MIRACLE
SOFTWARE SYSTEMS

DevOps with DataPower

Harish Chadalawada

Apigee developer

Miracle Software Systems, Inc.

Venkata Lakshmi

Apigee developer

Miracle Software Systems, Inc.

Agenda

- Introduction
- Prerequisites
- Architecture
- Preparation
- Bootstrapping a DataPower for Docker Project
- Docker build and the DataPower Application
- Containerize the Backend
- Containerize the Client
- Composed DataPower Appllication
- Continuous Integration

Introduction

- Modern application development is fast and getting faster. Manual processes are necessarily being eliminated in favor of automation.

- Automation itself is moving one off tasks maintained by a person or an organization to automation regimes that are hosted elsewhere.

- For developing an application will use DevOps, CI and CD

  methodology on an existing DataPower application.

Continue..

- Here we start with an existing DataPower configuration, turn it into a Docker image and save the configuration as files human-readable “source” files.

- Now, we’ll fully containerize the application by putting both the test client and the other servers into containers and the containers into a composed application with Docker Compose.

Continue..

- Once composed the application, will set up two different ways we can work with it.

  → The first way will be as a developer would – making small changes
  to configuration and source and testing them live.
  → The second way we want to run our composed application is as a standalone test.

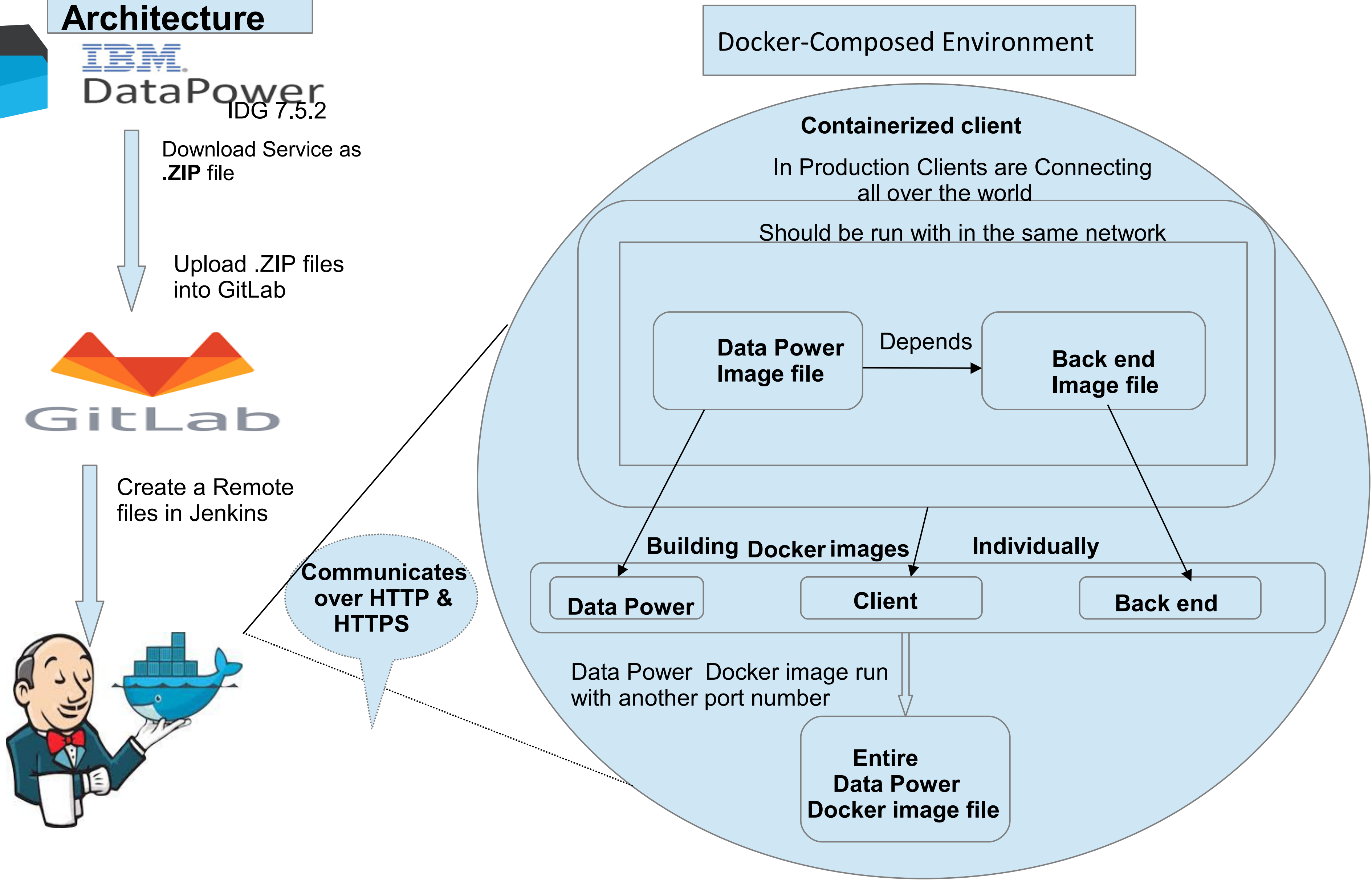
- Now implementation is complete then it is also used by the continuous integration regime to validate the application on every check-in.

- Now we will use Jenkins to perform Continuous Integration(CI) for our Docker Compose application.

Prerequisites

- Ubuntu Virtual Machine
- Docker
- Docker-Compose
- DataPower in Docker
- GitLab
- Jenkins

Architecture


IBM DataPower
IDG 7.5.2
Download Service as .ZIP file
Upload .ZIP files into GitLab
GitLab
Create a Remote files in Jenkins
Communicates over HTTP & HTTPS
Docker-Composed Environment
Containerized client
In Production Clients are Connecting all over the world
Should be run with in the same network
Data Power Image file
Depends
Back end Image file
Building Docker images
Individually
Data Power
Client
Back end
Data Power Docker image run with another port number
Entire Data Power Docker image file

Preparation

- We used Ubuntu machine and it has all the tools required to complete the POC.

- Installed Docker and docker-compose and the official DataPower Docker image is pre-loaded.

- Both Chrome and Firefox are configured to open a tab on the GitHub **Ibm-datapower/interconnect-labs** page.

- This repository contains, among other things, the DataPower application that we will use as the basis for the POC.

Bootstrapping a DataPower for Docker Project

- Now export the zip file from GitHub repository.
- Next setting up the project and create a directory structure for client,datapower and backend with the following command.
- Now use DataPower running inside Docker to create its own configuration for that issue the below command.

```
localuser@ubuntu-base:~/datapower-devops-lab/datapower$ \
docker run -it \
-v $PWD/config:/drouter/config \
-v $PWD/local:/drouter/local \
-e DATAPOWER_ACCEPT_LICENSE=true \
-e DATAPOWER_INTERACTIVE=true \
-e DATAPOWER_WORKER_THREADS=2 \
-p 9090:9090 –p 80:80 –p 443:443 \
--name datapower \ ibmcom/datapower
```

- Now we can see the datapower default log.

Continue..

- Now initialize the DataPower and enable the web management then point the browser at https://localhost:9090.
- Next log in as **admin** with the new password then we can see the DataPower GUI now.
- Create an application domain in UI and import the configuration files from the localuser home directory.
- Then see that all the configuration objects and files were successfully imported and close the import screen.
- This process was to build a DataPower configuration suitable for use in Docker.

Docker build and the DataPower Application

- Now create a Docker file in DataPower directory with the following command and add the text into that file.

```
localuser@ubuntu-base:~/datapower-devops-lab/datapower$ atom Dockerfile
```

- Now we compares the **foo** service in DataPower and the docker run commands we have been using.

- Now we have to build the DataPower docker image. It contains the application called Foo, so we will tag the resulting image as **foo**.

- Now we can check docker images and it will see **foo** in the list and test the build image.

Containerize the Backend:

- Now we have to connect with the backend images. In this tutorial Docker hub has the default image we want to work on.

  https://hub.docker.com/r/hstenzel/nodejs-hostname-automatic/

- Creating Network: **Foo**

Up to now, every container we have run has been in the default Docker network. But what we want for DataPower-foo and the backend to run all by themselves in the same dedicated network.

- Use the command to create the network that uses backend.

Docker network create foo

Containerize the client

- Since containerization is came as complementary for virtualization there is a need to containerize the client.
- Why to containerize the client? While in production clients will join from all over the world, it should run anywhere where docker runs irrespective of operating systems.
- So we need to write a docker file to automate this.
-
```
FROM  ubuntu:latest
 RUN    apt-get update && \
        apt-get –y install curl
 CMD   ["curl","-k","-s",”http://datapower”,”https://datapower”]
```

Composed DataPower Application

- We have three containers to work with, but we dont see these three containers running together.
- Docker-Compose is used to start all the containers depending on the instructions given to it.
- Though we have composed application but we don't have a development environment to access datapower web management.
- To view the web version of datapower start the containers using the command

```
docker-compose -f docker-compose-dev.yml up
```

Continous Integration

- For continuous Integration we have powerful tool Jenkins.
- The DataPower Foo application we are having should have a appropriate Jenkins file that calls Docker-Compose.
- Continuous Integration is possible by adding webhook to GitLab.
- When Jenkins is ready to build the job it pulls the code from Gitlab.
- Webhook is used to automate the build process in Jenkins i.e whenever the code is pushed into gitlab, Webhook alerts Jenkins to automate the job with respect to change in code.

THANK YOU