Microsoft Azure - Starter Kits for Partners

Hands on Lab

Application Servers Scenario

Open Source and Java

* (This scenario is under development – additional review to the references below and testing is recommended)





**MICROSOFT MAKES NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, AS TO THE INFORMATION IN THIS DOCUMENT.**

The information contained in this document represents the current view of Microsoft Corporation on the issues discussed as of the date of publication. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented after the date of publication.

Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

The descriptions of other companies’ products in this document, if any, are provided only as a convenience to you. Any such references should not be considered an endorsement or support by Microsoft. Microsoft cannot guarantee their accuracy, and the products may change over time. Also, the descriptions are intended as brief highlights to aid understanding, rather than as thorough coverage. For authoritative descriptions of these products, please consult their respective manufacturers.

© 2015 Microsoft Corporation. All rights reserved. Any use or distribution of these materials without express authorization of Microsoft Corp. is strictly prohibited.

Microsoft and Windows are either registered trademarks of Microsoft Corporation in the United States and/or other countries.

The names of actual companies and products mentioned herein may be the trademarks of their respective owners.

Contents

[Overview 5](#_Toc452017174)

[Scenario 1: Creating a LAMP Farm (Linux, Apache, MYSQL, PHP) – Without docker 5](#_Toc452017175)

[Installing on Ubuntu 5](#_Toc452017176)

[Installing on CentOS & Oracle Linux 6](#_Toc452017177)

[Installing on SUSE Linux Enterprise Server 6](#_Toc452017178)

[Scenario 2: Deploying Docker to Azure with VM Extension (Nginx) 7](#_Toc452017179)

[Scenario 3: Deploying Docker to Azure with Visual Studio and ARM (Nginx) 8](#_Toc452017180)

[Create an Azure Resource Group project 9](#_Toc452017181)

[Scenario 4: Java + Spring + Tomcat + Nginx Farm with Docker 10](#_Toc452017182)

[Rapid Environment Deployment and Repeatability 11](#_Toc452017183)

[Mongo DB and Docker 12](#_Toc452017184)

[Java Spring and Docker 14](#_Toc452017185)

[nginx and Docker 15](#_Toc452017186)

[Solution Architecture and Container Approach 16](#_Toc452017187)

[Docker Build Script 17](#_Toc452017188)

[Docker Run Script 17](#_Toc452017189)

[Docker and Linking 18](#_Toc452017190)

[Code Artifacts 20](#_Toc452017191)

[Additional Documentation 20](#_Toc452017192)

# Overview

Using Microsoft Azure, as your Infrastructure as a Service (IaaS) platform, will enable you to create and manage your infrastructure quickly, provisioning and accessing any host ubiquitously. Grow your business through the cloud-based infrastructure, reducing the costs of licensing, provisioning and backup.

**Estimated time** to complete this lab: **180 minutes**.

**Audience**: IT Pro, Architect, Application Owners and Developers

# Scenario 1: Creating a LAMP Farm (Linux, Apache, MYSQL, PHP) – Without Docker

A LAMP stack consists of the following different elements:

* **L**inux - Operating System
* **A**pache - Web Server
* **M**ySQL - Database Server
* **P**HP - Programming Language

## Installing on Ubuntu

You will need the following packages installed:

* apache2
* mysql-server
* php5
* php5-mysql

After running apt-get update to update the local list of packages, you can then install these packages with a single apt-get install command:

Copy to clipboardCopy

# sudo apt-get update

# sudo apt-get install apache2 mysql-server php5 php5-mysql

After running the above command you will be prompted to install these packages and a number of other dependencies. Press 'y' and then 'Enter' to continue, and follow any other prompts to set an administrative password for MySQL.

This will install the minimum required PHP extensions needed to use PHP with MySQL. Run the following command to see other PHP extensions that are available as packages:

Copy to clipboardCopy

# apt-cache search php5

## Installing on CentOS & Oracle Linux

You will need the following packages installed:

* httpd
* mysql
* mysql-server
* php
* php-mysql

You can install these packages with a single yum install command:

Copy to clipboardCopy

# sudo yum install httpd mysql mysql-server php php-mysql

After running the above command you will be prompted to install these packages and a number of other dependencies. Press 'y' and then 'Enter' to continue.

This will install the minimum required PHP extensions needed to use PHP with MySQL. Run the following command to see other PHP extensions that are available as packages:

Copy to clipboardCopy

# yum search php

## Installing on SUSE Linux Enterprise Server

You will need the following packages installed:

* apache2
* mysql
* apache2-mod\_php53
* php53-mysql

You can install these packages with a single zypper install command:

Copy to clipboardCopy

# sudo zypper install apache2 mysql apache2-mod\_php53 php53-mysql

After running the above command you will be prompted to install these packages and a number of other dependencies. Press 'y' and then 'Enter' to continue.

This will install the minimum required PHP extensions needed to use PHP with MySQL. Run the following command to see other PHP extensions that are available as packages:

Copy to clipboardCopy

# zypper search php

**Follow**:

<https://azure.microsoft.com/en-us/documentation/articles/virtual-machines-linux-install-lamp-stack/#setting-up>

# Scenario 2: Deploying Docker to Azure with VM Extension (Nginx)

Docker is a popular container management and imaging platform that allows you to quickly work with containers on Linux (and Windows as well). With Azure, you have the flexibility to deploy Docker in a few different manners depending on your needs:

* To quickly prototype an app, or if you already know and use Docker Machine, you can [use the Docker Machine Azure driver](https://azure.microsoft.com/en-us/documentation/articles/virtual-machines-linux-docker-machine/) to deploy Docker hosts within Azure.
* For a template based deployment, the Docker VM extension for Azure virtual machines can be used. This approach can integrate with Azure Resource Manager template deployments and includes all of the related benefits such as role base access, diagnostics, and post deployment configuration.
* You can also [deploy a full Docker Swarm cluster on Azure Container Services](https://azure.microsoft.com/en-us/documentation/articles/container-service-deployment/) for production-ready, scalable deployments that leverage the additional scheduling and management tools provided by Swarm.

This article focuses on using resource manager templates to deploy the Docker VM Extension in a custom, production-ready environment that you define.

**Follow**:

<https://azure.microsoft.com/en-us/documentation/articles/virtual-machines-linux-dockerextension/#deploy-a-template-with-the-docker-vm-extension>

# Scenario 3: Deploying Docker to Azure with Visual Studio and ARM (Nginx)

The Azure Resource Manager Tools in [Azure SDK 2.7 for .NET](https://azure.microsoft.com/en-us/downloads/) enables you to:

* Create an application using the Azure Gallery templates.
* Create and edit Azure Resource Manager deployment templates to provision resource including websites, database, virtual machines and more using the JSON Outline and Editor.
* Create and edit Azure Resource Manager deployment templates and parameter files to customize deployments for different environments, such as development, staging and production.
* Create resource groups and deploy templates into these to simplify the creation of resources.

The Azure Resource Manager enables you to create reusable deployment templates that declaratively describe the resources that make up your application such as an Azure Website and a SQL Azure database. This simplifies the process of creating complex environments for development, testing and production in a repeatable manner. It also provides a unified way to manage and monitor the resources that make up an application from the [Azure Preview Portal](https://portal.azure.com/).

You are able to create an application using the Azure Gallery Templates and define and manage your Azure resources using JSON templates. This makes it easier for you to quickly setup the environment you need to Dev/Test your application in Azure. The two key features are the Visual Studio integration with the Azure Gallery and the ability to create and edit Azure Resource Manager deployment templates.

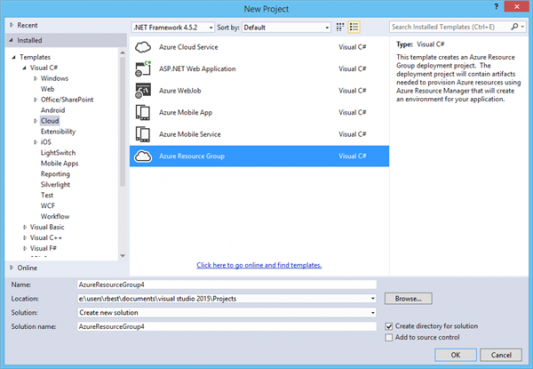
We will get started using this tooling by walking through a scenario. First, we will create a web site based on a Cloud Deployment project and we will look at what artifacts are added to your solution when you create your project. Then we are going to create and deploy the Azure resource group and resources we need for our application, which will include publishing of our application.

This tooling is available in the [Azure SDK 2.7 for .NET](https://azure.microsoft.com/en-us/downloads/) [download for [VS 2015](http://go.microsoft.com/fwlink/?linkid=518003&clcid=0x409) | [VS 2013](http://go.microsoft.com/fwlink/p/?linkid=323510&clcid=0x409)

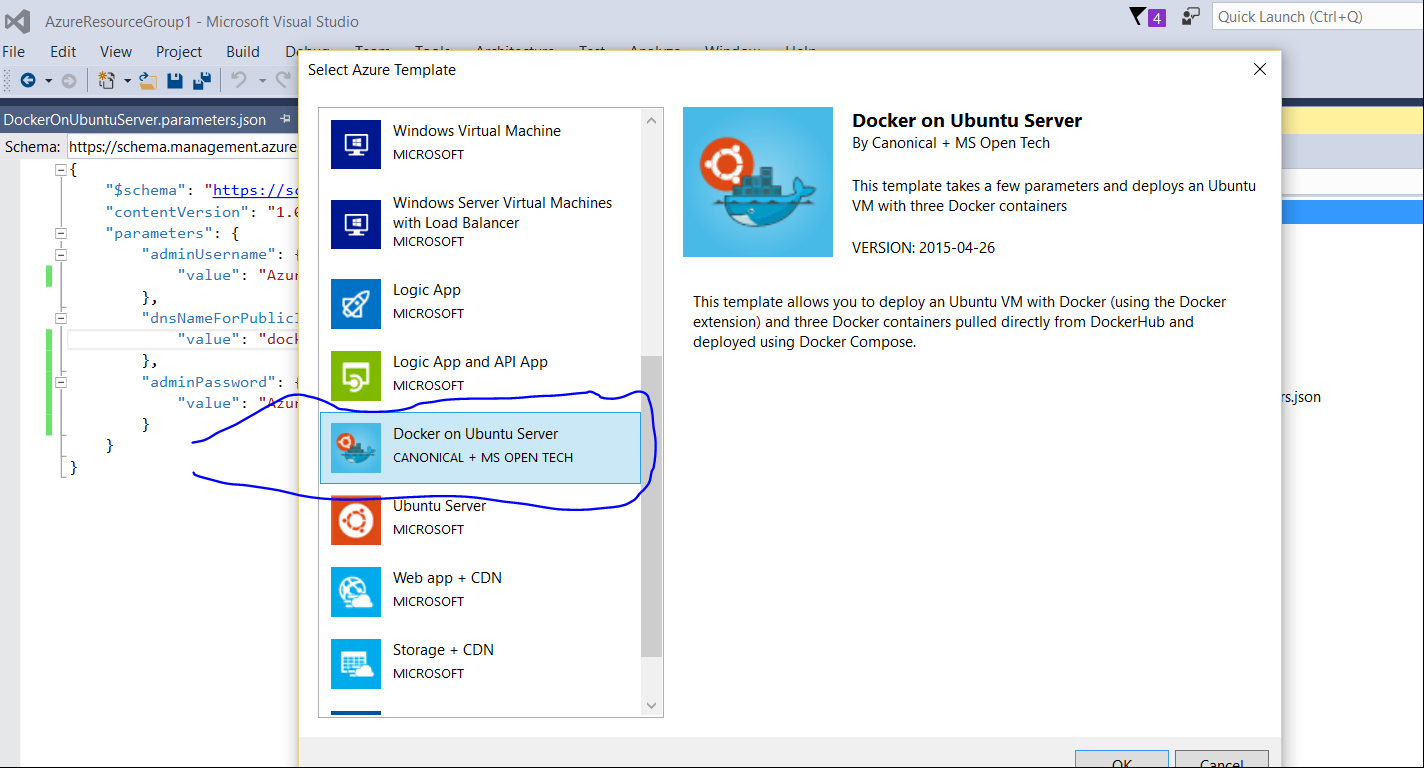
## Create an Azure Resource Group project

This walkthrough will cover how to deploy a website to a resource group. If you don’t have an ASP.NET 4.5.2 website already, create one now, by choose File -> Project and selecting Visual C#/Visual Basic -> Cloud -> ASP.NET Web Application.

With the Azure Resource Manager Tooling, we have made it possible to create Visual Studio applications using the Azure Gallery templates. As mentioned previously, the Azure SDK 2.7 must be installed to see the cloud deployment project templates. You can find these templates by selecting File->New Project. Select Templates -> Visual C#/Visual Basic -> Cloud-> “Azure Resource Group”.

[](http://acom.azurecomcdn.net/80C57D/blogmedia/blogmedia/2014/11/26/NewProjectDlg.png)

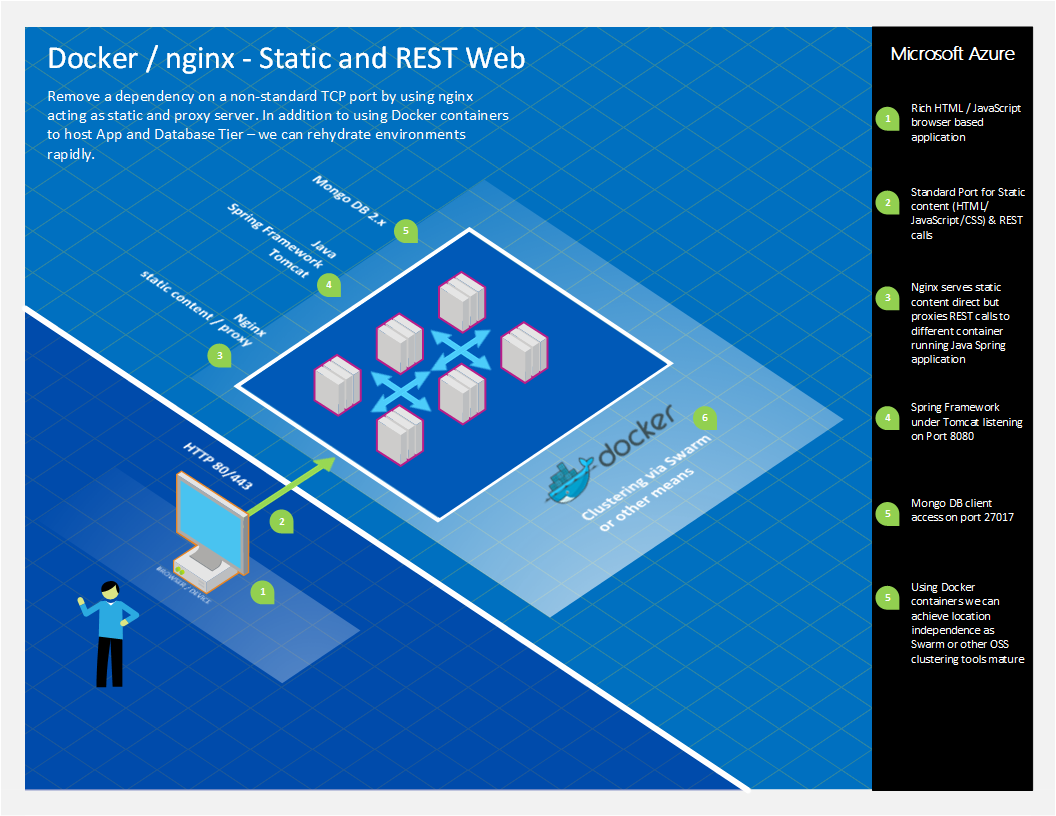
**Docker default Image with Linux + Nginx:**



Read more at: <https://azure.microsoft.com/en-us/blog/azure-resource-manager-2-5-for-visual-studio/>

# Scenario 4: Java + Spring + Tomcat + Nginx Farm with Docker

1. This scenario focuses on migrating existing multi-tiered applications to Azure, using Docker containers and nginx to expedite rehydration and reduce the overall surface area for open ports for the main application. After reviewing this case study you’ll have a basic understanding on a use case for nginx and dealing with a common scenario for on-premises applications and migrating them to the cloud – specifically, handling non-standard HTTP ports that are artifacts of developer choices and masked when on-premises.



### Rapid Environment Deployment and Repeatability

Docker is utilized as a deployment tool for defining and creating several Dockerfile files and Builds that are built and deployed rapidly – and run across 1 or more Docker Host containers as required.

Docker offers up many attributes that assist in the repeatability. While this case study did not cover clustering approaches (as there are several in Docker), Docker offers Swarm as their native (in beta) approach for Docker clustering.

This solution also did not publish to the Docker Hub – it took advantage of creating a build in a local Docker host container and running direct from there. While Docker Hub is an option, there are approaches to running your own Docker repository:

<http://azure.microsoft.com/blog/2014/11/11/deploying-your-own-private-docker-registry-on-azure/>

### Mongo DB and Docker

Docker offers an official Mongo DB image; the following solution is based upon that official image, and includes some modifications to insert seed data and check for ready-state.  This seed data is just for demonstration purposes and not intended for production. Note that the primary purpose of these Dockerfiles and the scripts was to rapidly establish a running Solution environment with all tiers working, so that developers and testers can validate a new build.

#### Dockerfile for Mongo

# Dockerizing MongoDB: Dockerfile for building MongoDB images  
 # Based on ubuntu:latest, installs MongoDB following the instructions from:  
 # http://docs.mongodb.org/manual/tutorial/install-mongodb-on-ubuntu/  
   
 FROM ubuntu:latest  
 MAINTAINER Shawn Cicoria shawn.cicoria@microsoft.com  
   
 ADD mongorun.sh .  
 # Installation:  
 # Import MongoDB public GPG key AND create a MongoDB list file  
 RUN apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv 7F0CEB10  
 RUN echo 'deb http://downloads-distro.mongodb.org/repo/ubuntu-upstart dist 10gen' | tee /etc/apt/sources.list.d/10gen.list  
   
 # Update apt-get sources AND install MongoDB  
 RUN apt-get update && apt-get install -y mongodb-org  
   
 # Create the MongoDB data directory  
 RUN mkdir -p -m 777 /data/db  
   
 RUN mkdir -p -m 777 /tmp  
   
 COPY mongodb.catalog.json /tmp/mongodb.catalog.json  
 COPY mongodb.dealers.json /tmp/mongodb.dealers.json  
 COPY mongodb.quotes.json /tmp/mongodb.quotes.json  
 COPY mongorun.sh /tmp/mongorun.sh  
   
 WORKDIR /tmp  
   
 RUN chmod +x ./mongorun.sh  
   
 # Expose port #27017 from the container to the host  
 EXPOSE 27017  
   
 VOLUME /data/db  
   
 ENTRYPOINT ["./mongorun.sh"]

#### Mongo Startup Script

The script utilized as the entry point in the Dockerfile initializes a log file, and then starts MongoDB on a background process.  It then waits for MongoDB to be ready by reading the log file for the message **“wating for connections on port”**.  This represents one way to check for readiness.  In the OrderService approach that follows in the next section, you will see another way.

#!/bin/bash  
   
 # Initialize a mongo data folder and logfile  
 sudo rm -r /data/db 1>/dev/null 2>/dev/null  
 mkdir -p -m 777 /data/db   
 touch /data/db/mongodb.log  
 echo step 1  
 # Start mongodb with logging  
 # --logpath Without this mongod will output all log information to the standard output.  
 # --logappend Ensure mongod appends new entries to the end of the logfile. We create it first so that the below tail always finds something  
 /usr/bin/mongod --smallfiles --quiet --logpath /data/db/mongodb.log --logappend &  
 MONGO\_PID=$!  
 echo step 2  
 # Wait until mongo logs that it's ready (or timeout after 60s)  
 COUNTER=0  
 grep -q 'waiting for connections on port' /data/db/mongodb.log  
 while [[ $? -ne 0 && $COUNTER -lt 90 ]] ; do  
 sleep 2  
 let COUNTER+=2  
 echo "Waiting for mongo to initialize... ($COUNTER seconds so far)"  
 grep -q 'waiting for connections on port' /data/db/mongodb.log  
 done  
   
 # Now we know mongo is ready and can continue with other commands  
 echo now populate  
 #some point do something to chedk if already run; but for this demo just do it.  
 /usr/bin/mongoimport -d ordering -c catalog < /tmp/mongodb.catalog.json  
 /usr/bin/mongoimport -d ordering -c dealers < /tmp/mongodb.dealers.json  
 /usr/bin/mongoimport -d ordering -c quotes < /tmp/mongodb.quotes.json  
   
 wait $MONGO\_PID

### Java Spring and Docker

For the Java Spring framework based application tier, which provides various REST services that the HTML front-end calls directly, the following Dockerfile was used

#### OrderService Dockerfile

FROM java:8-jre  
 MAINTAINER Shawn Cicoria shawn.cicoria@microsoft.com  
   
 ENV APP\_HOME /usr/local/app  
 ENV PATH $APP\_HOME:$PATH  
 RUN mkdir -p "$APP\_HOME"  
   
 WORKDIR $APP\_HOME  
   
 ADD ordering-service-0.1.0.jar $APP\_HOME/  
 ADD startService.sh $APP\_HOME/  
   
 RUN chmod +x startService.sh   
   
 EXPOSE 8080  
 #CMD ["java", "-jar", "ordering-service-0.1.0.jar"]  
 CMD ["./startService.sh"]

#### StartService Script

The OrderService Dockerfile references the following script, which polls MongoDB using a hostname of ‘mongodb’ (you’ll see that alias used in the ‘docker run –link’ commands later) to check for ready state.

#!/bin/bash  
   
 while ! curl http://mongodb:27017/  
 do  
 echo "$(date) - still trying"  
 sleep 1  
 done  
 echo "$(date) - connected successfully"  
   
 java -jar ordering-service-0.1.0.jar

### nginx and Docker

nginx [engine x] is an HTTP and reverse proxy server that is popular and used in this solution to provide the static file content needs along with providing a proxy to the OrderService, using simple rules. The primary benefit is squashing down to a single IP port for all traffic – however, nginx offers up other potential for distributed and rule based proxy routing and an abstraction of the OrderService endpoints – thus location transparency.

#### Nginx Dockerfile

Note that the Dockerfile uses the base image from Docker for nginx and extends that for custom configuration and content. There is a parameter of “daemon off” to nginx as without it the container would just exit as Docker requires the process to continually run in order for it to assume it’s running and manageable.

FROM nginx:1.7.10  
 MAINTAINER Shawn Cicoria shawn.cicoria@microsoft.com  
   
 ENV WEB\_HOME /usr/local/web  
   
 ADD Web.tar $WEB\_HOME/  
   
 COPY nginx.conf /etc/nginx/nginx.conf  
   
 EXPOSE 8000  
 CMD ["nginx", "-g", "daemon off;"]

#### Nginx Configuration

worker\_processes 1;  
   
 events {  
 worker\_connections 1024;  
 }  
   
 http {  
 include mime.types;  
 default\_type application/octet-stream;  
   
 sendfile on;  
 keepalive\_timeout 65;  
   
 gzip on;  
   
 server {  
 listen 8000;  
 server\_name localhost;  
   
 location / {  
 root /usr/local/web/Web;  
 index index.html index.htm;  
 }  
   
 error\_page 500 502 503 504 /50x.html;  
   
 location = /50x.html {  
 root html;  
 }  
   
 location /catalog {  
 proxy\_pass http://orderservice:8080;  
 }

location /quotes {  
 proxy\_pass http://orderservice:8080;  
 }  
 location /shipments {  
 proxy\_pass http://orderservice:8080;  
 }  
 }  
 }

## Solution Architecture and Container Approach

In the diagram at the end of this section, the solution takes advantage of Docker and can scale up to numerous Docker hosts (containers). Note that we’ve also achieved use of a single IP port along with location transparency to the running containers and the calling clients.

In addition, for full clustering support you must ensure that any calls made between each tier leverages Docker’s linking of containers as Docker provides dynamic IP addresses and container names for each running container.

### Docker Build Script

The Docker build script is below and is fairly standard.

#!/bin/sh  
 cd mongoseed  
 docker build -t scicoria/mongoseed:0.1 .  
 cd ../orderService  
 docker build -t scicoria/orderservice:0.1 .  
 cd ../staticsite  
 docker build -t scicoria/staticsite:0.1 .  
 cd ..

### Docker Run Script

For running the containers, it’s important that the “client” is aware or “linked” to containers is needs to initiate requests to.  In the following script the following links are in place:

**1.       StaticWeb -> OrderService (DNS name OrderService) on Port 8000**

**2.       OrderService -> MongoDB (DNS name mongodb) on Port 27017**

#!/bin/sh  
 docker run -d -p 27017:27017 --name mongodb -v /data/db:/data/db scicoria/mongoseed:0.1  
 docker run -d -p 8080:8080 --name orderservice --link mongodb:mongodb scicoria/orderservice:0.1  
 docker run -d -p 8000:8000 --link orderservice:orderservice scicoria/staticsite:0.1

### Docker and Linking

Docker provides virtual networking among containers and any communication among containers must be explicitly declared when running a container. While the ports that are identified on the command line with the **‘-p’**parameters are for external communication into the container, Docker prevents that communication – unless it is explicitly linked.  In addition you should be aware of the hostnames and dynamic IP addressing and Ports that are also used, as Docker effectively provides NAT (network address translation) services among containers.

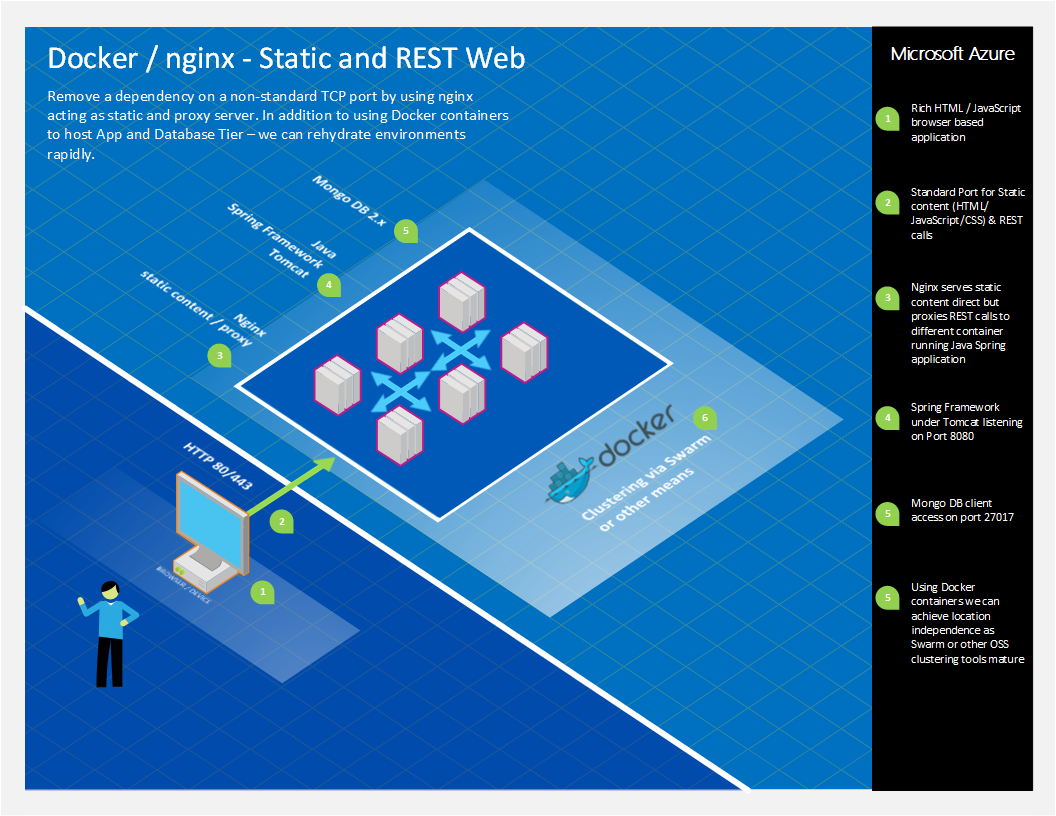
#### Eliminate use of Non-Standard Ports

Finally, with this solution we can eliminate the use of non-standard ports and reduce configuration, support, and troubleshooting issues with regards to corporate firewalls, proxy servers, etc. that may prevent non-standard port traffic from traversing the network.

The final code used in the static web site for the JavaScript calls simply builds the endpoint name using the browser’s location provider.  Since all traffic is now being routed through the same nginx front-ends, and nginx is determining which traffic to proxy and send to **OrderService**, we’ve reduced that complexity by a significant amount.

So, in updating the static web sites “serverConfig.js” – which executes in the Browser to the following, we have a minimal impact on the static web site and it’s supporting JavaScript:

var baseAddress = window.location.protocol + '//' + window.location.hostname;

[](https://msdnshared.blob.core.windows.net/media/MSDNBlogsFS/prod.evol.blogs.msdn.com/CommunityServer.Blogs.Components.WeblogFiles/00/00/01/71/24/0624.ng2.png)

## Code Artifacts

The source files are published to GitHub here – with the [**nginix**] branch represented here:

<https://github.com/cicorias/IgniteARMDocker/tree/nginix> [**nginx**]   (note the different branch from ‘master’).

This solution and source is part of a Pre-MS Ignite Session on DevOps – with additional walk-through items here: <https://github.com/Microsoft/PartsUnlimitedMRP>

## Additional Documentation

**Using the Docker VM Extension to deploy your environment**

From <[*https://azure.microsoft.com/en-us/documentation/articles/virtual-machines-linux-dockerextension/*](https://azure.microsoft.com/en-us/documentation/articles/virtual-machines-linux-dockerextension/)>

**Deploy an Azure Container Service cluster**

From <[*https://azure.microsoft.com/en-us/documentation/articles/container-service-deployment/*](https://azure.microsoft.com/en-us/documentation/articles/container-service-deployment/)>

**Container management with Docker Swarm**

From <[*https://azure.microsoft.com/en-in/documentation/articles/container-service-docker-swarm/*](https://azure.microsoft.com/en-in/documentation/articles/container-service-docker-swarm/)>

**Container management through the web UI**

From <[*https://azure.microsoft.com/en-in/documentation/articles/container-service-mesos-marathon-ui/*](https://azure.microsoft.com/en-in/documentation/articles/container-service-mesos-marathon-ui/)>