Customers

Read the full blog post by Anders Fredriksson, Co-founder & CEO at ProLeads

Background
ProLeads helps enterprises target qualified leads with remarkably personalized e-mails through its SaaS platform

Application Stack
The application consists of Ruby along with Redis and MongoDB Replica Sets

Stage
PRODUCTION & DEVELOPMENT

Challenges:

- Existing container application composition frameworks do not address complex dependencies, high-availability or auto-scaling workflows post-provision.
- Struggling to achieve their core objectives with containers (1) high-availability in production, and (2) updating production environment with zero downtime.
- Setting up development environments is time consuming -- especially as the company continues to on-board new software engineers

Solution:

- DCHQ is expected to **save ProLeads more than $150k over two years** by making the switch from Heroku to DCHQ
- Used DCHQ to automate the building of the Ruby image from a private GitHub project. Two image builds were scheduled on DCHQ: one that creates an image with the time-stamp in the tag name using {{timestamp}} for backup, and another that creates an image with the tag latest (i.e. always overriding the latest image with the latest code commits).
- DCHQ simplified the containerization of their Ruby application allowing ProLeads to cluster their Ruby containers for high-availability, automatically configure Nginx for load balancing through BASH script plug-ins, and connect to existing services like pre-configured Rackspace Stack Templates.
- DCHQ allowed ProLeads to address their top priorities in production -- i.e. (1) achieving high-availability, and (2) pushing updates seamlessly and with zero downtime.
- DCHQ allowed ProLeads to better manage their live application in production with access to monitoring for the running containers, along with alerts and notifications. If the application becomes resource constrained, ProLeads can use DCHQ to automatically scale out the Ruby cluster.

DURAARK is funded by the European Commission within the 7th Framework Programme (Grant Agreement no: 600908)

Read the full blog post by Martin Hecher, a research assistant and software architect at Fraunhofer Austria Research GmbH

Background
DURAARK.eu is an open source system to do semantic archival and retrieval of architectural data. The tool helps stakeholders from the architecture, engineering and construction (AEC) community to manage data like 3D models, point cloud scans or context information over the lifecycle of a building.

Application Stack
DURAARK relies on a micro-services architecture using a set of components written in different programming languages, including Java, Python, C++ and Javascript.

Stage
PRODUCTION & DEVELOPMENT

Challenges:

- Interested in using a hosted service where stakeholders can preview the DURAARK prototype on their own.
- Existing container application composition frameworks do not address complex dependencies for the micro-service architecture.
- Stakeholders do not have the expertise to deploy multi-container web applications on their own.

Solution:

- DCHQ allowed DURAARK to **model and deploy their site in a matter of minutes**
- DCHQ simplified the containerization of their micro-services architecture allowing DURAARK to automatically configure Nginx for load balancing through BASH script plug-ins and use cross-image environment variable bindings to model their complex dependencies
- Stakeholders can now request DURAARK from the self-service library and deploy the prototype using a one-click deploy button
- DCHQ allowed DURAARK to better manage their live application in production with access to monitoring for the running containers, along with alerts and notifications.

Background

Streaming Networks builds security and surveillance systems. Their flagship iRecord® License Plate Recognition (LPR-100) is a stand-alone, network enabled, high performance, License Plate Recognition (LPR) system for law enforcement applications, covert operations, toll collection, parking management and vehicle access control.

Application Stack

The iRecord® License Plate Recognition management application is written in Java. The architecture includes GlassFish, MySQL, and ApacheDS.

Stage

PRODUCTION & DEVELOPMENT

Challenges:

- Delivering this application to customers (like law enforcement agencies) requires manual configuration and a lot of hand-holding. An easy deployment on different cloud environments is needed.
- Running POC's required a lot of time and effort by pre-sales engineers
- Setting up development environments is time consuming -- especially as the company continues to on-board new software engineers

Solution:

- DCHQ helped Streaming Networks create an application template that can be deployed on any Linux host running on any cloud. This template **allows customers to deploy the LPR management application in a matter of seconds** -- regardless of the cloud provider they are using.
- 80% reduction in the amount of time it takes to set up development environments and POC's.
- DCHQ allowed Streaming Networks to **move to containers without making any changes to their development processes**. Using DCHQ's continuous delivery workflows, Streaming Networks can deploy the latest Java WAR file from Jenkins on running GlassFish app server containers in DEV/TEST environments. Read more about this **here**.
- DCHQ allowed Streaming Networks to better manage their live applications in production with access to monitoring for the running containers, along with alerts and notifications. Additionally, the application backup workflows allow Streaming Networks to roll back to older images if needed

Background
MarkaVIP is the Leading Online Shopping Community in the Middle East

Application Stack
The application consists of Nginx, Magento and MySQL.

Stage
DEVELOPMENT (moving to PRODUCTION soon)

Challenges:

- Existing container application composition frameworks do not address complex dependencies, high-availability or auto-scaling workflows post-provision.
- Setting up development environments is time consuming -- especially as the company continues to on-board new software engineers

Solution:

- Used DCHQ to automate the building of the Nginx, PHP and MySQL images from a private GitHub project.
- DCHQ simplified the containerization of their Magento application allowing MarkaVIP to automatically configure Nginx for load balancing through BASH script plug-ins and leverage environment variable bindings to model the complex dependencies.
- DCHQ allowed MarkaVIP to optimize server utilization by running several instances of the same application on the same host -- without running into container name or port binding conflicts.
- DCHQ allowed MarkaVIP to better manage their live applications in development with access to monitoring to correlate issues to container updates or build deployments. Moreover, DCHQ's application sharing capability allowed developers to collaborate more closely to debug issues on their deployed applications in DEV.

Background
Project Ricochet provides cutting-edge web development and design services. They specialize in Drupal Development and lightweight Node JS frameworks like Meteor, with an emphasis on responsive mobile design.

Application Stack
The application consists of Nginx, Meteor, and MongoDB

Stage
DEVELOPMENT (moving to PRODUCTION soon)

Challenges:

- Struggling to automate the provisioning of Meteor apps for their customers -- while optimizing server utilization
- Existing container application composition frameworks do not address complex dependencies, high-availability or auto-scaling workflows post-provision.
- Looking for an easy way to push updates in production with minimal downtime

Solution:

- Used DCHQ to automate the building of the Meteor images from a private GitHub project.
- DCHQ simplified the containerization of their Meteor application allowing Project Ricochet to automatically configure Nginx for load balancing through BASH script plug-ins and leverage environment variable bindings to model the complex dependencies.
- DCHQ allowed Project Ricochet to optimize server utilization by running several instances of the same application on the same host -- without running into container name or port binding conflicts.
- DCHQ allowed Project Ricochet to better manage their live applications in development with access to monitoring for the running containers, along with alerts and notifications. If the application becomes resource constrained, Project Ricochet can use DCHQ to automatically scale out the Meteor cluster.
- DCHQ's REST API's will be used to automate plug-in executions programmatically for running containers.