InterConnect 2016
The Premier Cloud & Mobile Conference

Session: Lab 7369

ERRATA DOCUMENT

Medical Minecraft

Lab Instructions

Authors:
Srinivas Cheemalapati, IBM Cloud Advisor, email (sriniva@us.ibm.com)

February 21 – 25
MGM Grand & Mandalay Bay
Las Vegas, Nevada

***February 2016* edition**

NOTICES

This information was developed for products and services offered in the USA.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
United States of America

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

TRADEMARKS

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at “Copyright and trademark information” at www.ibm.com/legal/copytrade.shtml.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Cell Broadband Engine is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license therefrom.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

IT Infrastructure Library is a Registered Trade Mark of AXELOS Limited.

ITIL is a Registered Trade Mark of AXELOS Limited.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Linear Tape-Open, LTO, the LTO Logo, Ultrium, and the Ultrium logo are trademarks of HP, IBM Corp. and Quantum in the U.S. and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Resource guide

IBM Technical Client Training has enhanced its training capabilities, and extended reach into new cities and countries, by partnering with five highly qualified IBM Business Partners who provide high quality, authorized training for IBM Clients, IBM Business Partners, and IBM employees. IBM continues to provide authorized training curriculum and content, and also maintains overall ownership of the IBM Training ecosystem.

The delivery of public, private and customized training to IBM Clients and IBM Business Partners is now done by the IBM Global Training Providers (GTPs):

- Arrow
- Avnet
- Global Knowledge
- Ingram Micro
- LearnQuest

See ibm.com/training for information on the classes that the GTPs offer.

Completing this InterConnect lab is a great first step in building your IBM skills. IBM offers several resources to keep your skills on the cutting edge. Resources available to you range from product documentation to support websites and social media websites, including the following examples:

- IBM Training website
  - Bookmark the IBM Training website for easy access to the full listing of IBM training curricula. The website also features training paths to help you select your next course and available certifications.
  - For more information, see http://www.ibm.com/training
- IBM Certification
  - Demonstrate your mastery of IBM products to your employer or clients through IBM Professional Certification. Certifications are available for developers, administrators, and business analysts.
  - For more information, see http://www.ibm.com/certify

InterConnect 2016

The Premier Cloud & Mobile Conference

February 21 – 25
MGM Grand & Mandalay Bay
Las Vegas, Nevada

Objective of the Interconnect 2016
Medical Minecraft Labs (Total Time 2 hours)

- *Get acquainted with cloud technologies like Bluemix PaaS in the context of Game integration using Docker Container, Watson Cognitive services using Java Plugin, all from a single platform to support new game design.*

- *How to create a container locally and invoke Minecraft to connect to the unmodified Minecraft Server*

- *Learn how to run Spigot Minecraft Server Container in the cloud specifically the Bluemix PaaS environment*

- *Create a Minecraft Server Mod Plugin that responds to commands from the Minecraft Client and while the Spigot Server is running in the Bluemix PaaS Cloud.*

- *Learn how to develop a Plugin to integrate with Watson Cognitive services such as Watson Dialog Service as a example. (If time permits we will execute the code end to end, otherwise we review how the code works and see a demo)*

- *Get a perspective of how a Medical Minecraft Game is created using the above technologies*

Important Information for the LAB

Prerequisites to successfully complete the full Lab:

STOP: *Need a Bluemix Account*

STOP: *Need a Minecraft client registration and username Login:*

STOP: *Need access to Bluemix containers service.*

Actions if pre-requisites are not met:

- *Create new account at: https://console.ng.bluemix.net/*
  - *Need access to Bluemix containers service.*

    Important Information:

    - ***Define a namespace for your IBM Containers repository***

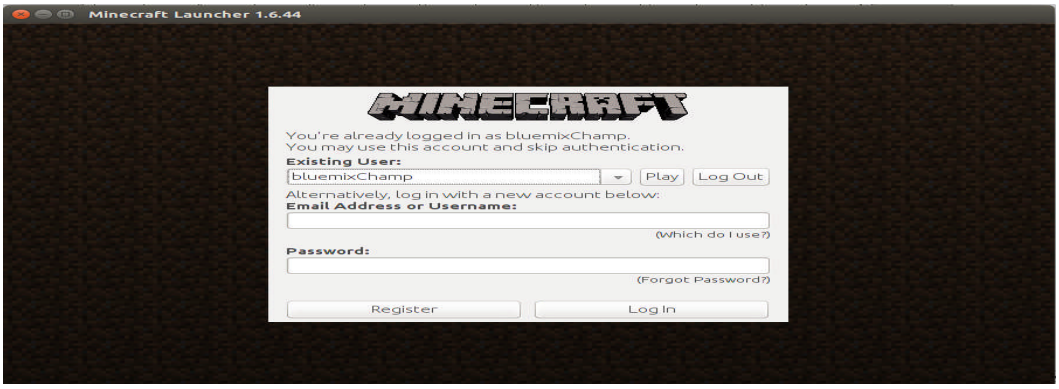
      In IBM Containers for Bluemix, the private Bluemix repository is a central repository within an organization to store your trusted Docker images. You can push and pull images from the repository, and you can deploy these images to any development, staging, or production environment.

      The first time that you create a container within an organization, you are prompted to enter a name for the namespace that is associated with the private Bluemix repository. The namespace is used to generate a unique URL that you use to access your private Bluemix repository. The URL is required whenever you perform an action, such as a pull request or a push request of an image, to the repository.

      The following rules apply to private repository names:

      1. *The name cannot be changed after it is set for an organization.*
      2. *The name must contain only lowercase letters, numbers, and underscores.*
      3. *The name must start with at least one letter or number.*
      4. *The name must be between 4 and 30 characters in length.*

- *Purchase a Minecraft client and register at https://minecraft.net/ your username for login.* ***Minecraft Client is not provided in Lab***



Lab Virtual Machine Information

Login: minecraft

Password: minecraft

Note: In order to save build time and increase the learning time in the lab, the longest running jobs that generate build outputs may be pre-populated by masking the execution of the commands in the scripts.

Software used:

- *Eclipse MARS*
- *Java JDK 1.7*
- *Bluemix PaaS Platform*
- *Bluemix Dialog Service*
- *Custom Minecraft Texture Packs simulating Medical Minecraft –demo only*

Setting up the lab – scripted for you. Execute the following:

- *Login to the Linux VM*
- *cd ~ (go to home directory);*
- *git clone* https://github.com/scheema/ilabs
- *type -> cd ~/ilabs*
- *type -> chmod +x ilab.sh*
- *run ilabs.sh → type → ./ilab.sh*

This will create all the appropriate folders needed for the project

---

Create a Bluemix app and obtain Watson Dialog credentials

- We need to obtain an instance of the Dialog service and get some credentials from that service instance in order to use them in our new plugin. Then we'll build the plugin and install it into Docker and push that into the IBM Container.
- To obtain the relevant credentials to access the Dialog instance in Bluemix, we have to follow a roundabout process

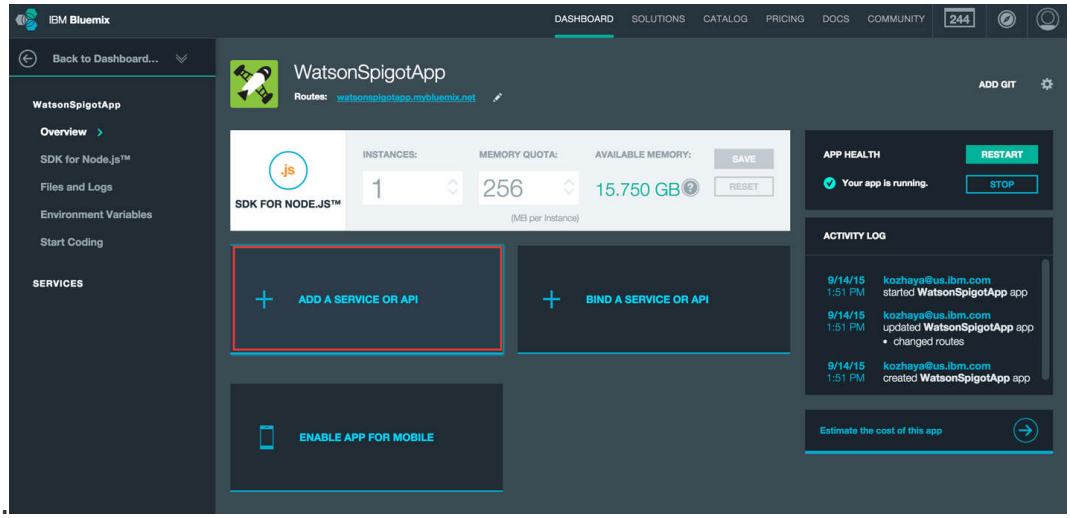
Log in to Bluemix using your IBM id and password. (If you don't have an IBM id, you can get one when you register for your free Bluemix trial account.)

1. Create a new app by clicking **CREATE APP > Web > SDK for Node.js > Continue**.

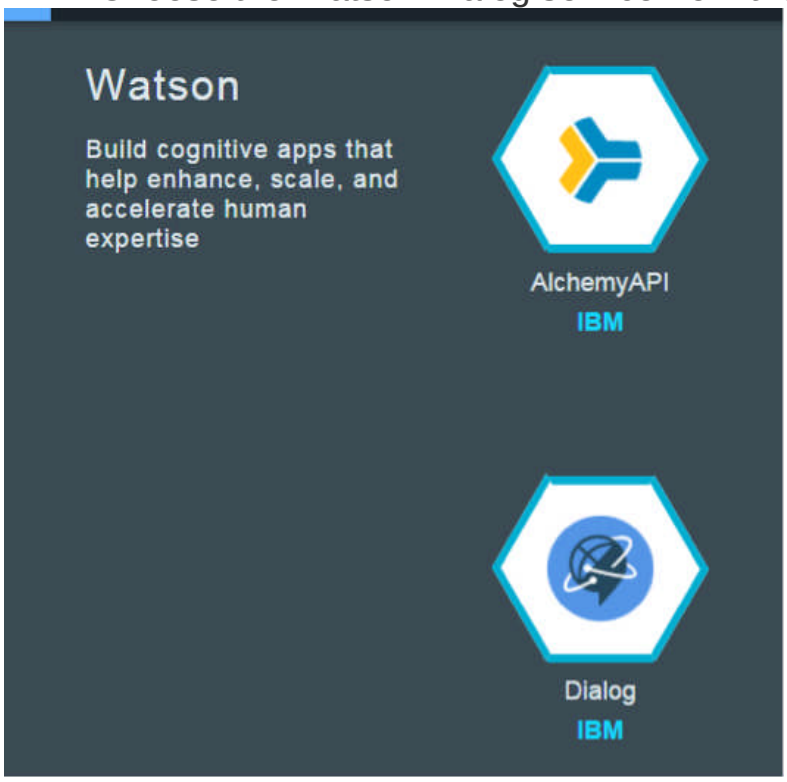
Specify an APP Name that is unique (we use WatsonSpigotApp in our example). Once WatsonSpigotApp is created, return to the Overview page (by clicking **Overview** in the left column).

2. Add the Watson service to the app you just created by clicking **ADD A SERVICE OR API**

The following screen shows the detail



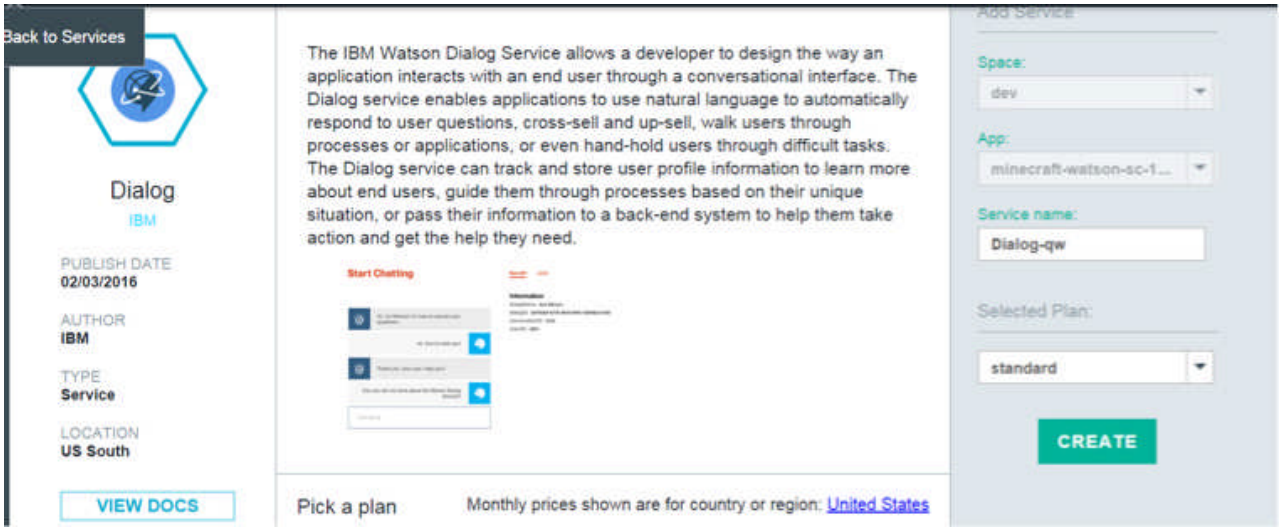
3. Choose the Watson Dialog service from the catalog services



4. Click **USE** on the next pop-up to add the new service.

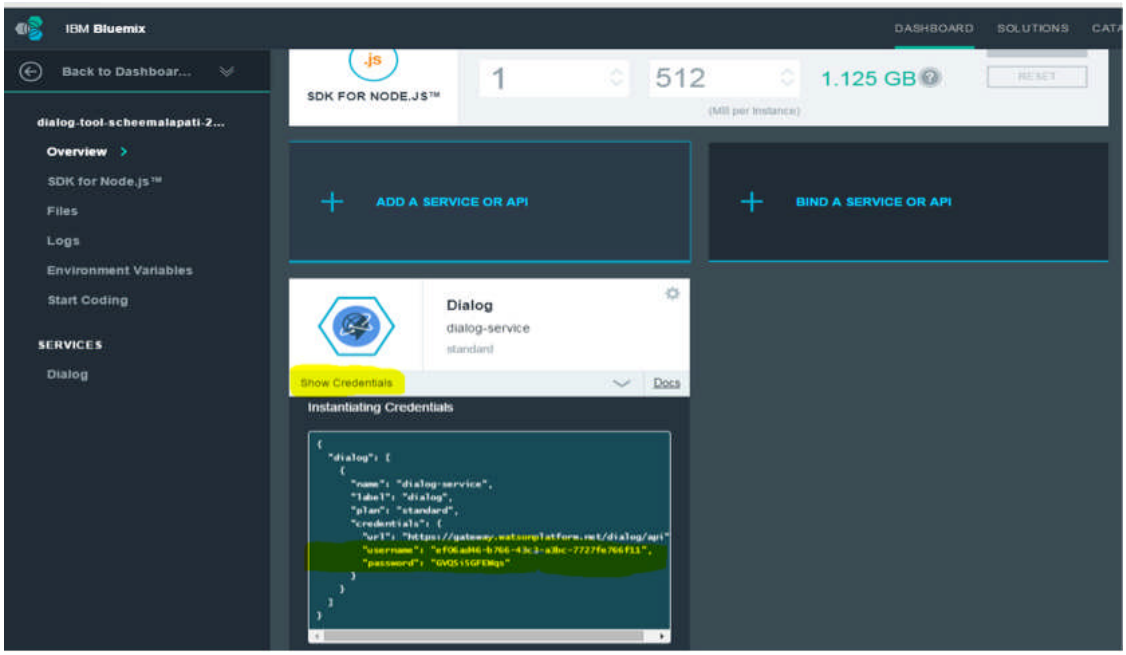
The following screen shows the detail

it.



5. Return to the WatsonSpigotApp overview by clicking **Overview** in the left column. Note the newly added Dialog service.

Click **Show Credentials** under that service to see the credentials needed to access this provisioned instance of the Dialog Service.



6. Note: These credentials (specifically, the URL, username, and password) will be needed and will be filled into the lab_ids.txt file. This file is in the ~/ folder (linux home directory)

```
"url": "https://gateway.watsonplatform.net/question-and-answer-beta/api"
"username": "some-long-hyphenated-set-of-characters"
"password": "AVeryLOngSer1esOfCharact3rs"
```

Fill the credentials.

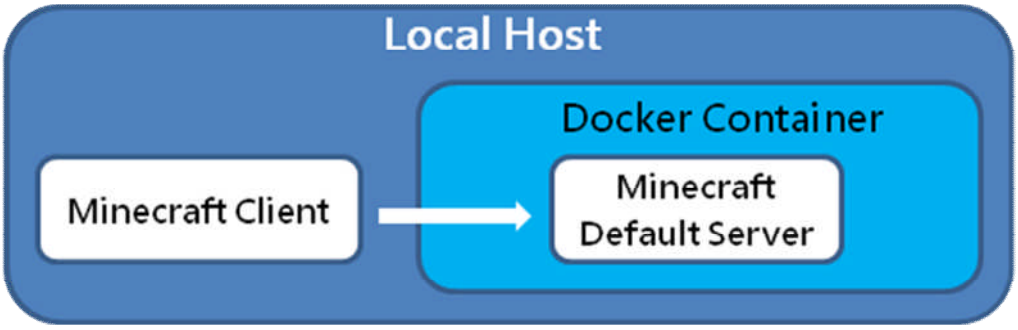
Open a Terminal -> type gedit & → open file ~/lab_ids.txt

Follow the instructions in the file lab_ids.txt to capture the information required for the eclipse project.

LAB 1 - Time to complete instructions 30 min (not build time)

Objective: *Install all pre-requisite, kick off with a basic docker file that hosts the default minecraft server (not extensible) version. Login from Minecraft Client and check the game is working from the docker container.*

Overview:



a) Speaker overview of the objective of Lab1 and expectations

b) Kick of build; review and understand Lab 2 while wating

c) Login to the VM (no quotes) with username "minecraft" and password "minecraft"

d) Go to folder minecraft-projects

```
minecraft@ubuntu:~/minecraft-project$ ls
bluemix-minecraft-master  README.md   spigot            spigot-plugin-bluemix
cleanup.sh                setupcf.sh  spigot-1.8.3.jar  spigot-plugin-watson
minecraft                 setup.sh    spigot-plugin     SpigotPlugin.zip
minecraft@ubuntu:~/minecraft-project$ cd minecraft
minecraft@ubuntu:~/minecraft-project/minecraft$ ls
dockerfile
minecraft@ubuntu:~/minecraft-project/minecraft$
```

e) cd minecraft

f) sudo docker build –t minecraft/minecraft188 .

a. *Build time : ~ 8 minutes (output is below)- Review Lab2 while building.*
b. *Run "docker images" and successful build as below*

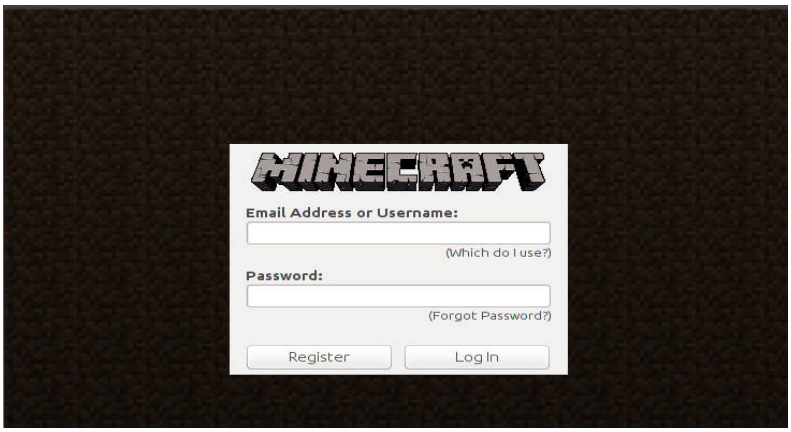
g) Start minecraft client with your credentials

a. *Go to ~/minecraft-client*

b. *Issue: ./run.sh – this will start the minecraft-client*

```
minecraft@ubuntu:~/minecraft-project/minecraft$ docker images
REPOSITORY                TAG                 IMAGE ID            CREATED
minecraft/minecraft188    latest              f75357dec2b2        7 minutes ago
ubuntu                    14.04               6cc0fc2a5ee3        2 weeks ago
```

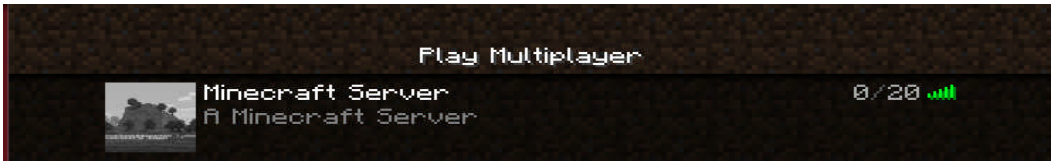
h) Login with your credentials



i) Open terminal and get the IP address of the VM by "ifconfig eth0" and note ip: example - 192.168.87.128

j) You should see the following screen

k) Select Multiplayer -> Add Server (use IP address got earlier) and use port 25565 as follows: <your earlier ip>:25565. The server should start as follows



LAB 2 - Time to complete instructions 70 min (not build time)

Commands that will be used in the lab:

- *To set your namespace, use this command:*

```
cf ic namespace set <namespace>
```

5. *To find out what your namespace is, use this command:*

```
cf ic namespace get
```

6. To list the images in the Bluemix repository:

```
cf ic images
```

7. To show running containers in Bluemix:

```
cf ic ps
```

8. To show all containers in Bluemix:

```
cf ic ps –a
```

9. To stop a running container:

```
cf ic stop CONTAINER
```

10. To remove the image from the Bluemix repository:

```
cf ic rmi IMAGE
```

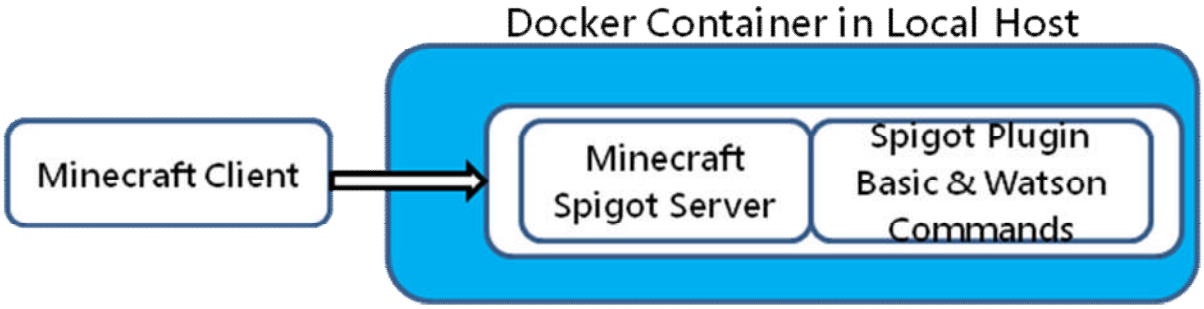
11. To remove a container from the Bluemix repository:

```
cf ic rm CONTAINER
```

LAB 2 IS IN 2 PARTS – PART 1

Part 1*: Deploy Spigot Server with Watson Plug-In Code into Local server –Commands entered from the Minecraft Client will get responses from Watson.*

Objective: To demonstrate how to deploy a Spigot server into a Localhost Docker Container that has the Plug-In to recognize basic commands from the Minecraft client and talk to Watson to get the responses. We will use Eclipse environment and go through the code that enables this functionality to respond to commands from the client. We will go through the important files that are required to support this. At the end of this task you will be familiar with understanding the Plug-In Code, how to run a docker container with Spigot Server integrated with Plug-In that communicates with Watson over through Internet.



Files that will be reused instead of building to save time:

craftbukkit-1.8.8.jar, spigot-1.8.8.jar, the Java Project code that can be imported into Eclipse and script files.

We will create a watsonqa.jar file from eclipse that will be exported to the ***directory ~/bluemix-minecraft/minecraft-projects/spigot-watson-bluemix where the dockerfile for build resides.*** *.*

The local.sh file creates the environment for the server to run. The Minecraft Client can then access the Spigot Server with Watson Plugin locally running in the container.

Part 2 of the lab we will Deploy Spigot Server with Plugin integration with Watson services and then deploy this into Bluemix Container that could be used for hosting.

Objective: Install all pre-requisite, kick off with a basic docker file that hosts the default minecraft server (not extensible) version. Login from Minecraft Client and check the game is working from the docker container.

Step 1:

Speaker overview of the objective of Lab2 PART 1 and expectations

a) The git clone that was performed in home directory initially will have a folder ***minecraft@ubuntu:~/bluemix-minecraft/WatsonDialogSpigot***

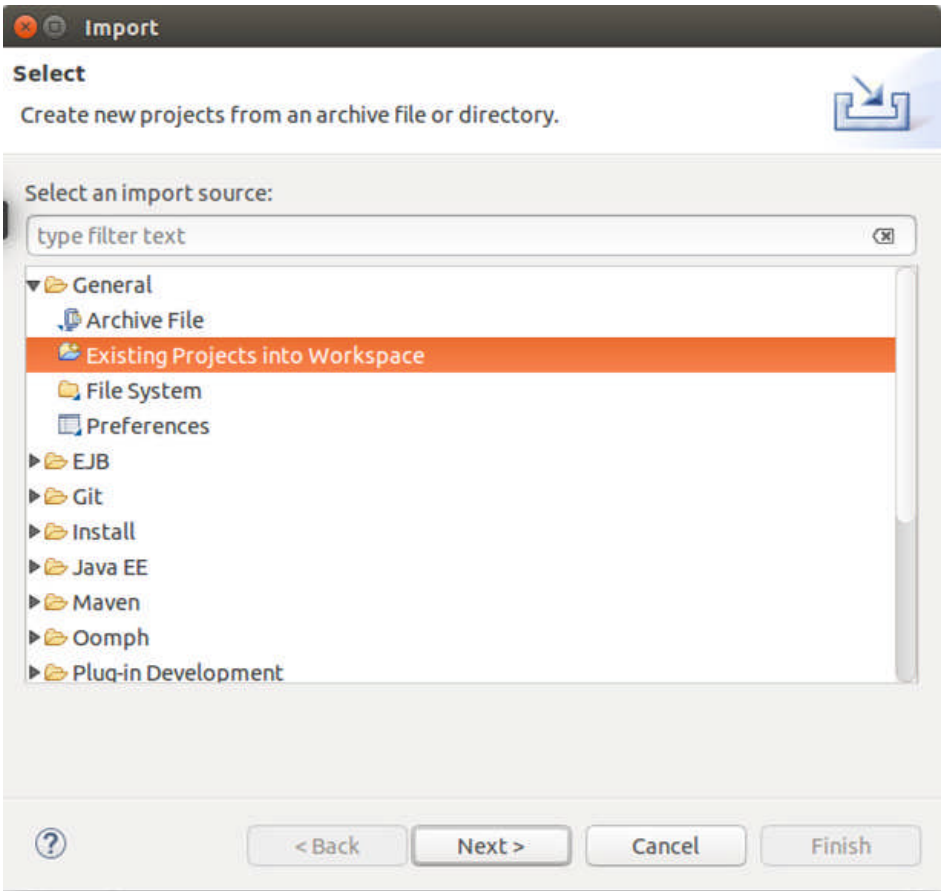
b) Open terminal, cd ***~/bluemix-minecraft/WatsonDialogSpigot***

c) start eclipse –" eclipse &"

d) Type the workspace location ***"/home/minecraft/workspace"***

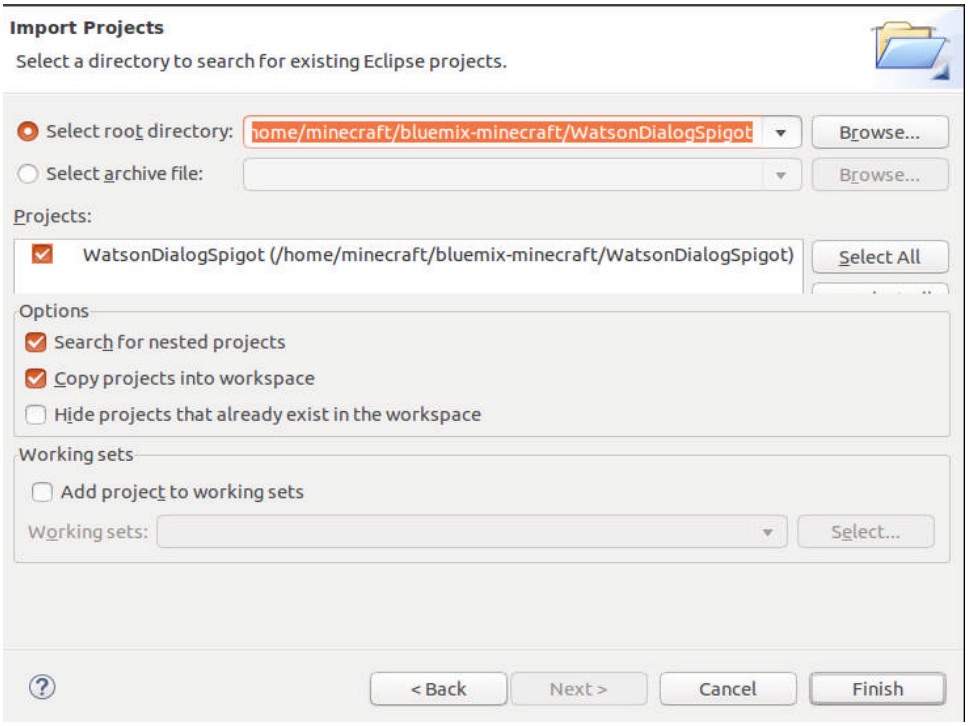
e) Select ***File > Import*** *from the menu at the top of the Eclipse Workbench.*

f) *Select* ***General***



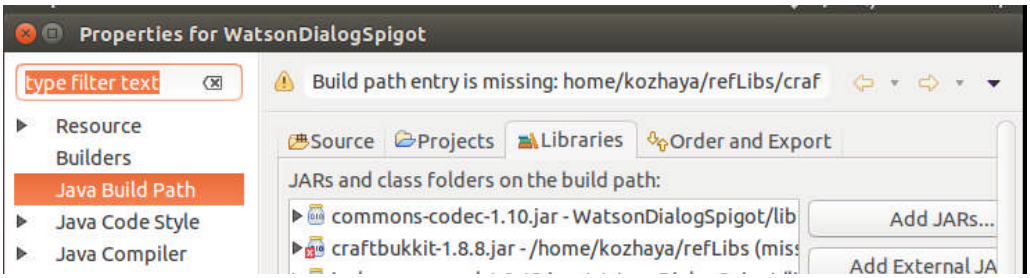
g) ***-> Existing Projects into Workspace****, then click* ***Next.***

h) *Select* ***Select Root Directory****,* ***browse to folder ~/bluemix-minecraft/WatsonDialogSpigot (~ is /home/minecraft)***



i) *Below fig shows the project Structure.. We need to update the build path with craftbukkit-1.8.8.jar file to get rid of the error*

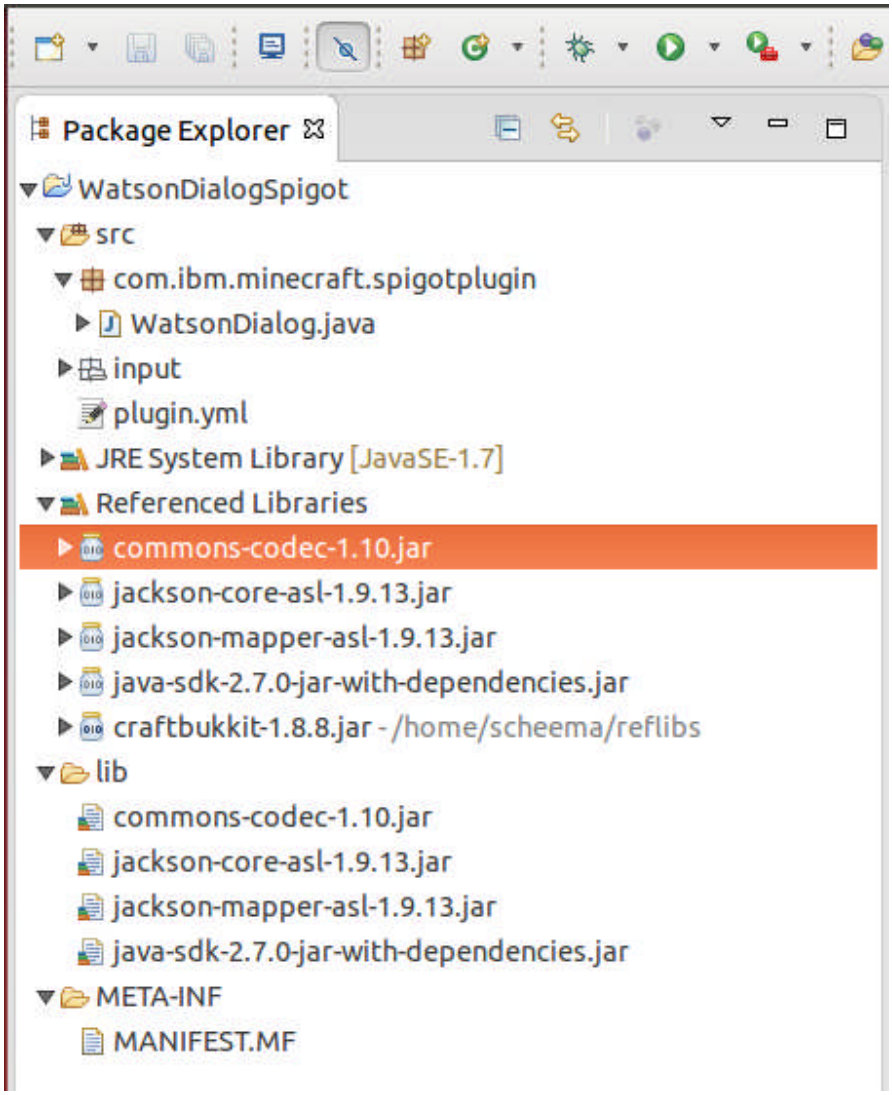
j) *. Right click project in Project Explorer "WatsonDialogSpigot" -> Build Path -> Configure Build Path*



Need to replace the "X" marked library referencing to a new location. ***Click the "x" library*** *and* ***click "Remove". Select Add External JARs.***

In the Pop Up window Select craftbukkit-1.8.8.jar *from* ***/home/reflibs*** *folder****. Click OK***

The following will be the folder structure.

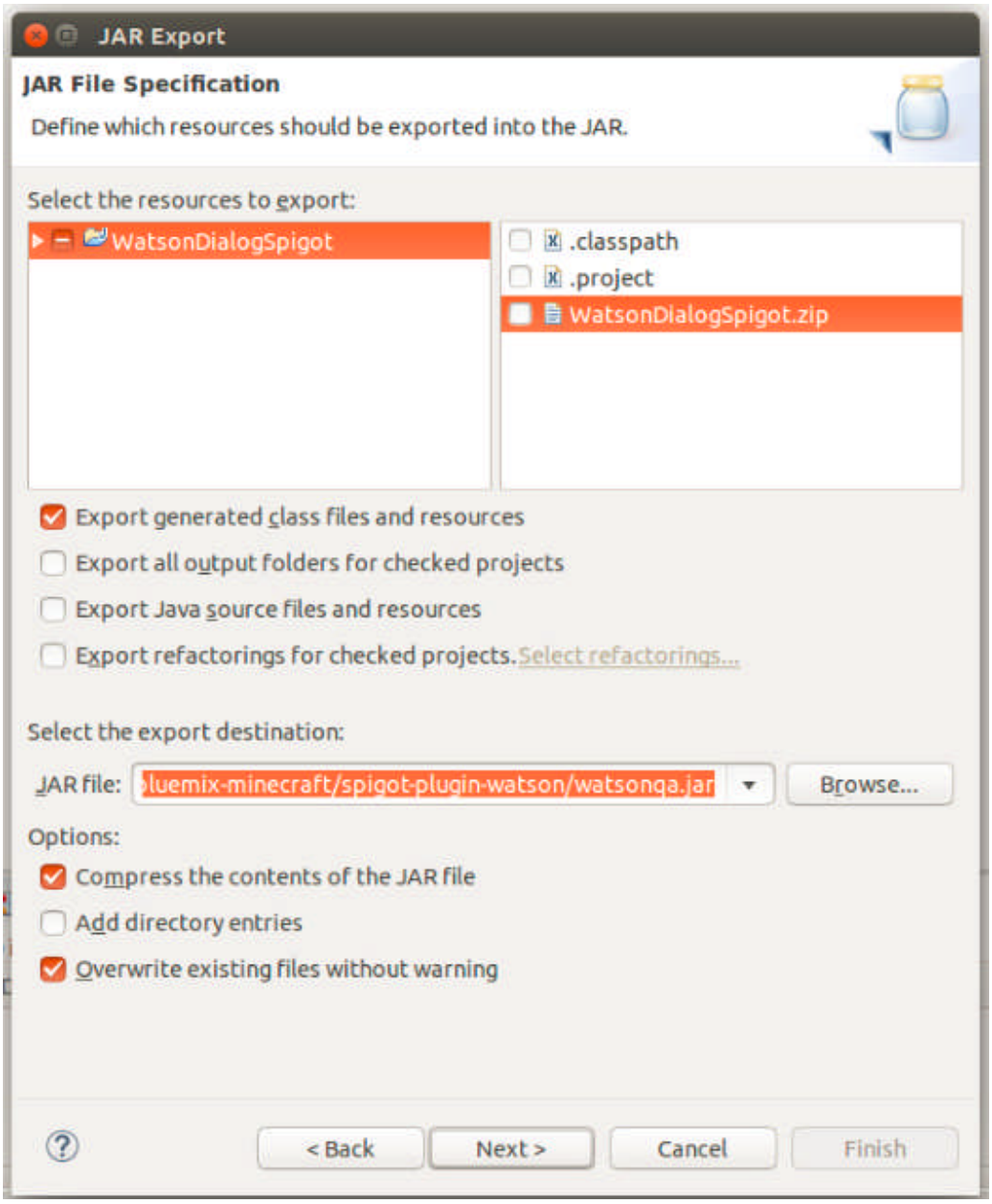


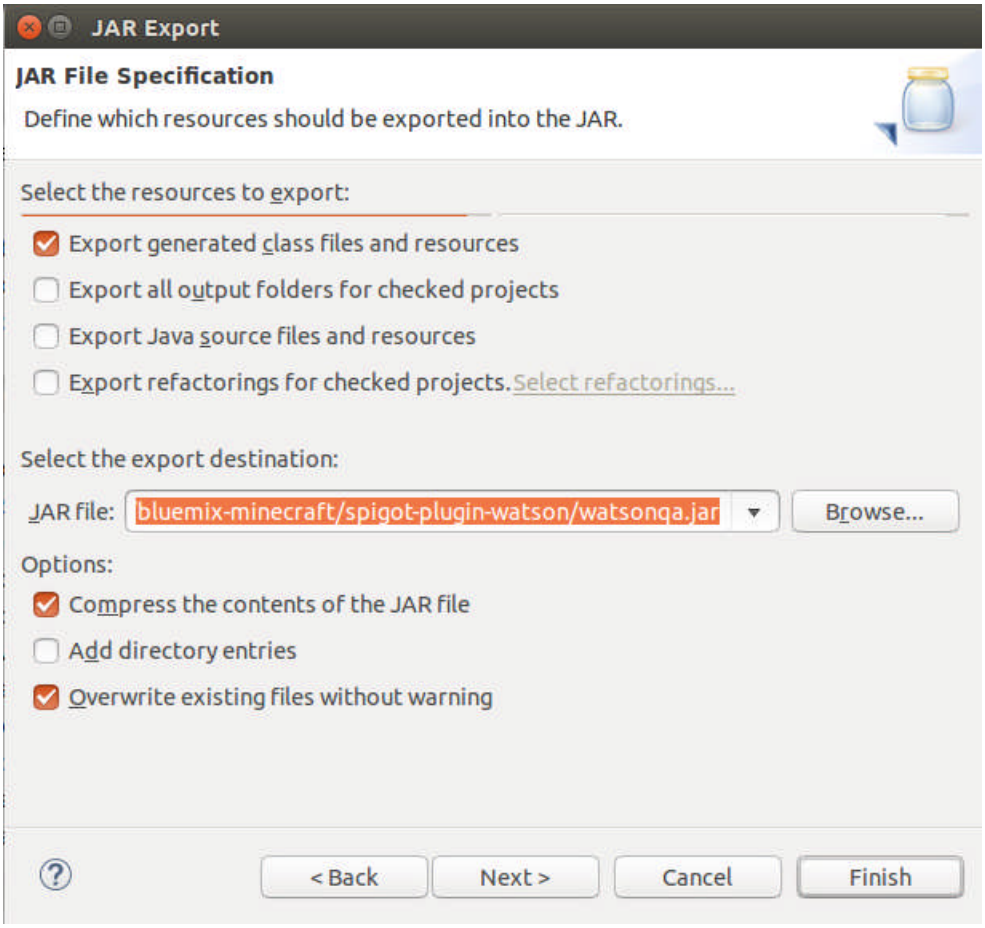
Review the source code, file structure and files plugin.xml, DialogTest.xml files

k) In Eclipse, select File -> Export -> Java -> JAR file. Click Next.

l) In the Jar file -> browse to ~/bluemix-minecraft/spigot-plugin-watson/watsonqa.jar (~ is /home/minecraft)

m)


JAR Export
JAR File Specification
Define which resources should be exported into the JAR.
Select the resources to export:
WatsonDialogSpigot
.classpath
.project
WatsonDialogSpigot.zip
Export generated class files and resources
Export all output folders for checked projects
Export Java source files and resources
Export refactorings for checked projects. Select refactorings...
Select the export destination:
JAR file:
luemix-minecraft/spigot-plugin-watson/watsonqa.jar
Browse...
Options:
Compress the contents of the JAR file
Add directory entries
Overwrite existing files without warning
< Back
Next >
Cancel
Finish

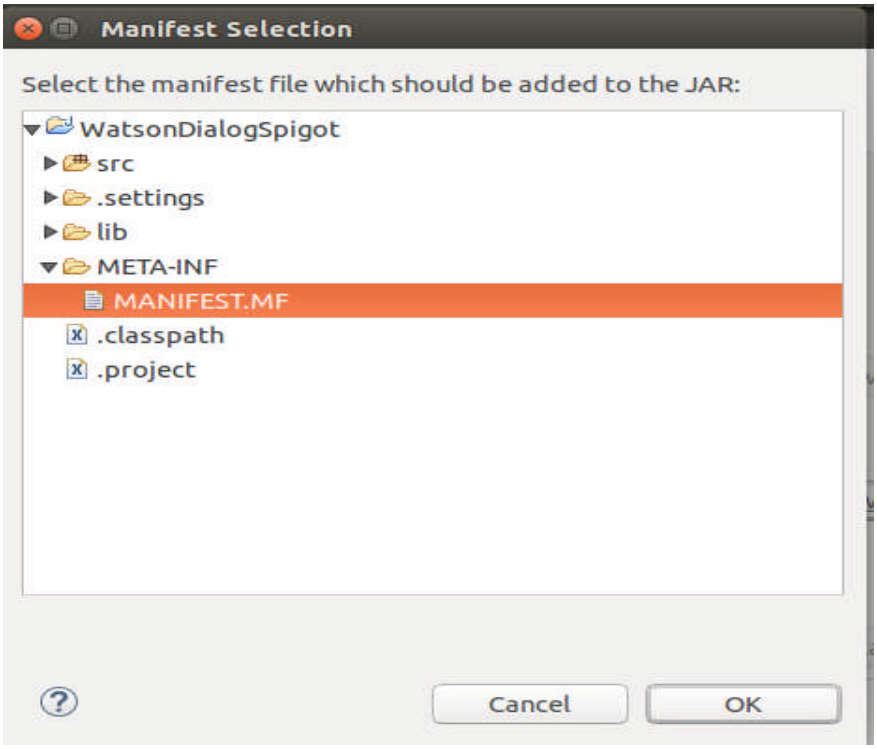


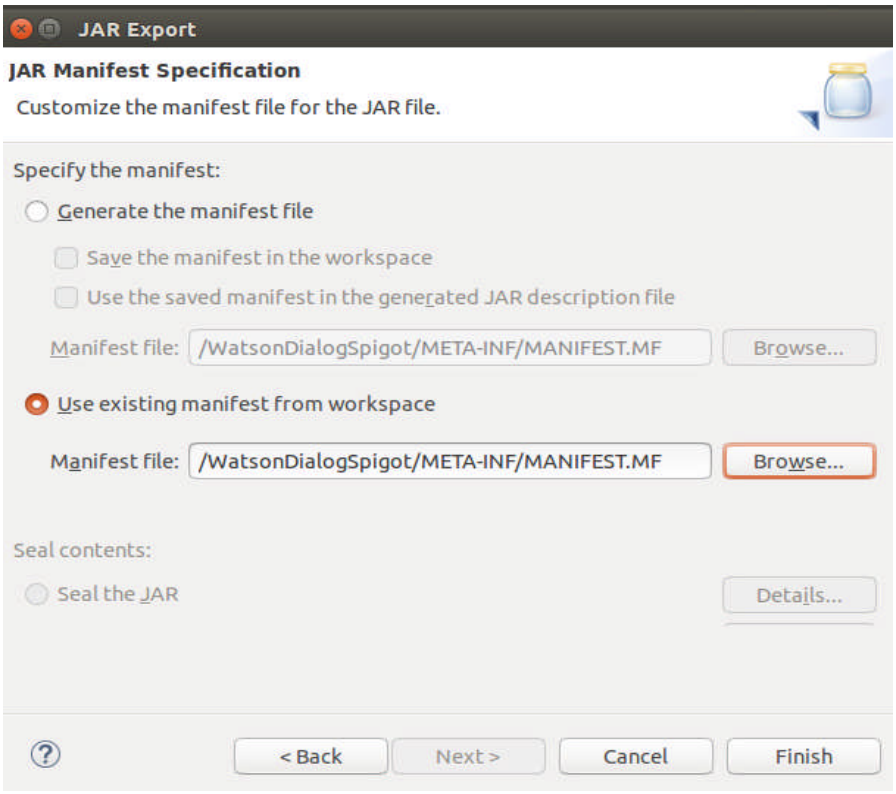
Check per the above figure.

Click Next

Select the Manifest File *that is under the project at path WatsonDialogSpigot -> META-INF -> MANIFEST>MF*

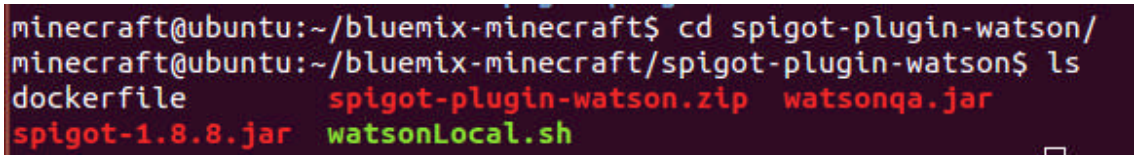
The following figures can be used as the reference.





Click Finish

*The new **watsonqa.jar** file from the previous step will be at the following path.*



Time to Test the Watson QA service

a) At the terminal type "cd"

b) Type cd bluemix-minecraft/spigot-plugin-watson

c) Build the docker image: Go To folder

~/bluemix-minecraft/spigot-plugin-watson

d) Type "docker build -t minecraft/watsonpluginspigot . (need .)

<< This will take about 12 to 15 minutes >>

e) Run the docker container locally

docker run –i –t –p 25565:25565 minecraft/watsonpluginspigot

LAB 2– PART 2

1. Host the Spigot server with the WatsonDialog plugin in Bluemix

   So far, we've verified the functionality of the Spigot server with the Watson plugin locally in a Docker container. Next, let's host this server on Bluemix.

1. In a Linux terminal, create a new directory $HOME/watsonspigotbluemix:

```
mkdir $HOME/watsonspigotbluemix
```

2. Copy watsonqa.jar that we extracted into $HOME/watsonspigot:

```
cp $HOME/watsonspigot/watsonqa.jar $HOME/watsonspigotbluemix/
```

The following steps are needed to run to deploy in Bluemix

3. Log in to Bluemix using your Bluemix id, password, and dev space (no need to set namespace again if it is already set):

```
cf login
cf ic login
cf ic namespace set <your name space registered>
```

4. Build a new Docker image:

```
docker build -t watsonspigotbluemix .
```

5. Tag the created image:

```
docker tag watsonspigotbluemix registry.ng.bluemix.net/scheemalapati/watsonspigot
```

The screen will look like this after tagging. Please replace with your namespace

```
minecraft@ubuntu:~/watsonspigotbluemix$ docker images
REPOSITORY                                      TAG                 IMAGE ID            CREATED             VIRTUAL SIZE
watsonspigotbluemix                             latest              617ebabc6a4e        5 hours ago         724.4 MB
registry.ng.bluemix.net/scheemalapati/watsonspigot   latest         617ebabc6a4e        5 hours ago         724.4 MB
ubuntu                                          14.04               6cc0fc2a5ee3        2 weeks ago         187.9 MB
minecraft@ubuntu:~/watsonspigotbluemix$
```

6. Push the Docker image to Bluemix:

```
docker push registry.ng.bluemix.net/<replace your name space>/watsonspigot
```

```
minecraft@ubuntu:~/watsonspigotbluemix$ docker push registry.ng.bluemix.net/scheemalapati/watsonspigot
The push refers to a repository [registry.ng.bluemix.net/scheemalapati/watsonspigot] (len: 1)
617ebabc6a4e: Pushed
6a46abf01632: Pushed
b68cffacb3bc: Pushed
017354ccf8ec: Pushed
48b62152cd55: Pushed
87ab0d2feb43: Pushed
ead996144475: Pushed
e8632671f7f8: Pushed
0e968bdc1b97: Pushed
7c4d88bf3ddb: Pushed
9afccbe2e7e4: Pushed
c13323c1b08f: Pushed
9605c412b605: Pushed
06b93faad447: Pushed
e8d242388900: Pushed
dd121bc0aa4e: Pushed
cd892d0d38c6: Pushed
f80999a1f330: Pushed
2ef91804894a: Pushed
92ec6d044cb3: Pushed
latest: digest: sha256:f3d0dcc1445d0d871c5ba8c297623a6d3913ec6018eec668b21e2fe9958d801c size: 34058
minecraft@ubuntu:~/watsonspigotbluemix$
```

7. Run Docker container on Bluemix as a single command in step 10:

```
cf ic run --name=watsonspigot  -p 9085 registry.ng.bluemix.net/<
replace your name space > /watsonspigot
```

When this runs successfully, it returns an id as below.

```
minecraft@ubuntu:~/watsonspigotbluemix$ cf ic run --name watsonspigot -p 9085 registry.ng.bluemix.net/scheemalapati/watsonspigot
ae418305-9318-4168-a913-8ed1a1e10e33
minecraft@ubuntu:~/watsonspigotbluemix$
```

8. Check if an IP address is already allocated with: cf ic ip list

```
IpAddress      ContainerId
169.44.2.89    ae418305-9318-4168-a913-8ed1a1e10e33
minecraft@ubuntu:~/watsonspigotbluemix$
```

If the container ID already matches to step **10 – skip step 12, Step 13**

9. Request an available IP address: (note – run only if Step 10 not satisfied)

```
cf ic ip request
```

```
parallels@ubuntu:~/watsonspigotbluemix$ cf ic ip request
Successfully obtained ip: "134.168.16.121"
```

10. Bind the returned IP address to the Bluemix Docker container you ran in step 10: (note – run only if Step 10 not satisfied)

```
cf ic ip bind <ip address> watsonspigot
```

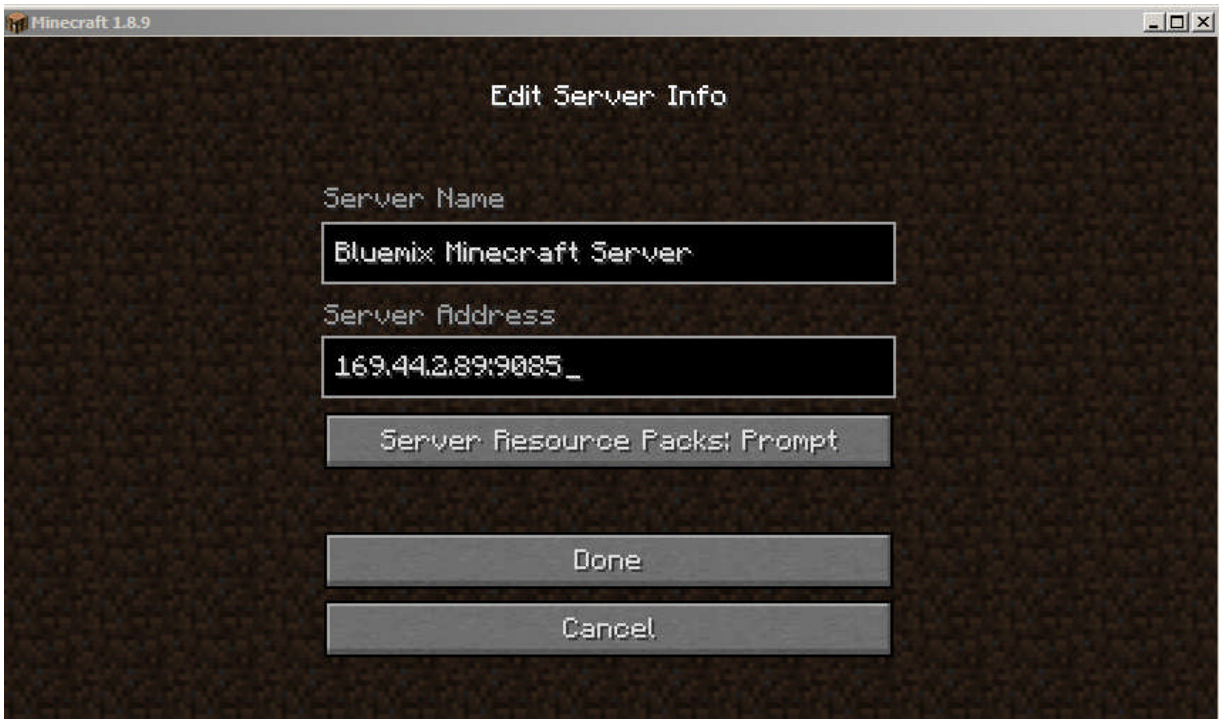
```
parallels@ubuntu:~/watsonspigotbluemix$ cf ic ip bind 134.168.16.121 watsonspigot
Successfully bound IP
```

11. Verify that the IP address is bound to the Bluemix Docker container:

```
IpAddress      ContainerId
169.44.2.89    ae418305-9318-4168-a913-8ed1a1e10e33
minecraft@ubuntu:~/watsonspigotbluemix$
```

At this point it is Time to test the Docker Container hosted in Bluemix

12. Connect the client to the Spigot server hosted on Bluemix. Note the IP address in the rectangle and the port number in the oval.



13. Verify functionality by asking Watson a question:

```
/watson
/watson what is diabetes
/Watson what is flu
/Watson what is measles
```

References:

Read through the Tutorials in the next page without having to worry about the commands. This will enable you to get an overview of what is being performed in the lab. We will go through this material it in the lab.

Print the articles for better reading and reference (optional0

The TUTORIAL LINKS THAT WILL BE USED IN THE LAB and the description are provided below.

Create cognitive plugins for Minecraft with Bluemix, Docker, and Watson

Part 1: Running Minecraft servers within Docker

Link:

http://www.ibm.com/developerworks/cloud/library/cl-bluemix-minecraft-docker-trs-1/index.html

---

Create cognitive plugins for Minecraft with Bluemix, Docker, and Watson

Part 2: Building plugins for Minecraft with Docker and Eclipse

Link:

http://www.ibm.com/developerworks/cloud/library/cl-bluemix-minecraft-plugins-trs-/index.html

---

Create cognitive plugins for Minecraft with Bluemix, Docker, and Watson
Part 3: Deploy Spigot Minecraft servers on Bluemix

Link:

http://www.ibm.com/developerworks/cloud/library/cl-bluemix-minecraft-spigot-trs-3/index.html

---

Create cognitive plugins for Minecraft with Bluemix, Docker, and Watson

Part 4: Integrating Watson into Minecraft

Link:
http://www.ibm.com/developerworks/cloud/library/cl-bluemix-minecraft-watson-trs-4/index.html