Preity Gupta

containeriZe C, C++ and VC++ Applications to Azure

Contents

[Solution Description 2](#_Toc505870974)

[Solution Approach 2](#_Toc505870975)

[Value Added 4](#_Toc505870976)

[Accelerators used and its purpose 4](#_Toc505870977)

[Activity breakup and efforts required 4](#_Toc505870978)

[Limitation 5](#_Toc505870979)

[Demo link 5](#_Toc505870980)

[Summary 5](#_Toc505870981)

[Appendixes 6](#_Toc505870982)

# Solution Description

Containerization of C, C++ and VC++ applications on Docker VM on Azure can be done using Docker for Linux software. Here on Windows based Azure VM, install Docker for Windows software from <https://docs.docker.com/docker-for-windows/> and switch it to Docker for Linux.

# Solution Approach

Methods of Containerizing existing C, C++ and VC++ applications with Linux Containers on Windows VM or on Linux VM.

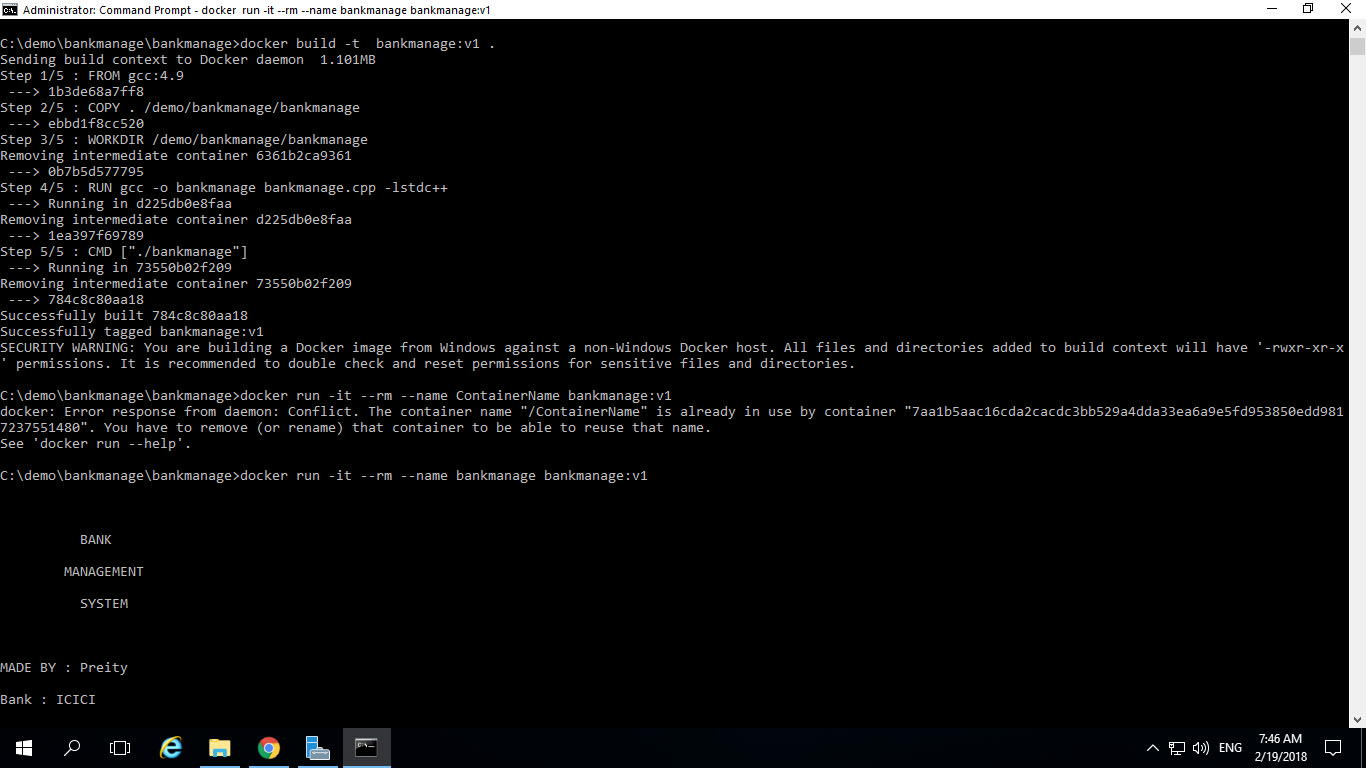
|  |  |  |  |
| --- | --- | --- | --- |
|  | C | C++ | VC++ |
| Compiler | Using gcc complier that compiles \*.c files of C applications | Using g++ compiler and compiles \*.cpp files of C++ applications | Using g++ compiler and compiles \*.cpp files of VC++ applications |
| Prequisites | * \*.c file * Dockerfile | * \*.cpp file * Dockerfile | * \*.cpp file * Dockerfile |
| Dockerfile | FROM gcc:4.9  COPY . /AppsName  WORKDIR /AppsName  RUN gcc --o AppsName AppsName.c  CMD ["./AppsName"] | FROM gcc:4.9  COPY . /AppsName  WORKDIR /AppsName  RUN g++ --o AppsName AppsName.cpp --lstdc++  CMD ["./AppsName"] | FROM gcc:4.9  COPY . /AppsName  WORKDIR /AppsName  RUN g++ --o AppsName AppsName.cpp --lstdc++  CMD ["./AppsName"] |

Create a Dockerfile (in a vi editor or in Notepad and save it with no extension while selecting file type as “All files”) in the same directory as the AppsName.cpp or AppsName.c file. A Dockerfile contains instructions to build a Docker image that could be used to run a Docker container. Meaning of instructions of Dockerfile is:

|  |  |
| --- | --- |
| **Dockerfile Instruction** | **Description** |
| FROM gcc:4.9 | The Docker image to use as the base image is gcc with tag 4.9. |
| COPY . /AppsName | Copy the files in the current directory to the / AppsName directory. |
| WORKDIR /AppsName | Set the /AppsName directory as the working directory. |
| RUN g++ --o outputname AppName.cpp | Run the g++ command g++ with output as " outputname" and input as "AppName.cpp". The command generates a runnable application called " AppName " |
| CMD ["./AppsName "] | Run the compiled, runnable application ./ AppsName |

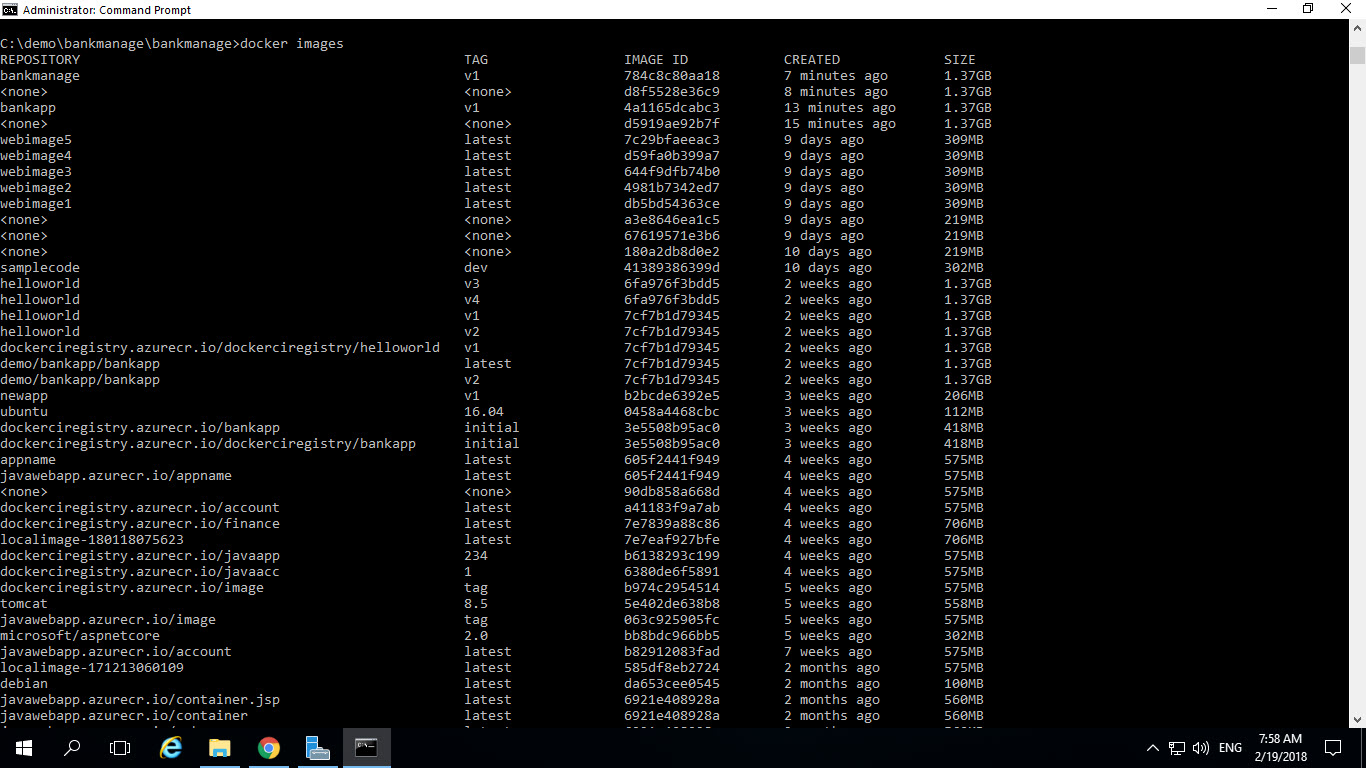
Within directory named AppsName, place both \*.cpp or \*.c file and Dockerfile file for respective application and sets its permission to global for active user. Right click the directory, add user and give user admin rights. Set the environment variable for docker.exe so that docker commands can be ran from anywhere in system from command prompt. In Command prompt, run cd command to move to the location of directory where \*cpp or \*.c file and Dockerfile is kept. Then run the docker build command to create a Docker image called AppsName:v1 from the Dockerfile.

docker build -t AppsName:v1 .



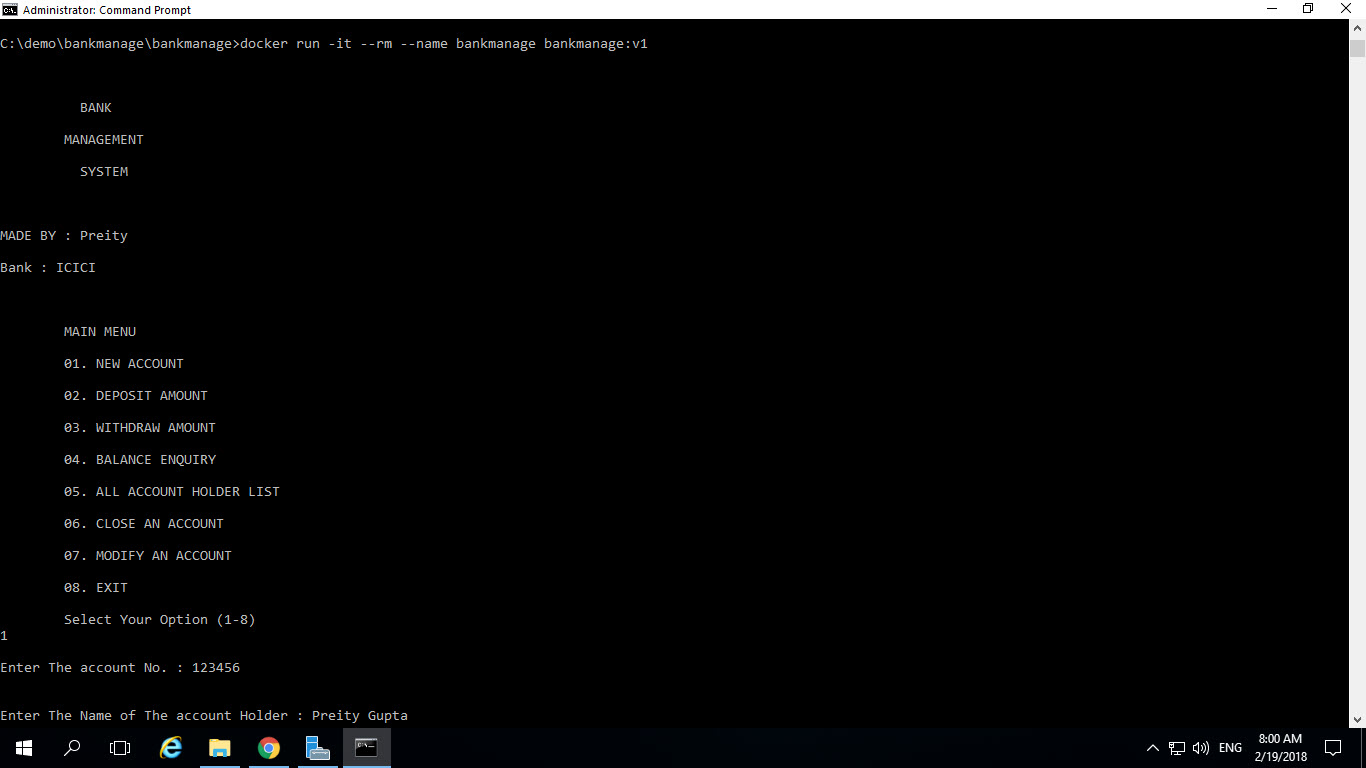
Subsequently, list the Docker images.

docker images



Having created the Docker image, run a Docker container with the docker run command. The Docker container may optionally be named, "ContainerName" for example, with the --name option. If the --name option is not used, a random name is used for the Docker container. The --rm option is called the "Clean up" option and removes the Docker container and the filesystem & volumes associated with the container after it has run. Run the following docker run command for the Docker image AppsName:v1.

docker run -it --rm --name ContainerName AppsName:v1



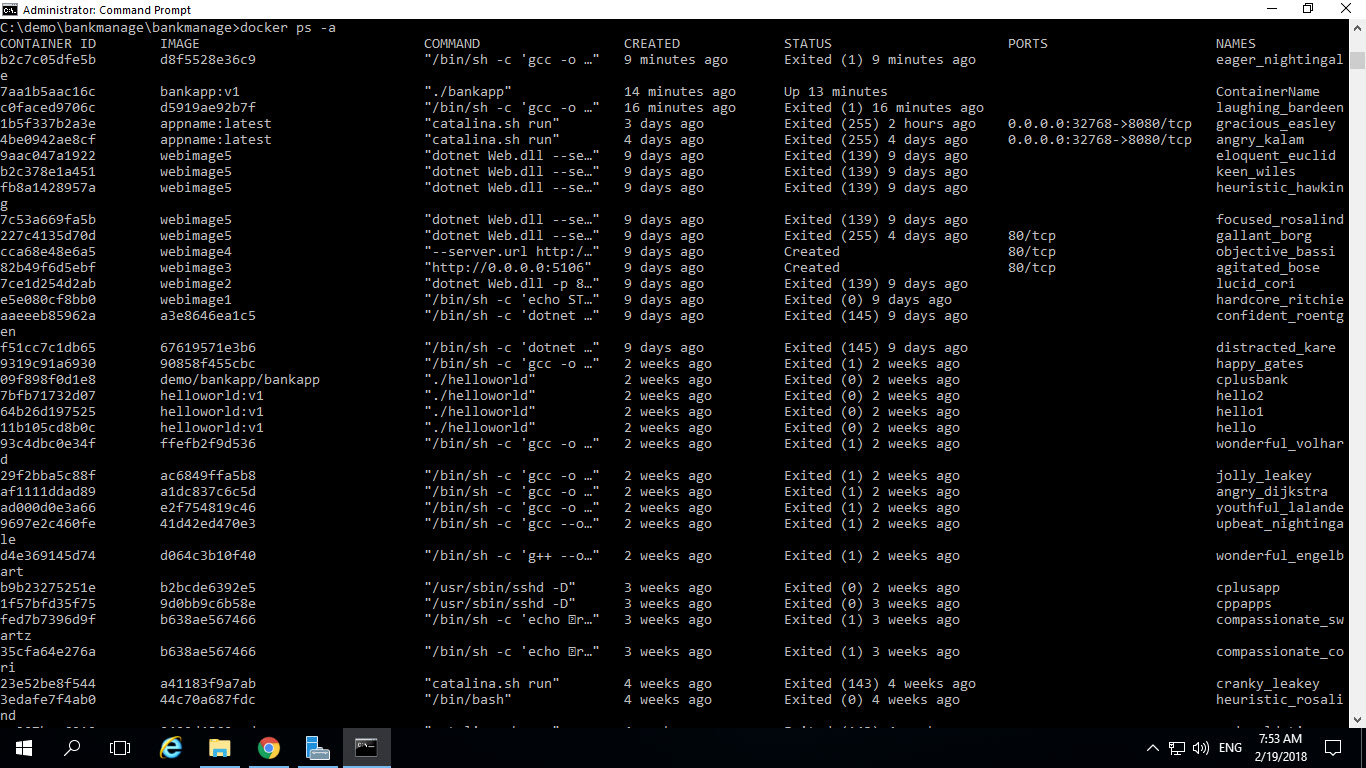
The Docker containers get removed when using the --rm option. To List the running containers:

docker ps



List of exited Docker containers:

docker ps -a



Docker applications are compiled into a Docker image with docker build and docker run commands. C, C++ and VC++ applications are run each time therefore to avoid over load of containers in Docker, container is moved to exited state after running an application.

# Value Added

* Docker Engine makes better use of the operating system kernel in comparison to a virtualization platform such as Virtual Box or VMWare. A single Docker container does not make use of a whole OS kernel, whereas a single virtual machine does.
* Each Docker container includes its own filesystem and networking, which makes it an isolated process on the Docker Engine. A single Docker Engine with multiple Docker containers running in isolation makes it feasible to run different applications. Even some containers make use of other containers.
* One of the main benefits of Docker Engine is the ease of installation and configuration for software.

# Accelerators used and its purpose

|  |  |
| --- | --- |
| Tools Used | Purpose |
| Docker for Linux | It enables Docker support and enables debugging and running of application on Docker |
| Command Prompt | Enables us to execute docker build and docker run commands |

# Activity breakup and efforts required

|  |  |
| --- | --- |
| Activities | Efforts |
| Installation of Docker of Windows Software | Go to <https://docs.docker.com/docker-for-windows/> URL and download stable version and install in client system. Within 5 minutes’ system is compliant for Docker support. |
| Creation of VM in Azure | Go to Azure Portal and create windows VM. It takes approx. 10 minutes |
| Build Image | Passing Docker Build commands creates image within 2 minutes |
| Run Image | Passing Docker Run commands run application on docker container with in 1 minute |

# Limitation

C, C++ and VC++ applications can be deployed to Linux VM or Windows VM with Docker for Linux support but cannot be deployed to Azure Container Services due to UI interface incompatibility.

# Demo link

<https://filer.hclets.com/index.php/apps/files/?dir=%2Fexchange%2FApp%20Modernization%2FAzure-Demos%2FPreity>

Filename: Containerize C, C++, VC++ application to Azure.zip

# Summary

Container-based deployments have quickly become the preferred approach for managing the build and release of complex applications. Container-based development is both productive and compelling, and it reduces the number of moving parts, which historically was the cause of many mistakes and system challenges. Containers delivers highly reliable and secure systems.

# Appendixes

<https://visualgdb.com/>

<https://blogs.msdn.microsoft.com/vcblog/2016/11/08/developing-linux-c-applications-with-azure-docker-containers/>

<https://blog.jessfraz.com/post/docker-containers-on-the-desktop/>