soft.com or email contact@shadow-soft.com

Since 2008, Shadow-Soft has been evangelizing and deploying open source software and open standards to help customers “take the power back” from their technology vendors. Shadow-Soft provides consulting and managed services across three specialties: DevOps, Application Infrastructure, and Cloud.