DevOps Tools

Day - 16

Prakash Ramamurthy
prakash.ramamurthy@wipro.com

Agenda

- Containerization and Docker
- Docker CLI
- Hands-On Demonstration

Containerization and Docker

Docker

Overview

- Docker provide container platform for every application across hybrid cloud
- Rapid digitization of business is forcing legacy applications to be hosted in diverse cloud platforms, datacentres as well with different application architecture.
- Containerization provide perfect isolation between applications and infrastructure.
- This would allow developers and operations teams to use their full potential yet with good collaboration and innovation.

Docker

Container Platform

- Complete solution to solve multiple problems across diverse set of applications having different requirements
- Provide a sustainable way of maintaining the applications isolated from each other
- Allow security, governance, automation, support and certification required for each application to be handled separately without affecting other applications
- Enterprise Edition (EE) of docker would provide Enterprise grade support and features to containerize any enterprise application without the fear being locked into an architecture or infrastructure

Containers

Container Image

- Container image has required executable package of application along with all required dependencies
- Dependencies include system tools, system libraries and everything required for run time environment
- Container isolate the application as well its run time environment from other content on the same machine
- As both application and run time environment are packed into one and same is used for development, test and production environment, it solves the dependency issues

Containers

Lightweight

- All containers running on same machine share the operating system kernel
- Kernel is not part of container image, which makes it lightweight
- Containers start the application quickly compared to starting a VM
- Containers also use less compute and RAM space
- Images are constructed in layered form allowing common files to be shared between images
- This would further minimizes the disk space
- Image downloads also will be faster when images share lower layers.

Containers

Standard

- Docker containers and images are constructed based on open standards
- Docker containers normally run on all major Linux flavours
- Docker containers with Microsoft applications can also run on Windows
- Docker containers can run across different infrastructure like physical servers, Virtual Machines as well on cloud servers

Containers

Secure

- Applications as well the run time environment inside container is isolated from its surroundings as well from other containers
- This isolation would limit app issues to that specific container
- Issues in one app will not be affecting other apps on the same machine

Docker

Freedom of Choice

- Containerization provide flexibility to evolve your application by isolating each application
- This would allow applications to evolve as required starting from the legacy
- Irrespective of size of organization, containerization would allow applications be divided into micro services and linking them together into bigger services
- Such dividing of application would allow easy manageability

Docker

Agile Operations

- Docker eliminate the major bug that stall the productivity – “Its working on my machine”
- Docker provide same environment packed into container to developers and IT operations
- This would eliminate the dependency issues across machines increasing productivity
- Teams working across application life cycle can work smarter and faster together

Docker

Integrated Security

- Docker packs the application and the environment to run the application in a container separating the same from surrounding environment
- This would allow an integrated security framework exclusively for specific application
- This would also improve policy automation yet maintaining the performance of application
- At the same time the performance of other applications too will not be affected.

Docker

Modernize Traditional Applications

- Legacy applications can be placed into containers by creating Docker image of application
- Once the traditional application is containerized security, governance, certification can be applied exclusively to the containerized application.
- Containers can be placed on cloud and can be moved easily between cloud servers reducing the cost.
- This would modernize legacy applications without changing any code.

Docker

Hybrid Cloud

- Application migration across cloud, or if an application is used across multiple cloud environment or a hybrid cloud environment is always challenging
- Once the application is containerized, as both application and their dependencies are packed into a container, it is easier to port them across multiple platforms.
- This eliminates the bug that “it worked there but not here” problem
- Containerized applications automatically work consistently across different infrastructure.

Docker

Docker and DevOps

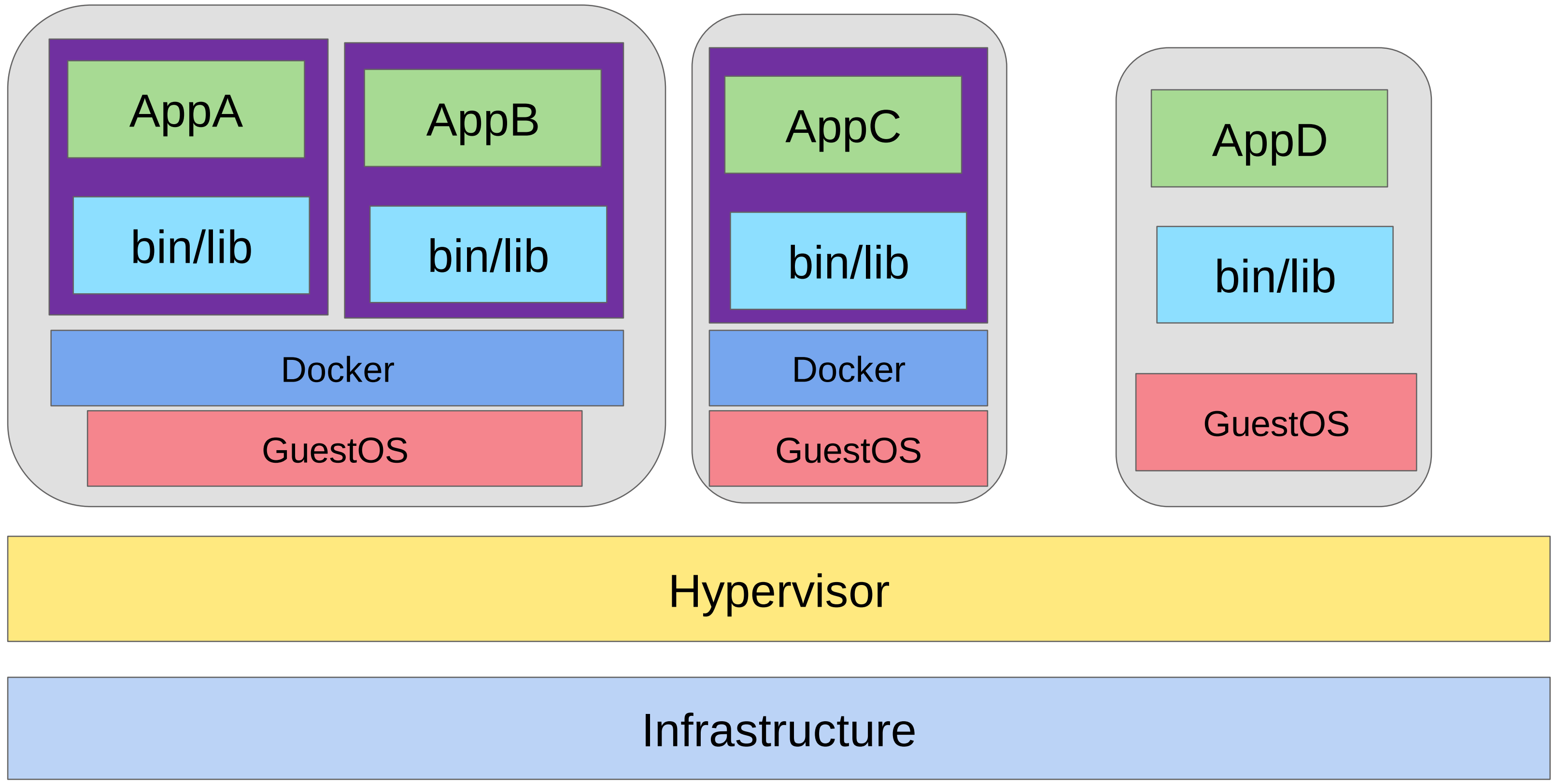
- Containerizing the applications would make DevOps pipeline smoother
- Since the containers include operating environment, the dependency issues never exists
- At different levels of application life cycle in DevOps pipeline, application always work in the same container environment
- This would eliminate application conflicts and increase productivity of team members
- Docker enables true separation across applications which accelerate adoption of DevOps processes.

Docker

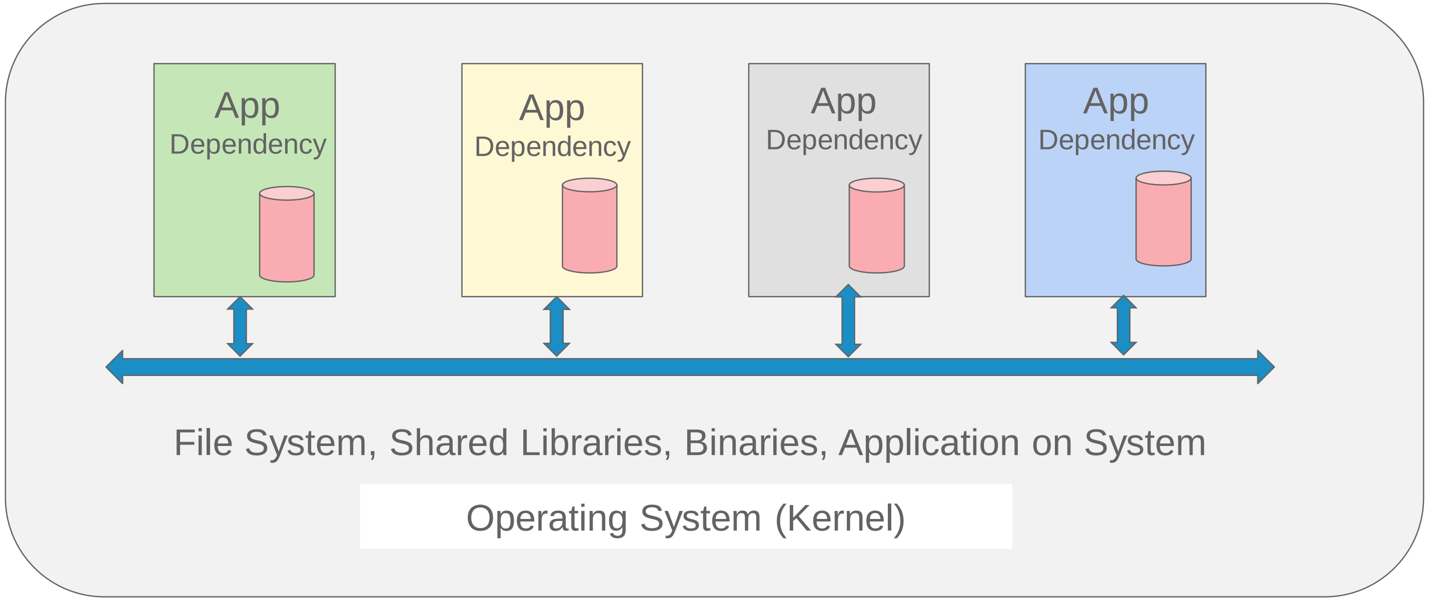
Microservices

- Docker enable Microservices based application development
- As the applications or services are divided into smaller microservices packed into containers, it will be faster to design, develop, deploy and release
- As microservices being smaller are managed by small well coordinated team, manageability will be easier
- With each microservice being containerized the dependencies are automatically taken care of.
- With containerization each microservice would be perfectly isolated interacting only through exposed APIs.

Containers and Virtual Machines



Containers on a Server


App
Dependency
App
Dependency
App
Dependency
App
Dependency
File System, Shared Libraries, Binaries, Application on System
Operating System (Kernel)

Hands-On Demonstration

Handson Demonstration

Docker availability

- Check the status of docker service

```
$ service docker status
$ service docker start
```

- Check the version of docker available

```
$ docker version
```

- Create a test container from hello-world image

```
$ docker container run hello-world
```

Handson Demonstration

Working on Container

- Know the images available in your machine

```
$ docker image ls
```

- Create a container with interactive access (-it)

```
$ docker container run -it ubuntu:16.04
```

- List the containers

```
$ docker container ls
```

Handson Demonstration

Change the container environment

- Create a folder and file inside container and exit from container

```
:/# mkdir MASTER
:/# touch testfile
:/# exit
```

- Start and connect (attach) with the container again for interaction

```
$ docker container start 301a4c3c816c
$ docker container attach naughty_brown
```

- Find that the environment inside the container is intact

```
:/# ls
```

Handson Demonstration

Replicating the Container

- Extract the image of container

```
$ docker container commit 301a4c3c816c
```

- Create another container from this extracted image

```
$ docker container run -it testimage /bin/bash
```

- Change container names

```
$ docker container rename naughty_brown original
```

Handson Demonstration

Docker Image Sharing as tarfile

- Export the image of a continer as a single tar file

```
$ docker container export -o test.tar original
```

- Import the image from the tar file

```
$ docker image import ./test.tar newimage
```

- Check whether the new container is replica of the former

```
$ docker container run -it --name replica2 newimage /bin/bash
```

Handson Demonstration

Docker image sharing via docker hub

- Create an account in docker hub: (https://hub.docker.com)
- Login into docker hub

```
$ docker login
```

- Change the image tag (Name) as required

```
$ docker image tag testimage prakaram/ti001
```

- Push the image to docker hub

```
$ docker image push prakaram/ti001
```

- Logout from docker hub

```
$ docker logout
```

Handson Demonstration

Docker image sharing via docker hub

- Pulling an image from docker hub

```
$ docker image pull docker/whalesay
```

- Creating a container

```
$ docker container run docker/whalesay cowsay "Always an Important Message"
```

```
 ______________________________ 
< Always an Important Message >
 ------------------------------ 
    \
     \
      \     
                    ##        .            
              ## ## ##       ==            
           ## ## ## ##      ===            
       /""""""""""""""""___/ ===        
  ~~~ {~~ ~~~~ ~~~ ~~~~ ~~ ~ /  ===- ~~~   
       \______ o          __/            
        \    \        __/             
          \____\______/   
```

Handson Demonstration

Removing containers & images

- Removing a container

```
$ docker container rm replica2
```

- Removing all stopped containers

```
$ docker container prune
```

- Removing an image

```
$ docker image rm prakaram/ti001
```

- Removing all images not attached to any containers

```
$ docker image prune -a
```

Handson Demonstration

Docker subcommands

- Docker container specific subcommands

```
$ docker container
```

- Docker Image specific subcommands

```
$ docker image
```

- Old general list of docker subcommands

```
$ docker
```

Thank You