Docker Containers – For Infrastructure Optimization

by

Matt Allen and Kevin Biggers

Submitted to
The Faculty of the School of Information Technology
in Partial Fulfillment of the Requirements for
the Degree of Bachelor of Science
in Information Technology

Matt Allen and Kevin Biggers — Date: 04/17/2017

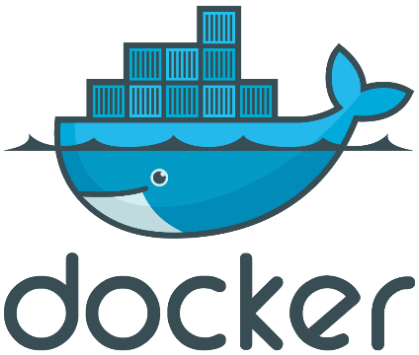
Brian Verkamp — Date: 04/17/2017

University of Cincinnati
College of
Education, Criminal Justice, and Human Services

Docker Containers

For

Infrastructure Optimization



Prepared by:

Matt Allen and Kevin Biggers

Students of

University of Cincinnati

College of Education, Criminal Justice, and Human Services

School of Information Technology cech.uc.edu/it

17 October 2016

CONTENTS

Chapter Page

TABLES

No. Page

FIGURES

No. Page

1.ABSTRACT

Traditional Virtual Environments have several overhead costs including physical hardware, hypervisor licensing, and other management tools. Docker containers provide a solution that compartmentalizes applications on one virtual machine which reduces the overall reoccurring infrastructure costs associated with managing multiple machines at once in a traditional environment. Docker Containers appeals to Systems Administrators by providing faster configurations and allowing them to run multiple applications on a single host. Developers are provided with a faster paced development environment due to the simplicity and speed of Dockers deployment capabilities. Docker's infrastructure allows containers to be scanned for vulnerabilities, patched more frequently, and allow more secure segregation. Utilizing Docker containers delivers a solution that provides an organization with a secure test and production environment that optimizes infrastructure utilization and decrease costs.

2.PROBLEM STATEMENT

2.1 Introduction

Docker containers provides a solution that compartmentalizes applications on one virtual machine which reduces the overall reoccurring infrastructures costs of managing multiple machines by reducing the amount of storage needed, and eliminating hypervisor licensing costs. Docker will create a more secure environment by consolidating the applications on a single machine creating less vulnerabilities and allow more frequent patching. This solution will allow a systems administrator to run more applications on a single machine and eliminate virtualized hardware. The top three benefits emphasized in this project are as follows: One, Dockers ability to provide simplicity and faster configurations due to its ability to allow a user to run any platform with its own configuration atop the infrastructure of a user without the overhead of a VM. Two, Docker increases productivity by creating a faster development environment allowing more interactive use. Three, Dockers rapid deployment reduces deployment time to mere seconds due to its ability to create a container for every process and that it does not boot an Operating System. Having a low cost of bringing up new instances allows resources to be allocated elsewhere.

2.2 Project Description

Deliver a solution that will provide a secure environment for application development and production on a single Operating System utilizing Docker containers. This will allow enterprises to optimize infrastructure utilization and decrease the cost of maintaining existing apps. Integrating Docker Containers will allow container's images to be patched more frequently as part of the application deployment process, which decreases the environments security risks. Containers also allows easier segregation of applications that would traditionally run directly on the same host.

2.3 Problem

Traditional VMs include guest operating systems within each machine which utilizes valuable storage capacity. Docker containers are lightweight and only include what is necessary to run the applications. Docker will provide a solution that allows enterprises to leverage Docker containers across any infrastructure or application environment types.

2.4 User Profile

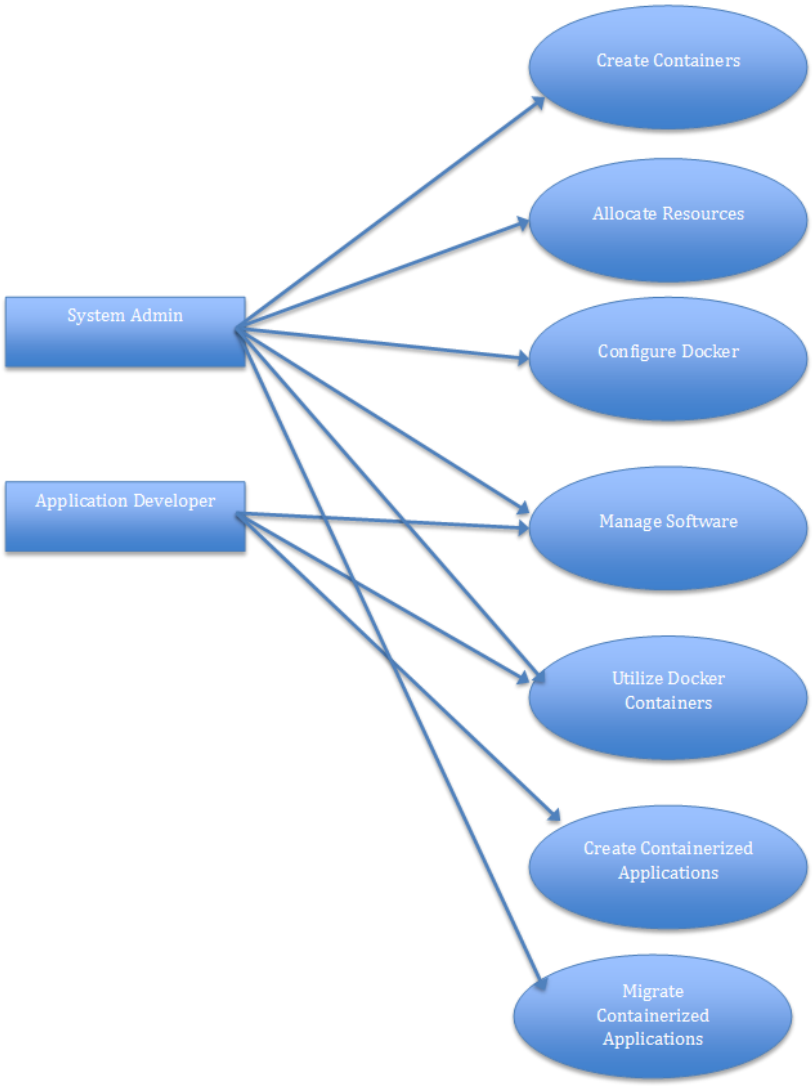
The potential users for our Docker Containers solution are system administrators as well as application managers/developers. The system administrators will use the solution to consolidate IT resources and will be in charge of the management and implementation of the Docker tool. Application developers/managers will use the tool to containerize their different applications on a single machine. This will give applications better portability as well as provide more stable test and production environments. The users are depicted in detail in Table 1 below.

Table 1. User Profile

2.4.1 Potential Users • System Administrators • Applications Developers/Managers
2.4.2 Software and Interface Experience • The system administrator user should have experience with the Linux operating system including using the Linux command line • The application developer/manager should know how to interact with the Docker user interface
2.4.3 Experience with Similar Applications • Containerizing applications is a relatively new technology without much commercialization so far. We do not expect users to have much similar experience.
2.4.4 Task Experience • The application developers will be comfortable with using the Docker user interface and they will be able to create, test and deploy applications faster. • The System Admins will manage the back-end of the Docker tool including but not limited to: installing the software, setting up the different containers for the developers and allocating resources for the different applications.
2.4.5 Frequency of Use • This solution will be used on a daily basis both by system administrators and application developers.
2.4.6 Key Interface Design Requirements That the Profile Suggests • Ability to use the web GUI to manage the different applications and containers • Applications must still run at a level at which they can still perform their basic functions.

The Use Case Diagram below demonstrates how Systems Administrators and Application Developers can utilize Docker Containers. A Systems Admin has the ability to create containers, allocate resources, configure the Docker system, manage software on the containers, utilize the Docker containers, and migrate containerized applications between hosts. An Application developer can use Docker to manage the software on the containers, utilize the Docker containers, and create containerized applications on the host. Figure 1 depicts a Use Case Diagram for Systems Administrators and Application Developers.

Figure 1. Docker Use Case Diagram



3.PROJECT MANAGEMENT

3.1 Budget

Table 2 displays the project budget. The numbers presented are estimated costs if implemented in a real world environment with one running Node and one AWS Server. The cost of labor takes the average salary for a Systems Administrator of $75,790/yr or $36.44/hr and multiplies that by an estimate of 50 labor hours that it would take to build this system. This does not reflect our senior design project as we are using a Cloud Solution provided by the University of Cincinnati.

Table 2. Project Budget

No.	Item	Unit, Each	Unit Price, $	Line Item Total
1	Amazon Web Services t2.medium Server	1	$302.00/yr	$302.00/yr
2	Docker Cloud Managed Nodes License	1	$180.00/yr	$180.00/yr
3	Labor	50	$36.44/hr	$1,822.00
Totals				$2,304/yr

3.2 Objectives/Deliverables

The project deliverables and deadlines are presented in **Table 3**.

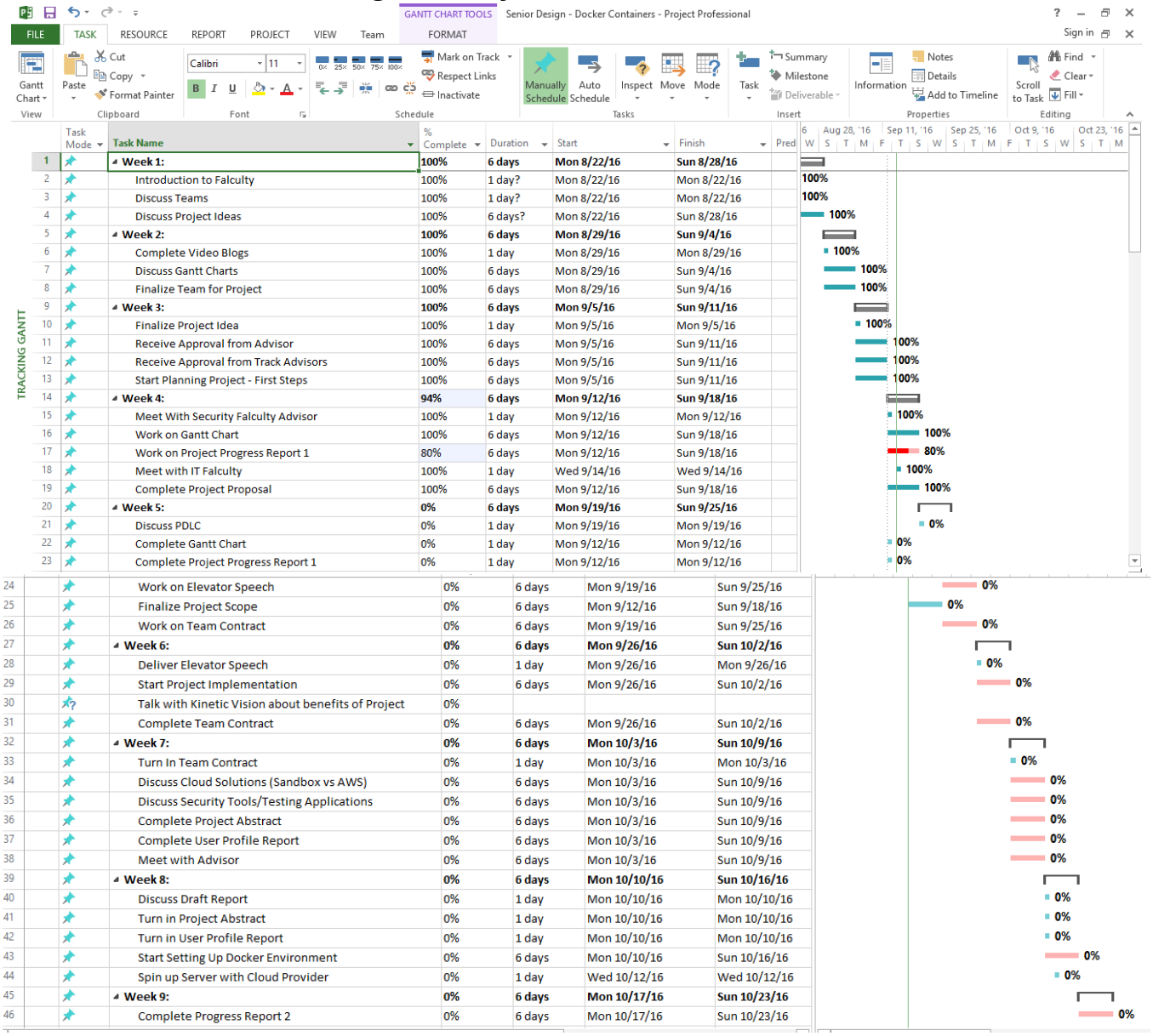
Table 3. Project Objectives/Deliverables Due Dates

Major Project Milestones (Deliverables)				
Finalize Project Objectives	9/12/2016		Stack Setup Milestone	10/31/2016
Pre Setup Planning	9/26/2016		Security Setup Milestone	11/07/2016
Docker Cloud Setup	10/17/2016		Presentation Milestone	11/28/2016
Docker Poster Design	03/06/2017		Tech Expo	04/11/2017

3.3 Project Schedule

Figure 2 depicts the project deliverable deadlines

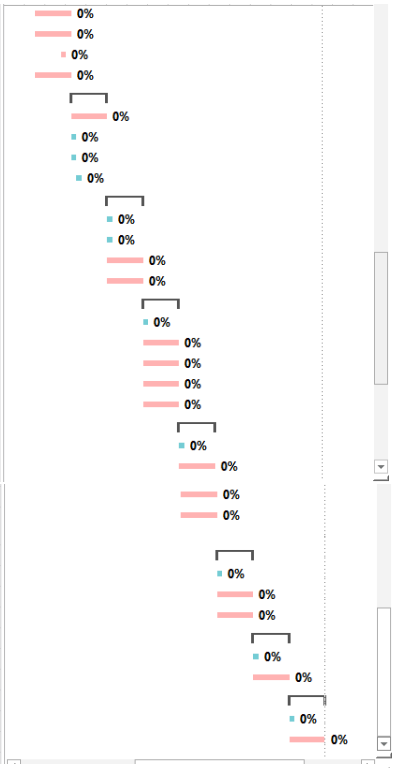
Figure 2. Project Schedule Gantt Chart



	Task Mode	Task Name	% Complete	Duration	Start	Finish	Pred
1		**Week 1:**	**100%**	**6 days**	**Mon 8/22/16**	**Sun 8/28/16**	
2		Introduction to Falculty	100%	1 day?	Mon 8/22/16	Mon 8/22/16	
3		Discuss Teams	100%	1 day?	Mon 8/22/16	Mon 8/22/16	
4		Discuss Project Ideas	100%	6 days?	Mon 8/22/16	Sun 8/28/16	
5		**Week 2:**	**100%**	**6 days**	**Mon 8/29/16**	**Sun 9/4/16**	
6		Complete Video Blogs	100%	1 day	Mon 8/29/16	Mon 8/29/16	
7		Discuss Gantt Charts	100%	6 days	Mon 8/29/16	Sun 9/4/16	
8		Finalize Team for Project	100%	6 days	Mon 8/29/16	Sun 9/4/16	
9		**Week 3:**	**100%**	**6 days**	**Mon 9/5/16**	**Sun 9/11/16**	
10		Finalize Project Idea	100%	1 day	Mon 9/5/16	Mon 9/5/16	
11		Receive Approval from Advisor	100%	6 days	Mon 9/5/16	Sun 9/11/16	
12		Receive Approval from Track Advisors	100%	6 days	Mon 9/5/16	Sun 9/11/16	
13		Start Planning Project - First Steps	100%	6 days	Mon 9/5/16	Sun 9/11/16	
14		**Week 4:**	**94%**	**6 days**	**Mon 9/12/16**	**Sun 9/18/16**	
15		Meet With Security Falculty Advisor	100%	1 day	Mon 9/12/16	Mon 9/12/16	
16		Work on Gantt Chart	100%	6 days	Mon 9/12/16	Sun 9/18/16	
17		Work on Project Progress Report 1	80%	6 days	Mon 9/12/16	Sun 9/18/16	
18		Meet with IT Falculty	100%	1 day	Wed 9/14/16	Wed 9/14/16	
19		Complete Project Proposal	100%	6 days	Mon 9/12/16	Sun 9/18/16	
20		**Week 5:**	**0%**	**6 days**	**Mon 9/19/16**	**Sun 9/25/16**	
21		Discuss PDLC	0%	1 day	Mon 9/19/16	Mon 9/19/16	
22		Complete Gantt Chart	0%	1 day	Mon 9/12/16	Mon 9/12/16	
23		Complete Project Progress Report 1	0%	1 day	Mon 9/12/16	Mon 9/12/16	
24		Work on Elevator Speech	0%	6 days	Mon 9/19/16	Sun 9/25/16	
25		Finalize Project Scope	0%	6 days	Mon 9/12/16	Sun 9/18/16	
26		Work on Team Contract	0%	6 days	Mon 9/19/16	Sun 9/25/16	
27		**Week 6:**	**0%**	**6 days**	**Mon 9/26/16**	**Sun 10/2/16**	
28		Deliver Elevator Speech	0%	1 day	Mon 9/26/16	Mon 9/26/16	
29		Start Project Implementation	0%	6 days	Mon 9/26/16	Sun 10/2/16	
30		Talk with Kinetic Vision about benefits of Project	0%				
31		Complete Team Contract	0%	6 days	Mon 9/26/16	Sun 10/2/16	
32		**Week 7:**	**0%**	**6 days**	**Mon 10/3/16**	**Sun 10/9/16**	
33		Turn In Team Contract	0%	1 day	Mon 10/3/16	Mon 10/3/16	
34		Discuss Cloud Solutions (Sandbox vs AWS)	0%	6 days	Mon 10/3/16	Sun 10/9/16	
35		Discuss Security Tools/Testing Applications	0%	6 days	Mon 10/3/16	Sun 10/9/16	
36		Complete Project Abstract	0%	6 days	Mon 10/3/16	Sun 10/9/16	
37		Complete User Profile Report	0%	6 days	Mon 10/3/16	Sun 10/9/16	
38		Meet with Advisor	0%	6 days	Mon 10/3/16	Sun 10/9/16	
39		**Week 8:**	**0%**	**6 days**	**Mon 10/10/16**	**Sun 10/16/16**	
40		Discuss Draft Report	0%	1 day	Mon 10/10/16	Mon 10/10/16	
41		Turn in Project Abstract	0%	1 day	Mon 10/10/16	Mon 10/10/16	
42		Turn in User Profile Report	0%	1 day	Mon 10/10/16	Mon 10/10/16	
43		Start Setting Up Docker Environment	0%	6 days	Mon 10/10/16	Sun 10/16/16	
44		Spin up Server with Cloud Provider	0%	1 day	Wed 10/12/16	Wed 10/12/16	
45		**Week 9:**	**0%**	**6 days**	**Mon 10/17/16**	**Sun 10/23/16**	
46		Complete Progress Report 2	0%	6 days	Mon 10/17/16	Sun 10/23/16	

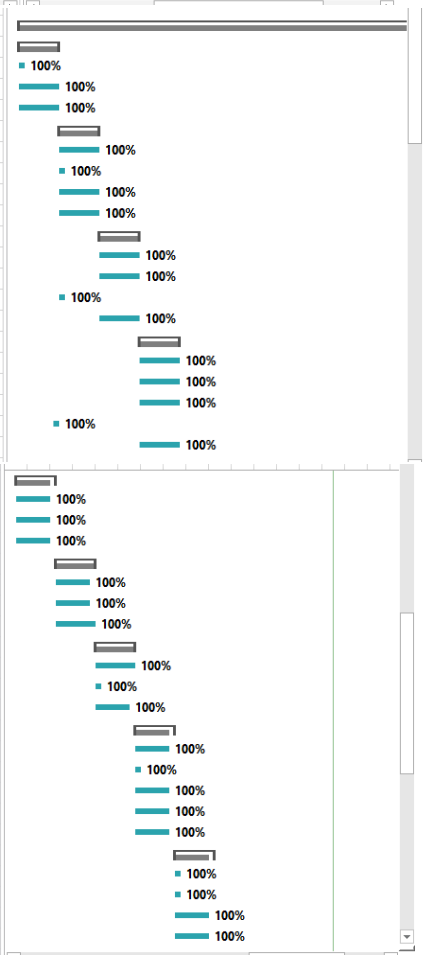
TRACKING GANTT

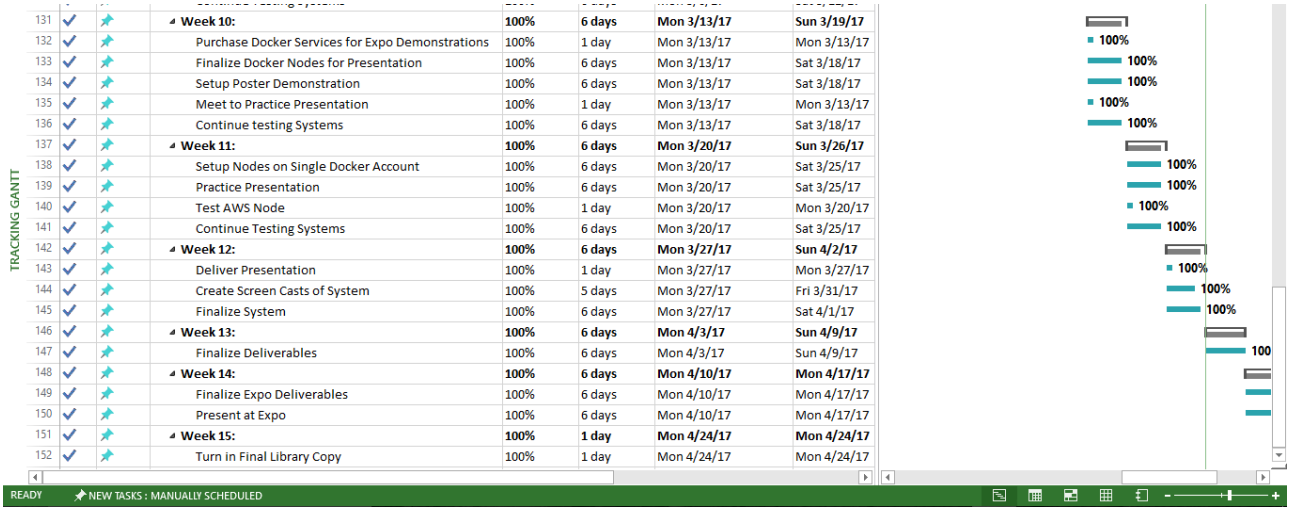
47		Complete Use Case Diagram	0%	6 days	Mon 10/17/16	Sun 10/23/16
48		Work on Draft Report	0%	6 days	Mon 10/17/16	Sun 10/23/16
49		Discuss Budgets	0%	1 day	Sat 10/22/16	Sat 10/22/16
50		Continue working on Project Implementation	0%	6 days	Mon 10/17/16	Sun 10/23/16
51		**Week 10:**	**0%**	**6 days**	**Mon 10/24/16**	**Sun 10/30/16**
52		Complete Draft Report	0%	6 days	Mon 10/24/16	Sun 10/30/16
53		Turn in Progress Report 2	0%	1 day	Mon 10/24/16	Mon 10/24/16
54		Turn in Use Case Diagram	0%	1 day	Mon 10/24/16	Mon 10/24/16
55		Meet with Technical Advisors	0%	1 day	Tue 10/25/16	Tue 10/25/16
56		**Week 11:**	**0%**	**6 days**	**Mon 10/31/16**	**Sun 11/6/16**
57		Discuss Fall Presentations	0%	1 day	Mon 10/31/16	Mon 10/31/16
58		Turn in Draft Report	0%	1 day	Mon 10/31/16	Mon 10/31/16
59		Work on Fall Final Presentation	0%	6 days	Mon 10/31/16	Sun 11/6/16
60		Continue building Environment for Docker Containers	0%	6 days	Mon 10/31/16	Sun 11/6/16
61		**Week 12:**	**0%**	**6 days**	**Mon 11/7/16**	**Sun 11/13/16**
62		Discuss Spring Semester Goals	0%	1 day	Mon 11/7/16	Mon 11/7/16
63		Update Gantt Chart	0%	6 days	Mon 11/7/16	Sun 11/13/16
64		Update Project Plans	0%	6 days	Mon 11/7/16	Sun 11/13/16
65		Contine Project Implementation	0%	6 days	Mon 11/7/16	Sun 11/13/16
66		Work on Fall Final Presentation	0%	6 days	Mon 11/7/16	Sun 11/13/16
67		**Week 13:**	**0%**	**6 days**	**Mon 11/14/16**	**Sun 11/20/16**
68		Finalize Fall Presentation	0%	1 day	Mon 11/14/16	Mon 11/14/16
69		Finalize Security Tool	0%	6 days	Mon 11/14/16	Sun 11/20/16
70		Build out Containers	0%	6 days	Mon 11/14/16	Sun 11/20/16
71		Work on setting up Security Tool/Test Applications in environment	0%	6 days	Mon 11/14/16	Sun 11/20/16
72		**Week 14:**	**0%**	**6 days**	**Mon 11/21/16**	**Sun 11/27/16**
73		Complete Fall Final Presentation	0%	1 day	Mon 11/21/16	Mon 11/21/16
74		Start Testing Security Tool	0%	6 days	Mon 11/21/16	Sun 11/27/16
75		Build out more Containers for testing	0%	6 days	Mon 11/21/16	Sun 11/27/16
76		**Week 15:**	**0%**	**6 days**	**Mon 11/28/16**	**Sun 12/4/16**
77		Fall Final Presentations	0%	1 day	Mon 11/28/16	Mon 11/28/16
78		Continue Testing Docker Environment	0%	6 days	Mon 11/28/16	Sun 12/4/16
79		**Week 16:**	**0%**	**6 days**	**Mon 12/5/16**	**Sun 12/11/16**
80		Fall Final Presentations	0%	1 day	Mon 12/5/16	Mon 12/5/16
81		Continue Testing Docker Environment	0%	6 days	Mon 12/5/16	Sun 12/11/16



TRACKING GANTT

87	✓	**Spring Semester**	**100%**	**79 days**	**Mon 1/9/17**	**Thu 4/27/17**
88	✓	**Week 1:**	**100%**	**6 days**	**Mon 1/9/17**	**Sun 1/15/17**
89	✓	Group Meetup	100%	1 day	Mon 1/9/17	Mon 1/9/17
90	✓	First Senior Design Class	100%	6 days	Mon 1/9/17	Sun 1/15/17
91	✓	Begin Spring Semester System Testing	100%	6 days	Mon 1/9/17	Sun 1/15/17
92	✓	**Week 2:**	**100%**	**6 days**	**Mon 1/16/17**	**Sun 1/22/17**
93	✓	Test Applications on Docker Environment	100%	6 days	Mon 1/16/17	Sun 1/22/17
94	✓	Discuss Expo Equipment Needs	100%	1 day	Mon 1/16/17	Mon 1/16/17
95	✓	Update Gantt Chart	100%	6 days	Mon 1/16/17	Sun 1/22/17
96	✓	Begin Revising Abstract	100%	6 days	Mon 1/16/17	Sun 1/22/17
97	✓	**Week 3:**	**100%**	**6 days**	**Mon 1/23/17**	**Sun 1/29/17**
98	✓	Work on Documentation from test results	100%	6 days	Mon 1/23/17	Sun 1/29/17
99	✓	Research Safe Security Steps for Containers	100%	6 days	Mon 1/23/17	Sun 1/29/17
100	✓	Plan Poster Layout	100%	1 day	Mon 1/16/17	Mon 1/16/17
101	✓	Test System Patching	100%	6 days	Mon 1/23/17	Sun 1/29/17
102	✓	**Week 4:**	**100%**	**6 days**	**Mon 1/30/17**	**Sun 2/5/17**
103	✓	Finalize Poster Ideas	100%	6 days	Mon 1/30/17	Sun 2/5/17
104	✓	Meet to discuss presentation	100%	6 days	Mon 1/30/17	Sun 2/5/17
105	✓	Start compiling Technical documents for expo	100%	6 days	Mon 1/30/17	Sun 2/5/17
106	✓	Finalize Abstract	100%	1 day	Sun 1/15/17	Sun 1/15/17
107	✓	Continue testing Systems	100%	6 days	Mon 1/30/17	Sun 2/5/17
108	✓	**Week 5:**	**100%**	**6 days**	**Mon 2/6/17**	**Sun 2/12/17**
109	✓	Create Portability Test	100%	6 days	Mon 2/6/17	Sat 2/11/17
110	✓	Work on Poster	100%	6 days	Mon 2/6/17	Sat 2/11/17
111	✓	Continue Testing Systems	100%	6 days	Mon 2/6/17	Sat 2/11/17
112	✓	**Week 6:**	**100%**	**6 days**	**Mon 2/13/17**	**Sun 2/19/17**
113	✓	Finalize Poster Ideas	100%	6 days	Mon 2/13/17	Sat 2/18/17
114	✓	Revise Report for Final Draft	100%	6 days	Mon 2/13/17	Sat 2/18/17
115	✓	Continue Testing	100%	6 days	Mon 2/13/17	Sun 2/19/17
116	✓	**Week 7:**	**100%**	**6 days**	**Mon 2/20/17**	**Sun 2/26/17**
117	✓	Continue Testing Systems	100%	6 days	Mon 2/20/17	Sun 2/26/17
118	✓	Meet to Discuss Presentation Deliverables	100%	1 day	Mon 2/20/17	Mon 2/20/17
119	✓	Implement Security Settings in System	100%	6 days	Mon 2/20/17	Sat 2/25/17
120	✓	**Week 8:**	**100%**	**6 days**	**Mon 2/27/17**	**Sun 3/5/17**
121	✓	Finalize Draft Expo Poster	100%	6 days	Mon 2/27/17	Sat 3/4/17
122	✓	Practice Presentation	100%	1 day	Mon 2/27/17	Mon 2/27/17
123	✓	Continue Revising Report	100%	6 days	Mon 2/27/17	Sat 3/4/17
124	✓	Finalize Documentation	100%	6 days	Mon 2/27/17	Sat 3/4/17
125	✓	Begin Preparing for Expo	100%	6 days	Mon 2/27/17	Sat 3/4/17
126	✓	**Week 9:**	**100%**	**6 days**	**Mon 3/6/17**	**Sun 3/12/17**
127	✓	Turn in Poster	100%	1 day	Mon 3/6/17	Mon 3/6/17
128	✓	Turn in Draft Final Report	100%	1 day	Mon 3/6/17	Mon 3/6/17
129	✓	Practice Presentation	100%	6 days	Mon 3/6/17	Sat 3/11/17
130	✓	Continue Testing Systems	100%	6 days	Mon 3/6/17	Sat 3/11/17





131	✓		**Week 10:**	**100%**	**6 days**	**Mon 3/13/17**	**Sun 3/19/17**
132	✓		Purchase Docker Services for Expo Demonstrations	100%	1 day	Mon 3/13/17	Mon 3/13/17
133	✓		Finalize Docker Nodes for Presentation	100%	6 days	Mon 3/13/17	Sat 3/18/17
134	✓		Setup Poster Demonstration	100%	6 days	Mon 3/13/17	Sat 3/18/17
135	✓		Meet to Practice Presentation	100%	1 day	Mon 3/13/17	Mon 3/13/17
136	✓		Continue testing Systems	100%	6 days	Mon 3/13/17	Sat 3/18/17
137	✓		**Week 11:**	**100%**	**6 days**	**Mon 3/20/17**	**Sun 3/26/17**
138	✓		Setup Nodes on Single Docker Account	100%	6 days	Mon 3/20/17	Sat 3/25/17
139	✓		Practice Presentation	100%	6 days	Mon 3/20/17	Sat 3/25/17
140	✓		Test AWS Node	100%	1 day	Mon 3/20/17	Mon 3/20/17
141	✓		Continue Testing Systems	100%	6 days	Mon 3/20/17	Sat 3/25/17
142	✓		**Week 12:**	**100%**	**6 days**	**Mon 3/27/17**	**Sun 4/2/17**
143	✓		Deliver Presentation	100%	1 day	Mon 3/27/17	Mon 3/27/17
144	✓		Create Screen Casts of System	100%	5 days	Mon 3/27/17	Fri 3/31/17
145	✓		Finalize System	100%	6 days	Mon 3/27/17	Sat 4/1/17
146	✓		**Week 13:**	**100%**	**6 days**	**Mon 4/3/17**	**Sun 4/9/17**
147	✓		Finalize Deliverables	100%	6 days	Mon 4/3/17	Sun 4/9/17
148	✓		**Week 14:**	**100%**	**6 days**	**Mon 4/10/17**	**Mon 4/17/17**
149	✓		Finalize Expo Deliverables	100%	6 days	Mon 4/10/17	Mon 4/17/17
150	✓		Present at Expo	100%	6 days	Mon 4/10/17	Mon 4/17/17
151	✓		**Week 15:**	**100%**	**1 day**	**Mon 4/24/17**	**Mon 4/24/17**
152	✓		Turn in Final Library Copy	100%	1 day	Mon 4/24/17	Mon 4/24/17

4.TECHNICAL ELEMENTS

4.1 Network

Docker will run in a Virtual Environment on a Linux Server. Docker Cloud links with the host and provides a Web Interface to manage the containers and services on the Linux Server. Docker Cloud will also provide easy deployment of Stacks to the different nodes in the Virtual Environment.

4.2 Sandbox

A Sandbox Environment will be utilized to host the Virtual Machines. The servers will be built in a Linux Server environment. Each server will be linked from the Sandbox to the Docker Cloud Web Interface for management. Because of Dockers capability of deploying multiple containers on a single host, the amount of Virtual Machines needed in the Sandbox environment will be limited to one or two Linux Servers.

4.3 Applications

Because of Docker's lightweight container virtualization technology, a hypervisor will not be required. Instead, the programs and applications will be executed in a container isolated from other processes. The applications used in the environment will be derived from a repository of Docker images, Docker Hub. In order to run the applications a Docker image will be used to create a container. A container will then contain all of the components needed to run applications in the environment.

4.3 Portability

Docker's portability will allow an administrator to deploy, replicate, move, and back up a workload more efficiently than using traditional virtual machines. Most applications rely on an operating system to function. Docker engine places applications in separate containers on an operating system creating independence. With Docker, a machines operating system can be upgraded and an application can be migrated between different operating systems without disrupting the functionality of the application.

4.4 Backup

Docker has the capability of creating backups without the need for any specialized software. A simple command can be used to launch a new container and mount the volume from the dbdata container. Docker will then mount a local directory /backup where it will back up the contents of the dbdata volume to a .TAR file within the /backup directory.

Figure 3 represents the Docker Container Infrastructure and **Figure 4** depicts Docker versus Traditional Virtualization Infrastructure. Docker Engine runs on a single Operating System. Separate containers run on the Docker Engine, and it is within these containers that applications are hosted.

Figure 3. Docker Container Infrastructure

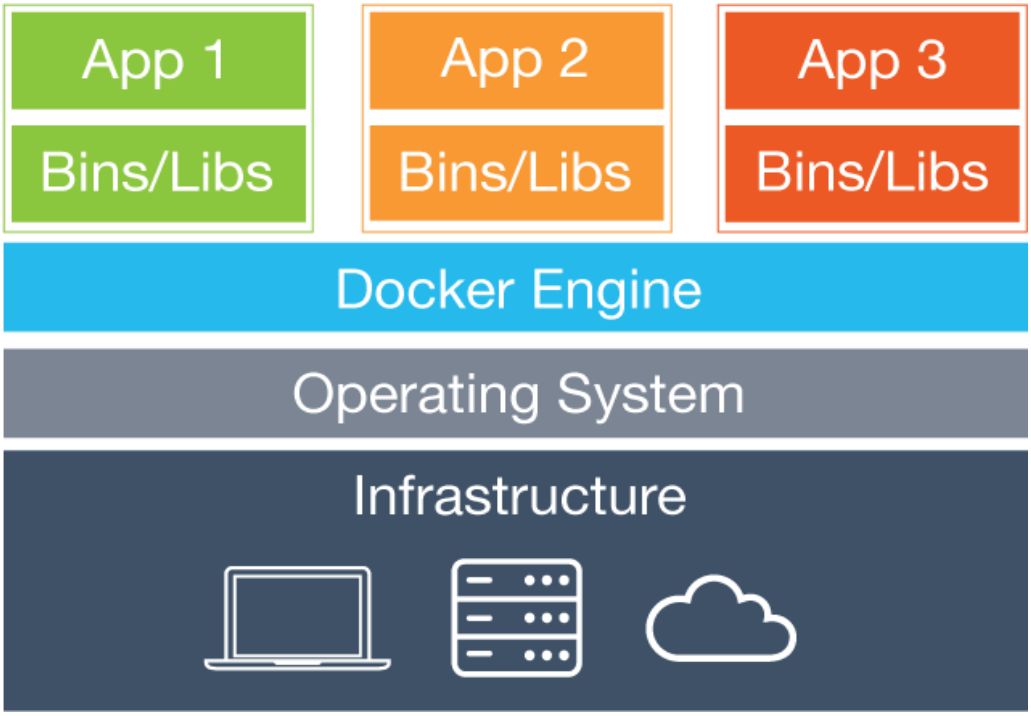
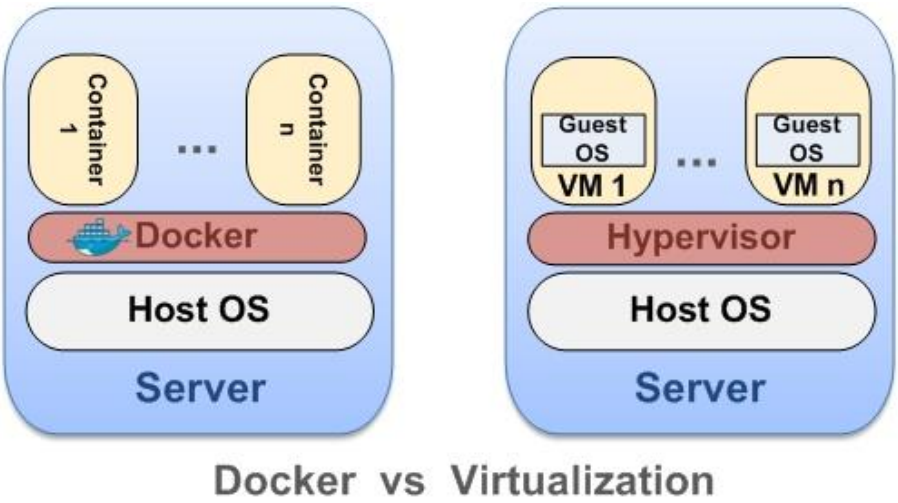


Figure 4. Docker vs Traditional Virtualization



5. WEB APPLICATION

5.1 Web Interface Design

Figure 5, 6, and 7 depict the user interface of the Docker Cloud Web Application.

Figure 5. Docker Cloud User Interface – Containers

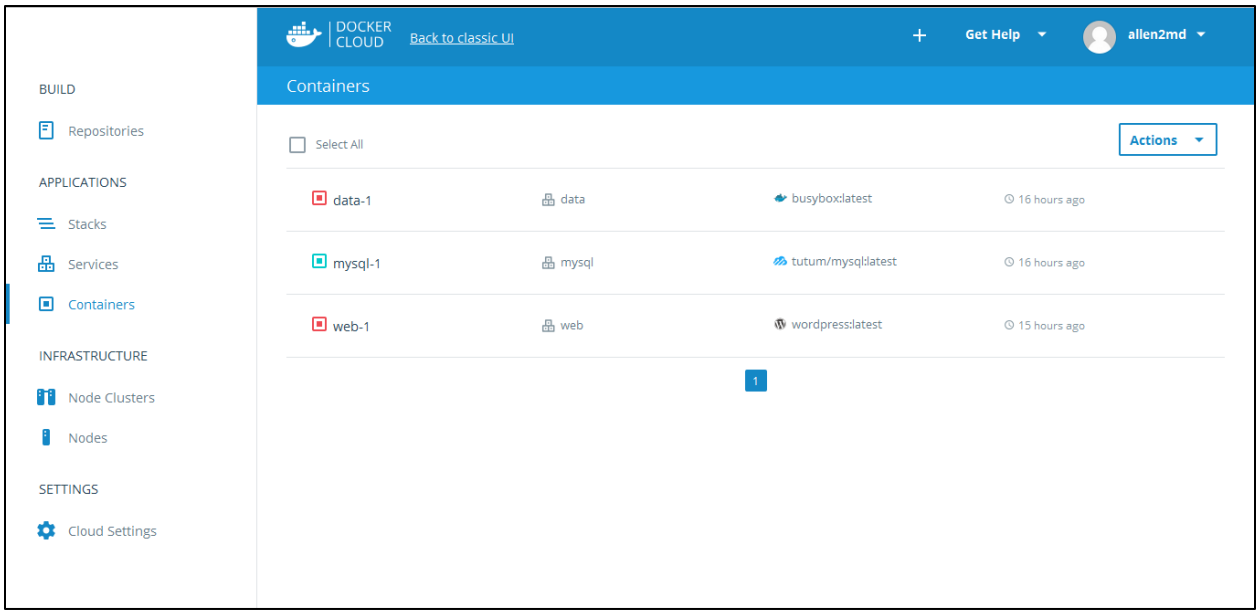


Figure 6. Docker Cloud User Interface - Nodes

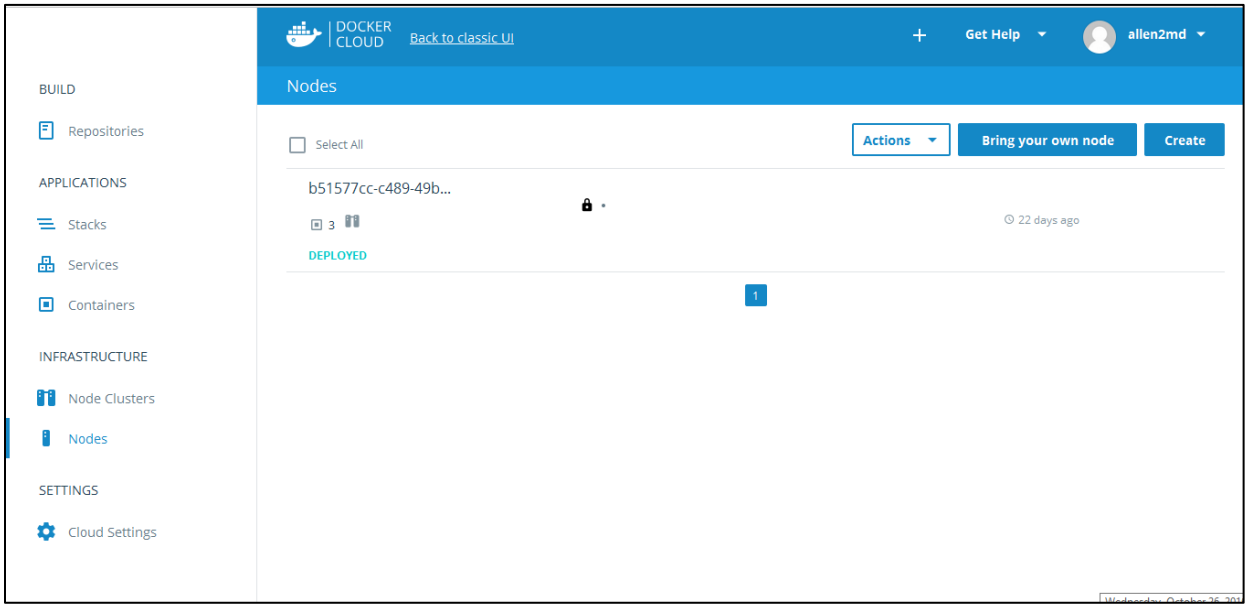
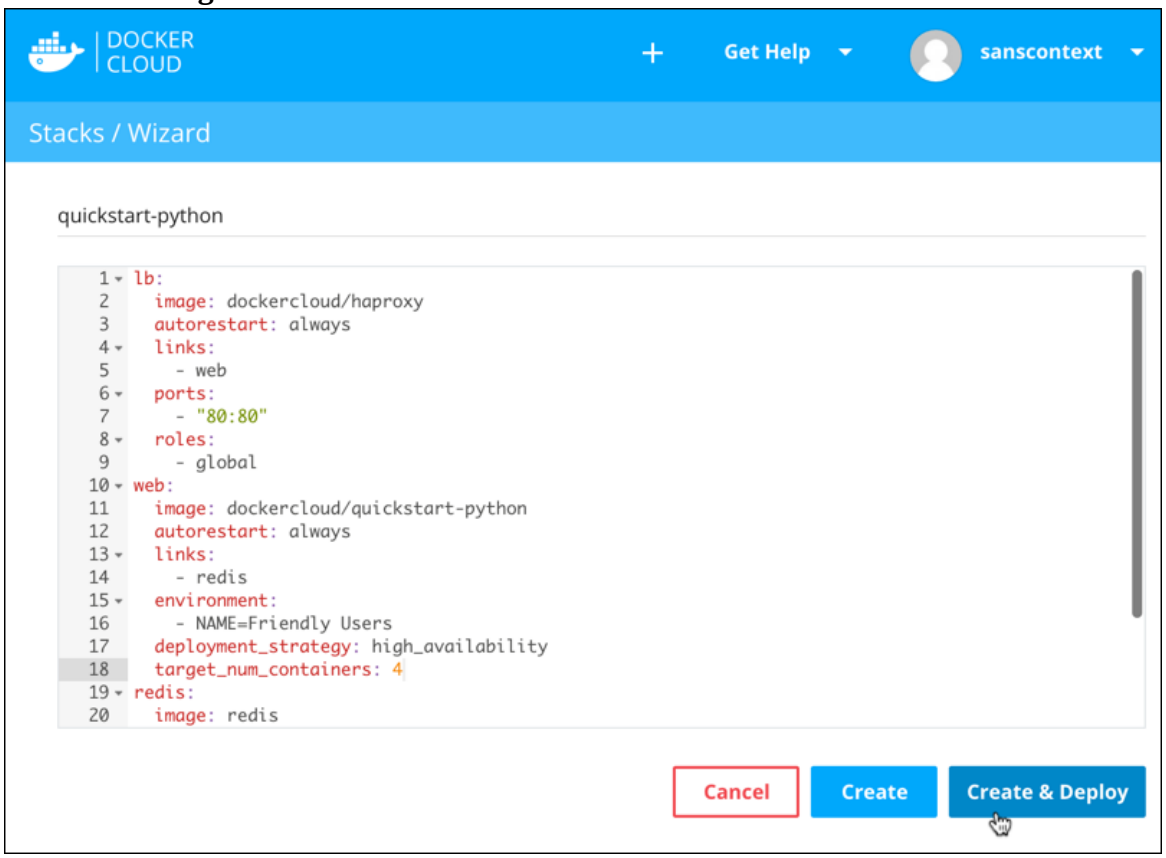


Figure 7. Docker Cloud User Interface – Stacks and Services



5.1.1 Docker Cloud

Docker Cloud is a hosted service that provides a Registry with build and testing facilities for Dockerized application images, tools that help manage the host infrastructure, and deployment features that help automate deploying images to an administrator's infrastructure. Docker Cloud can be accessed using a unique and free Docker ID. These credentials also provide access to the Docker Hub. Docker Hub is an online registry service that allows an administrator to publish Dockerized images on the internet publicly or privately. Docker Cloud allows an administrator to link their infrastructure or cloud service provider to the GUI in order to provision new nodes automatically, and deploy images directly from the Docker Cloud repositories on the hosts.

5.1.2 Nodes and Node Clusters

After a Docker Cloud account is linked with a host or multiple hosts, nodes can then be launched. Node clusters are a group of nodes of the same type from the same cloud provider, and they allow an administrator to scale the infrastructure by provisioning more nodes with ease.

5.1.3 Containers

A Docker Container takes a piece of software in a complete filesystem that contains everything needed to run including: code, runtime, system tools, system libraries, and anything else that can be installed on a server and wraps it into a segregated container. This guarantees that the software can run efficiently on any platform. Containers can be managed through the command line or on the Docker Cloud webpage.

5.1.4 Stacks and Services

Once an image is built, it can be used to produce containers that make up a service, or use Docker Cloud's stackfiles to combine it with other services and micro services to form a full application.

5.2 Security

Security is a common concern when utilizing Docker Containers. Containers can run the risk of privilege escalation. If an attacker can obtain root access within a containerized application, they could then potentially work their way to root access to the host system. A Denial of Service attack, where one container seizes control of all available system resources and stops the functionality of other containers, is another risk that containers face. Proactive steps can be taken

to ensure Docker containers always run as an ordinary user instead of root to prevent privileged attacks. The configuring of Docker control groups will allow an administrator to set limits on resources a container can use to avoid Denial of Service attacks. Last, avoiding images from untrusted repos, such as public repos, will help mitigate these risks.

5.3 Security Tools

Two security tools have been chosen to be implemented in the Docker environment to further protect the system from vulnerabilities. The tools chosen are called Docker Benchmark and OpenSCAP for Security Compliance. Docker Benchmark for Security is a script that scans the Docker system for several vulnerabilities that are commonly seen in container environments. Benchmark scans the system as a whole and runs a check against the CIS Docker 1.13 Benchmark guidelines and then returns a pass/fail report showing which areas need improvement. OpenSCAP provides resources and tools to assert compliance of the Docker containers. It does this by assessing and running containers and cold images as well as providing vulnerability and compliance audits. OopenSCAP returns an html report also displaying what areas need reconfigured to mitigate vulnerabilities.

5.4 Analyzing Containers

In order to analyze resource usage and performance characteristics of running containers an application called cAdvisor will be implemented in the environment. cAdvisor also known as Container Advisor will provide Systems Administrators a view of the resources being used by each container in the environment as well as the performance of the containers. Container Advisor collects information about the containers and exports the information in a user friendly format. Examples of the information it provide includes isolation parameters, historical resource usage, histograms and resource usage, and network statistics.

6.TEST PLAN

6.1 Overview

This section will provide an overview of the testing methodologies used in a Docker Container environment. This documentation will act as a guide for the following users:

- Systems Administrators
- Network Engineers
- Application Developers
- Managers

6.2 Scope

The scope of testing the functionality of a Docker Container environment will include general connectivity tests, testing local hosts with various commands, running targets inside of development containers, and how to build and test the documentation.

6.3 Objective

The objective of testing the Docker environment is to ensure that all aspects of the environment are functioning properly for System Administrators and Application Developers. This will ensure that the applications being hosted on the containers is working efficiently for the end users.

6.3.1 General Connectivity Test

This test is to ensure the Docker Environment is up and running and the administrator can connect to the appropriate Web interfaces as well as link nodes from the selected cloud provider.

- Log onto the Web interface

    1. In a Web Browser enter the following address to access the Docker Cloud Interface:

       https://cloud.docker.com

2. Enter the Administrator or Developer credentials depending on the current user
3. Click on the Nodes Tab to view connected nodes
4. Click on the desired node/s to view statistics of the node and ensure it is up and running and has desired disk space and memory as per the desired configuration

- Test Node Connectivity with Cloud Provider
  1. Access the Docker Cloud Interface with the following link: https://cloud.docker.com
  2. Enter the Administrator credentials for making changes to nodes
  3. Click on the Nodes Tab to View connected nodes
  4. Click the "Bring your own nodes" button
     a. Log into the selected cloud provider interface
     b. Configure a new image that will be connected to Docker Cloud
     c. Return to Docker Node page
  5. Run the given commands on the Node from the selected cloud provider in order to connect the node with the Web Interface

6.3.2 Testing the Local Host

This test would be used before submitting and pull requests with a code change. Running the entire Docker Engine test suite can take over half an hour. The commands to be used are as follows:

Table 4. Docker Test Commands

Target	What this target does
test	Run the unit, integration, and docker-py tests
test-unit	Run just the unit tests
test-integration-cli	Run the integration tests for the CLI
test-docker-py	Run the tests for the Docker API client

1. Open a terminal on the local host
2. Change to the root of your Docker repository ($ cd docker-fork)
3. Ensure you are in the development branch ($ git checkout dry-run-test)
4. Run the make test command ($ make test)
    a. This will create a temporary test container and will do the following:
        i. Creates a new binary
        ii. Cross-compiles all of the binaries for the various operating systems
        iii. Runs all the tests in the system
5. Estimated time for completion: 90 minutes

6.3.3 Running targets inside of development containers

When a developer is working inside of a development container they are using the hack/make.sh script in order to run tests. A single command line with multiple targets can be used to have a single target run all of the tests.

1. Open a terminal and change to the docker-fork root
2. Start a Docker development image
3. Run tests using the given hack/make.sh script
4. The tests run similarly to the local host tests, but can also run a subnet of these targets too.

6.3.4 Build and Test the Documentation

The Docker repository serves a central location for documentation source files and can be reached at the following location: https://hithub.com/docker/docker.github.io

- This documentation can be edited using extended Markdown language.
- The documentation can be edited in a plain text editor such as Notepad.
- It is important to always check documentation for grammar and spelling errors using an outside resource or the checker built into the text editor.
- When changing the source documentation it is important to also test the changes in the local Docker environment to ensure that the content changes are executed properly.

Building Documentation:

- To build the documentation a Docker container can be used without any additional set-up.
    - Jekyll is another resource that can be used resulting in a faster build, but it requires some additional installation prerequisites before it can be used
- Docker-compose will be used to build the docs locally on macOS, Windows, or Linux Operating Systems.
- The steps to use docker-compose are as follows:
    1. Enter the root of the repository
    2. Run the following command: $ docker-compose up
    3. In a web browser type: http://localhost:4000/ in order to view the documentation
    4. To stop the container, us CTRL+C
    5. To restart the container run: $ docker-compose start docs

7. CONCLUSION

7.1 Fall Semester 2016

In the 2015 Fall Semester, the team completed setting up a basic Docker Cloud environment. A Sandbox was used as a cloud provider to integrate a Linux Operating System with the Docker Cloud solution. The Linux Operating System acts as a host for the containerized applications that were deployed through Docker Cloud.

7.2 Spring Semester 2017

The Docker Cloud environment was completed, and the following elements were added to the system during the Spring Semester:

1. Setup and configured applications on hosts
2. Added a second Node to the environment
3. Fixed any “known” issues with system
4. Researched and implemented Security Tools
5. Created a demonstration featuring Portability and Security
6. Patched and configured further security requirements

APPENDIX A. ADDITIONAL INFORMATION

The team contacted the following professionals for clarification and additional insight.

1. Bo Vykhovanyuk, Instructor – IT Security
2. Mark Stockman, Instructor – IT Networking and Systems

APPENDIX B. REFERENCES

"Docker." *Docker*. N.p., n.d. Web. 26 Feb. 2017.

Sullivan, Dan. "How To Use Docker To Deploy Applications And Services - How to Use Docker to Deploy Applications and Services." *Tom's IT Pro*. N.p., 11 June 2015. Web. 26 Feb. 2017.

"Why did Docker Catch on Quickly and Why is it so Interesting?" *The New Stack*. N.p., 12 Oct. 2015. Web. 26 Feb. 2017.

Witbeck, Shane. "Making the Case for Docker." *Medium*. N.p., 12 Dec. 2014. Web. 26 Feb. 2017.

Christner, Brian. "How to Create a Docker Business Case." *BrianChristner.io*. BrianChristner.io, 01 Mar. 2016. Web. 26 Feb. 2017.

"Docker (software)." *Wikipedia*. Wikimedia Foundation, 18 Feb. 2017. Web. 26 Feb. 2017.