InterConnect 2016

The Premier Cloud & Mobile Conference

Original Session: Lab 7369

Medical Minecraft

Lab Instructions - Reusable

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

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

***February 2016* edition**

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

IBM

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

Important Information for the LAB – Assumption Docker running in Ubuntu Linux

---

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.*

STOP: *Need a local docker installation on your working machine.*

STOP: *Software Needed*

- *Eclipse MARS*
- *Java JDK 1.7*
- *Git*
- *Cloud Foundry CLI*

Actions for Bluemix pre-requisites if not met:

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

      Important Information:

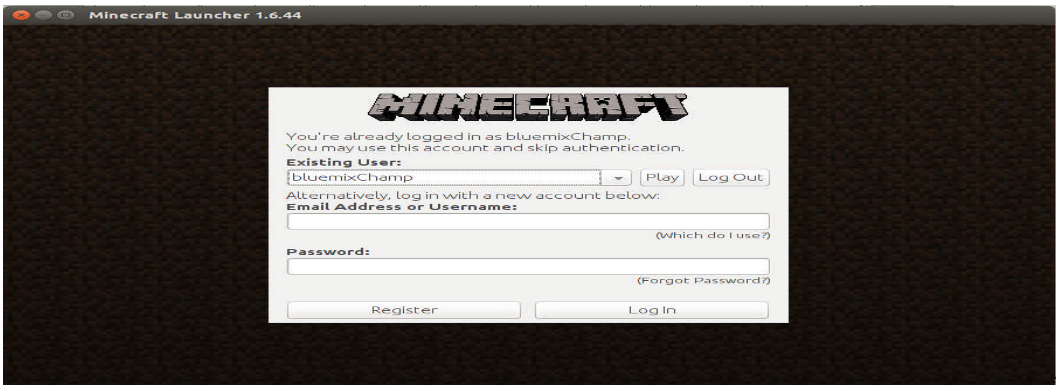
    - ***Define a namespace for your IBM Containers repository***

      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***



You're already logged in as bluemixChamp.
You may use this account and skip authentication.
Existing User: bluemixChamp

Alternatively, log in with a new account below:
Email Address or Username:
(Which do I use?)
Password:
(Forgot Password?)

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 – Execute the following:

- *git clone https://github.com/scheema/bluemix-minecraft*
- *cd bluemix-minecraft*

Create a Bluemix app and obtain Watson Dialog credentials

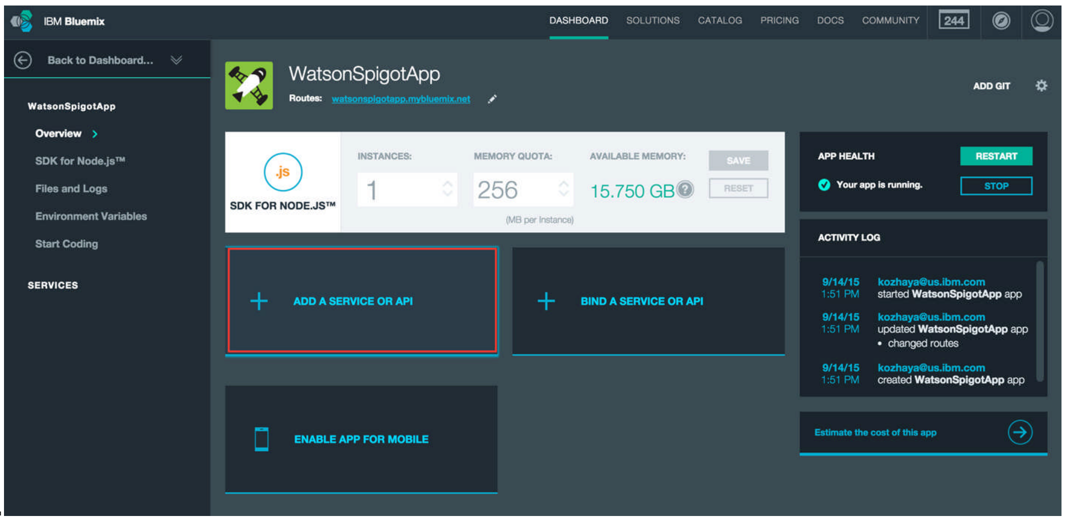
- We need to obtain an instance of the Dialog service and get some credentials from that service instance in order to use them in our new plugin. Then we'll build the plugin and install it into Docker and push that into the IBM Container.
- To obtain the relevant credentials to access the Dialog instance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



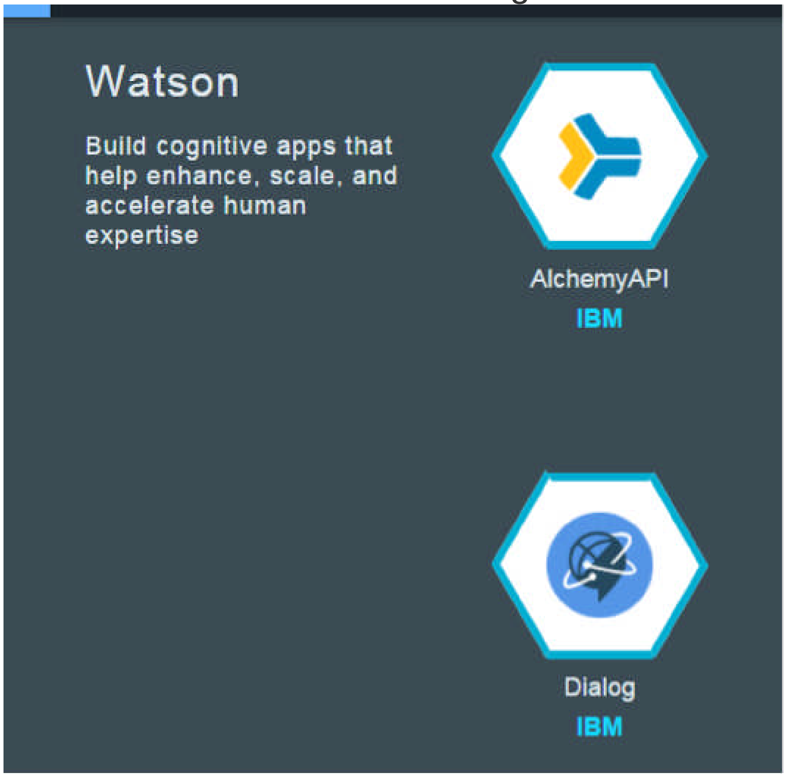
9/14/15 1:51 PM kozhaya@us.ibm.com started WatsonSpigotApp app

9/14/15 1:51 PM kozhaya@us.ibm.com updated WatsonSpigotApp app
• changed routes

9/14/15 1:51 PM kozhaya@us.ibm.com created WatsonSpigotApp app

.

3. Choose the Watson Dialog service from the catalog services

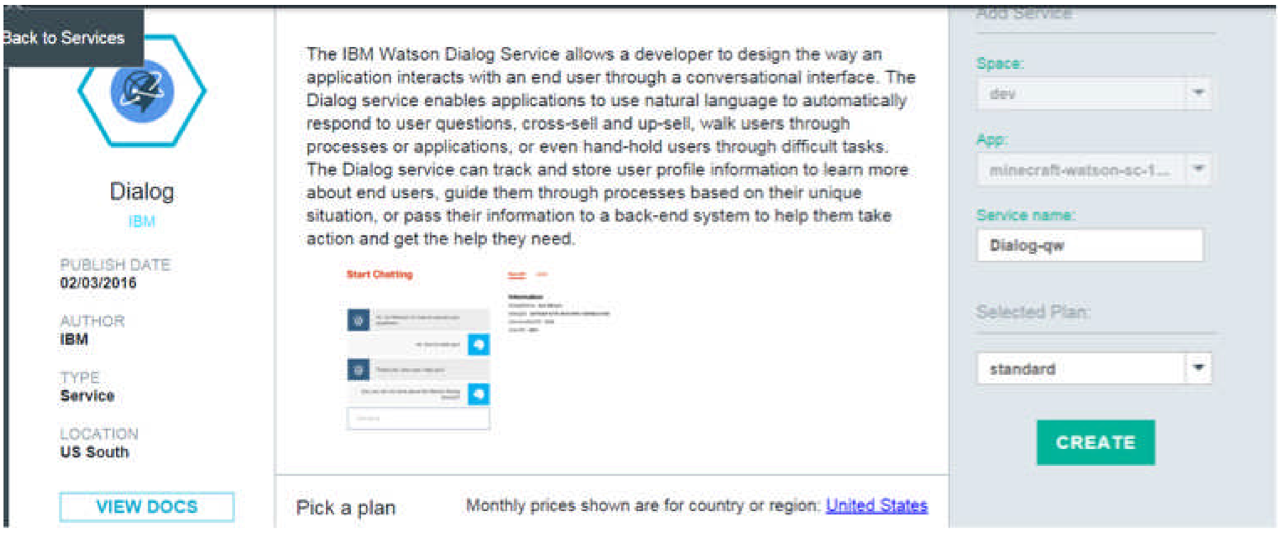

Watson

Build cognitive apps that help enhance, scale, and accelerate human expertise

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

The following screen shows the detail

it.


Back to Services

Dialog
IBM

PUBLISH DATE
02/03/2016

AUTHOR
IBM

TYPE
Service

LOCATION
US South

VIEW DOCS

The IBM Watson Dialog Service allows a developer to design the way an application interacts with an end user through a conversational interface. The Dialog service enables applications to use natural language to automatically respond to user questions, cross-sell and up-sell, walk users through processes or applications, or even hand-hold users through difficult tasks. The Dialog service can track and store user profile information to learn more about end users, guide them through processes based on their unique situation, or pass their information to a back-end system to help them take action and get the help they need.

Pick a plan   Monthly prices shown are for country or region: United States

Add Service

Space: dev

App: minecraft-watson-sc-1...

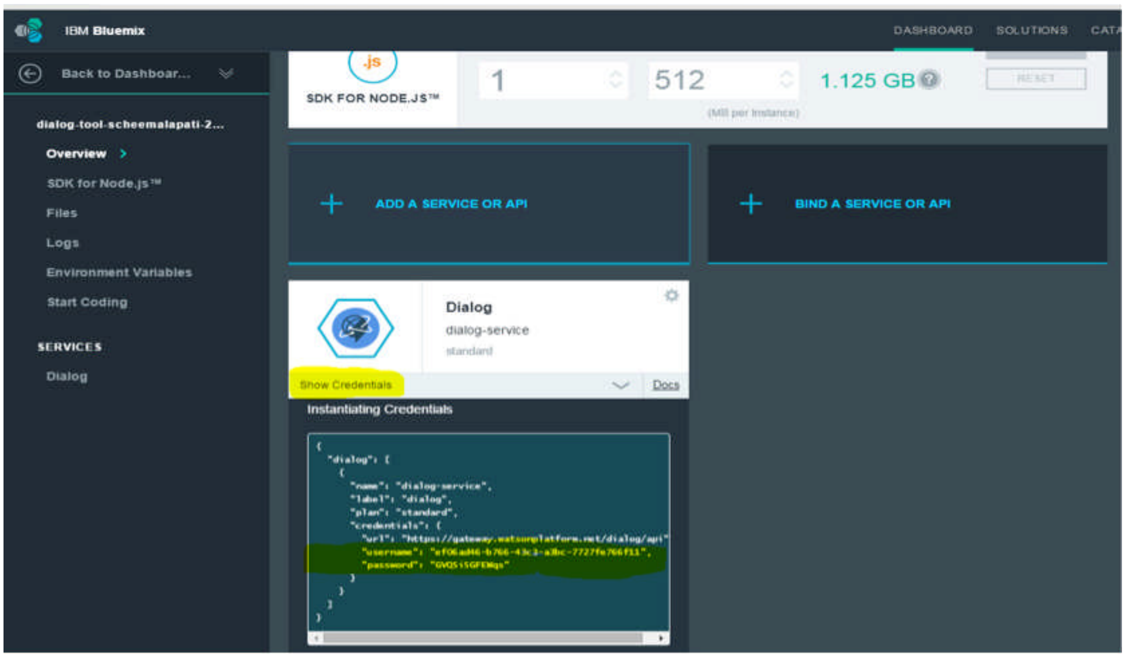
Service name: Dialog-qw

Selected Plan: standard

CREATE

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.



Instantiating Credentials

```
{
  "dialog": [
    {
      "name": "dialog-service",
      "label": "dialog",
      "plan": "standard",
      "credentials": {
        "url": "https://gateway.watsonplatform.net/dialog/api",
        "username": "ef06ad46-b766-43c3-a3bc-7727fe766f11",
        "password": "GVQ5iSGFEWqs"
      }
    }
  ]
}
```

6. Note: These credentials (specifically, the URL, username, and password) will be needed and will be filled into the “lab.txt” file you create. This file is just a easy way of storing the information to be used later in the project.

```
"url": "https://gateway.watsonplatform.net/question-and-answer-beta/api"
"username": "some-long-hyphenated-set-of-characters"
"password": "AVeryL0ngSer1esOfCharact3rs"
```

Create a dialog and capture the Watson Dialog service dialog_id

Use the CURL command line to create the dialog and capture the Dialog id as follows.
Change directory to minecraft-project:

```
cd ~/minecraft project
```

Type the `curl` command as shown; if you're copying it from a web page, make sure of the formatting of the double quotes:

```
curl -X POST -u "bxxxx69b-6xx1-45xx-axx9-03xxxxxxx925":"35533rrNN" --form name=kyletest --
form file=@./spigot-plugin-watson/input/DialogTest.xml
https://gateway.watsonplatform.net/dialog/api/v1/dialogs
```

You will see the following result:

```
{"dialog_id": "exxxxx2-0xx2-4bxc-xxf-48fffdddddd34"}
```

Fill the credentials.

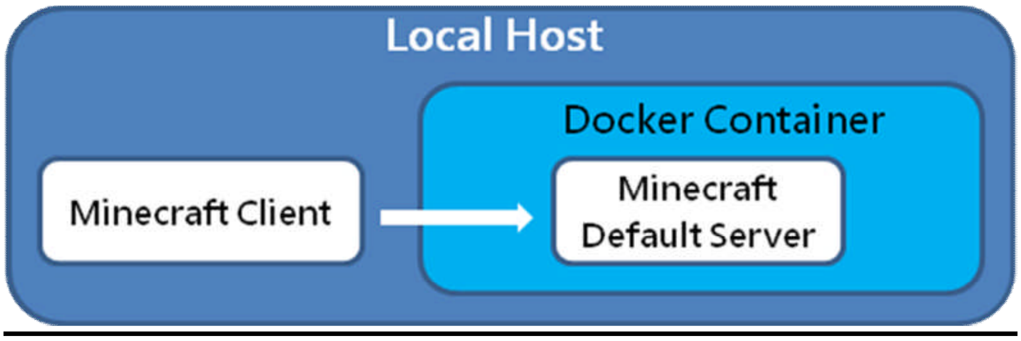
Open a Terminal -> type gedit & → open file ~/lab_ids.txt

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:



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

Install Docker in Ubuntu – if not already done

1. Enter the following commands into the "terminal" application in Ubuntu to install Docker 1.7.0:

```
wget -qO- https://get.docker.com/ | sed 's/lxc-docker/lxc-docker-1.7.0/' | sh
```

2. The first command downloads Docker and installs it into Ubuntu. However, in the default installation, Docker requires you to run Docker commands with root privileges. To avoid that and use a different id, run the following command:

```
sudo usermod -aG docker <your nonroot login id>
```

3. After you issue the command, log out of Ubuntu and log back in.

Install the Cloud Foundry command line tools – if not already downloaded

Next you need to install the Cloud Foundry command line tools. We won't use those in this first tutorial, but we'll begin using them in the third tutorial, and it's a good idea to install them now while you're setting up your Ubuntu image. If you're on any other platform, you'll need to go toGitHub to install the correct tools in your browser, but this procedure will work in Ubuntu.

1. At the command line, type the following line to download the Cloud Foundry command line tools:

```
2.          wget "https://cli.run.pivotal.io/stable?release=debian64&version=6.12.3&
source=github-rel" -O cf-cli_amd64.deb
```

3. Type the following at the command line to install the tools:

```
sudo dpkg -i cf-cli_amd64.deb
```

4. Install the IBM plugin for the Cloud Foundry tools that will allow you to upload and run Docker images later. Do that in Ubuntu by typing:

```
cf install-plugin https://static-ice.ng.bluemix.net/ibm-containers-linux_x64
```

5. After you install the IBM Containers plugin, you need to restart Docker. To do that, issue the following command in your terminal:

```
sudo service docker restart
```

Now that you've downloaded and installed both Docker and the Cloud Foundry tools for Docker (which will allow you to do some interesting things with both!), you're ready to download the sample code that we'll examine in the rest of this tutorial and in the following two tutorials. The sample code is available in git, and you can download it with the following command:

```
git clone  https://github.com/kgb1001001/minecraft-project.git
```

a) cd bluemix-minecraft

b) cp spigot-plugin/dockerfile SpigotPlugin

c) cd SpigotPlugin

d) cp Tutorial.jar HelloWorld.jar

e) Edit dockerfile & append the line as follows
RUN java -jar minecraft/BuildTools.jar --rev 1.8.8

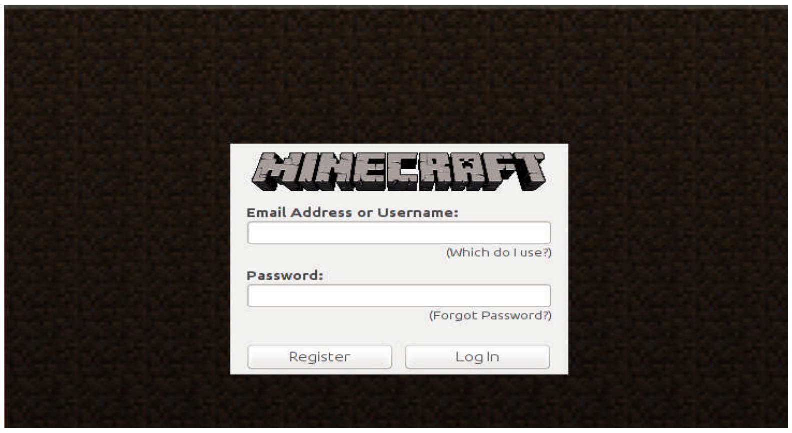
f) sudo docker build –t <login id>/spigot188-plugin **.**

a. Build time : ~ 8 minutes (output is below)- Review Lab2 while building.

b. Run "docker images" and successful build as below

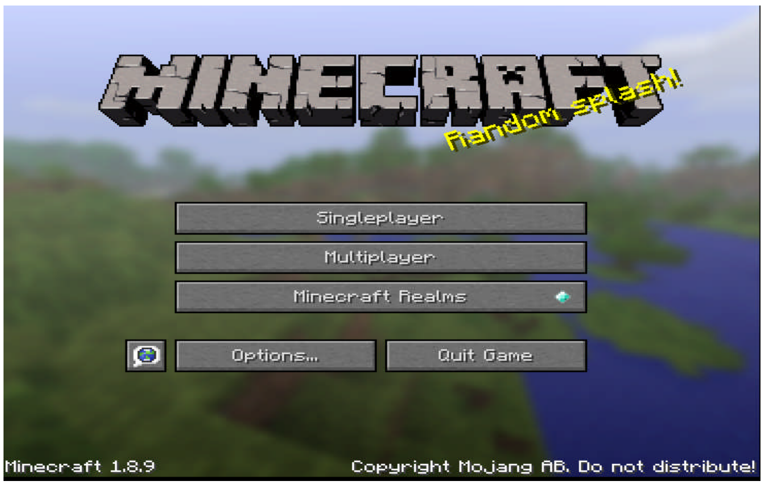
g) Start minecraft client with your credentials

h) Login with your credentials – In the edit profile select the version 1.8.8 instead of the latest version

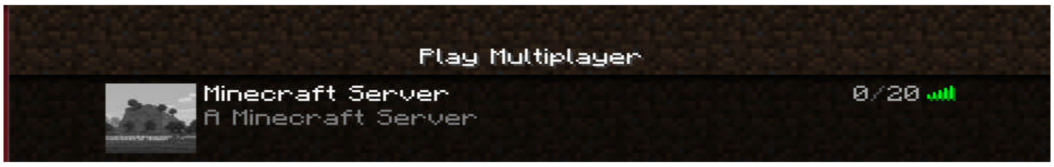


i) *Open terminal and get the IP address of the VM by "ifconfig eth0" and note ip: example - 192.168.87.128*

j) *You should see the following screen*



k) *Select Multiplayer -> Add Server (use IP address got earlier or 127.0.0.1) 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
```

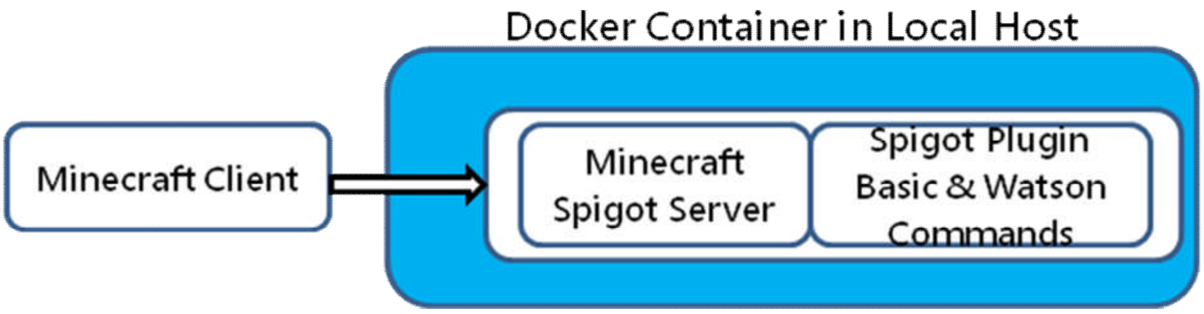
11. To remove a container from the Bluemix repository:

```
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.*


Docker Container in Local Host
Minecraft Client
Minecraft Spigot Server
Spigot Plugin Basic & Watson Commands

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 re-sides.*** *.*

The local.sh file creates the environment for the server to run. The Minecraft Client can then ac-cess 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:

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

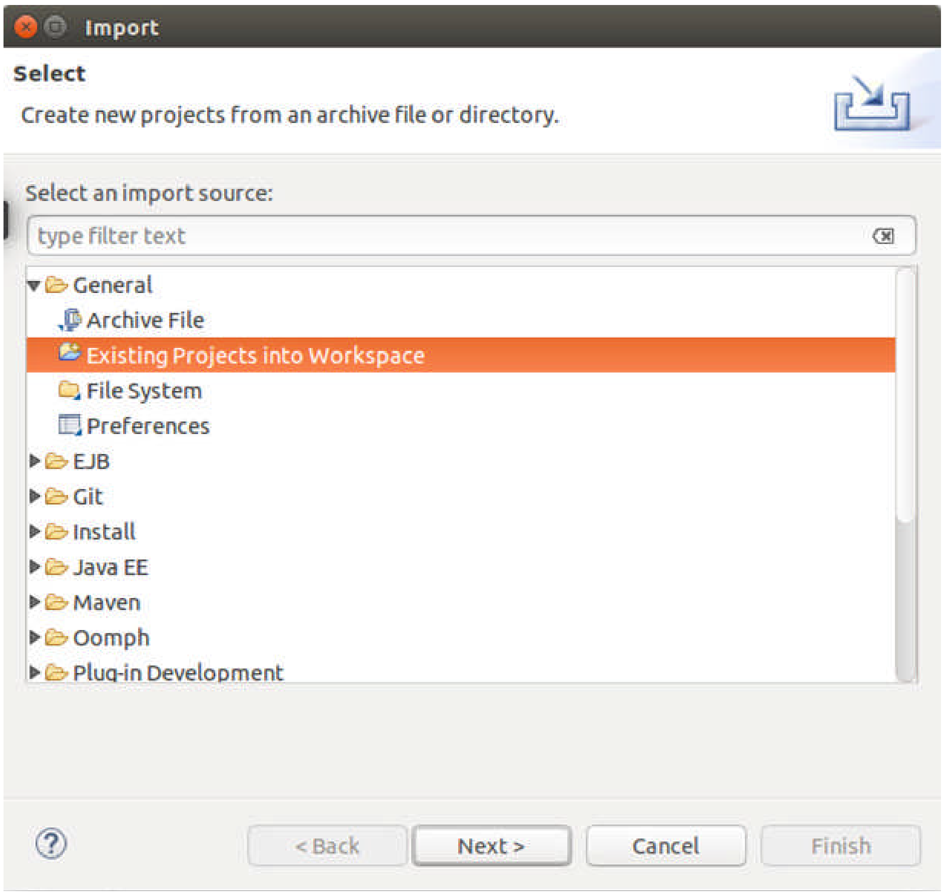
b) *Open terminal, cd* ***~/bluemix-minecraft/WatsonDialogSpigot***

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***


Import

Select

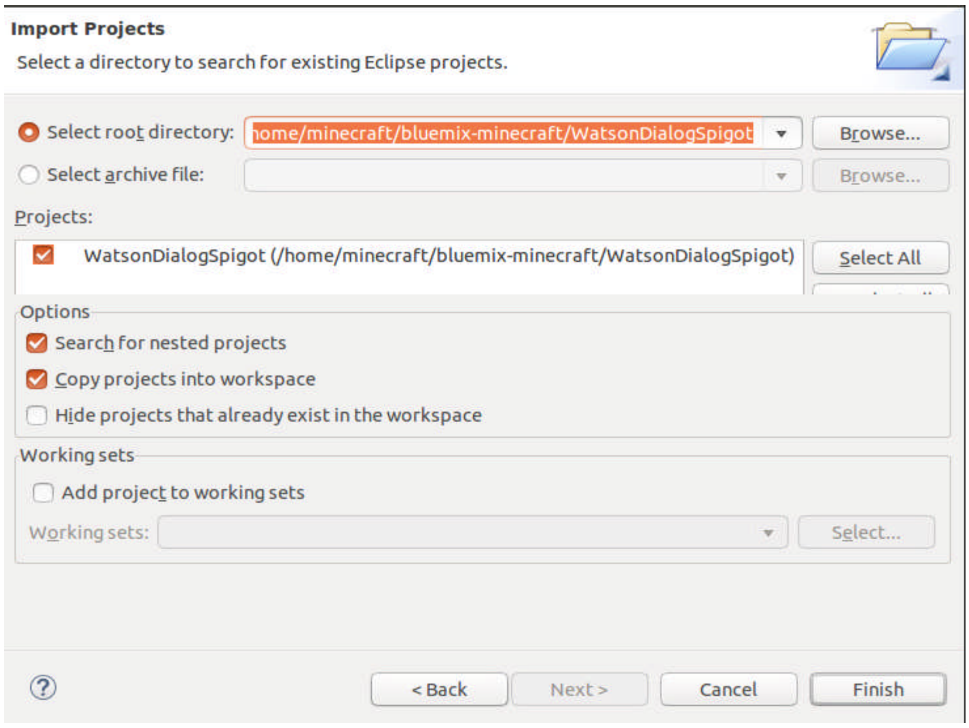
Create new projects from an archive file or directory.

Select an import source:

type filter text

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

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



Import Projects

Select a directory to search for existing Eclipse projects.

Select root directory: ome/minecraft/bluemix-minecraft/WatsonDialogSpigot

Select archive file:

Projects:

WatsonDialogSpigot (/home/minecraft/bluemix-minecraft/WatsonDialogSpigot) ✓

Options

✓ Search for nested projects

✓ Copy projects into workspace

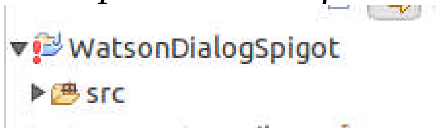
Hide projects that already exist in the workspace

Working sets

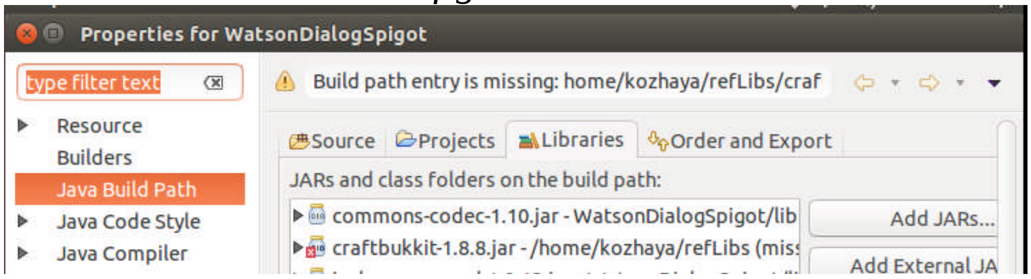
Add project to working sets

Working sets:

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 "WatsonDialogSpig-ot" -> Build Path -> Configure Build Path*



Properties for WatsonDialogSpigot

Build path entry is missing: home/kozhaya/refLibs/craf

JARs and class folders on the build path:

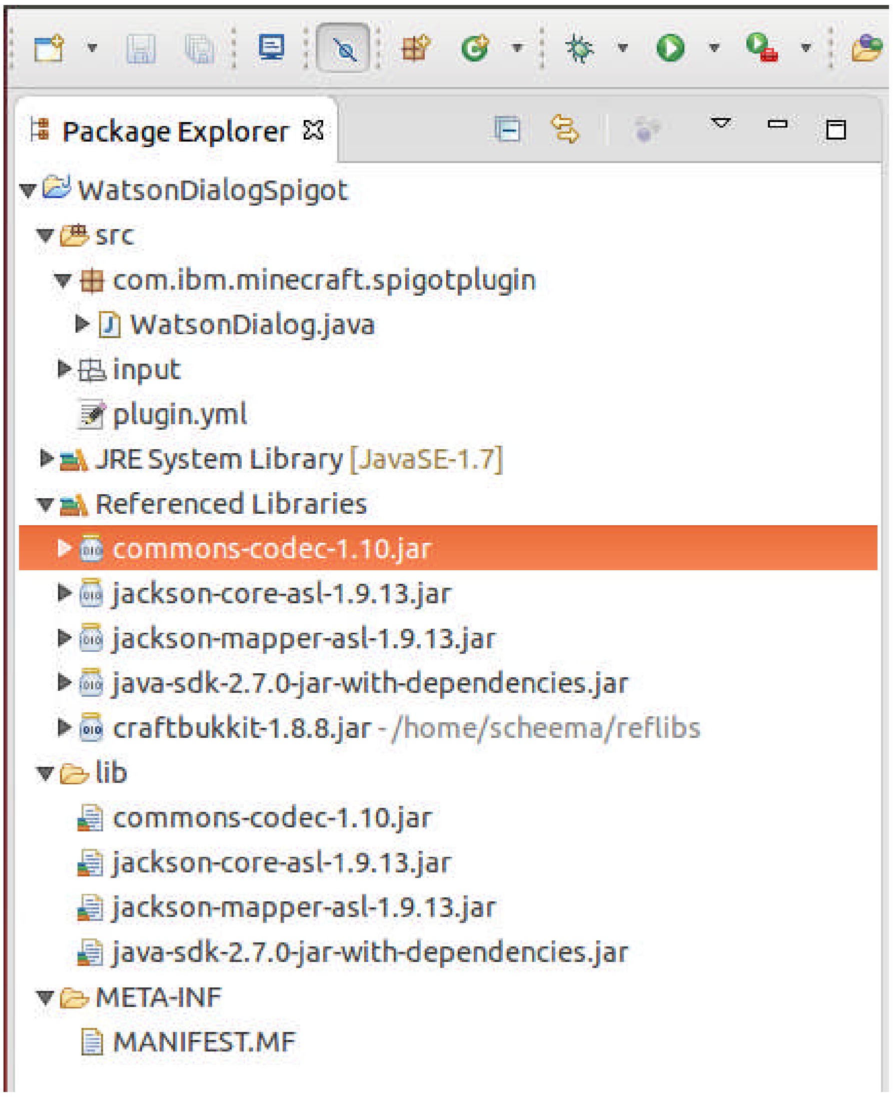
commons-codec-1.10.jar - WatsonDialogSpigot/lib

craftbukkit-1.8.8.jar - /home/kozhaya/refLibs (miss

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.

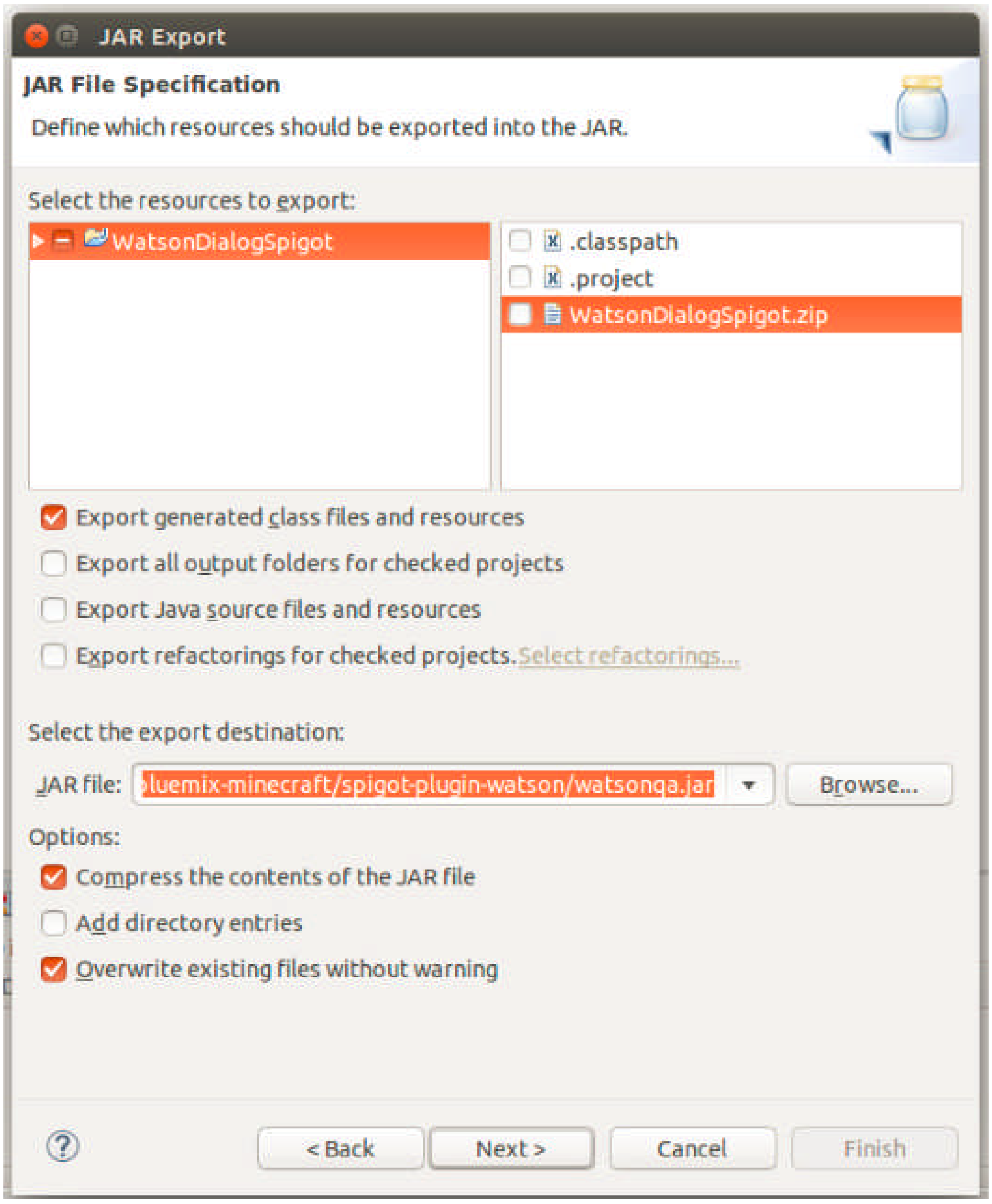

Package Explorer

```
▼ WatsonDialogSpigot
  ▼ src
    ▼ com.ibm.minecraft.spigotplugin
      ▶ WatsonDialog.java
    ▶ input
      plugin.yml
  ▶ JRE System Library [JavaSE-1.7]
  ▼ Referenced Libraries
    ▶ commons-codec-1.10.jar
    ▶ jackson-core-asl-1.9.13.jar
    ▶ jackson-mapper-asl-1.9.13.jar
    ▶ java-sdk-2.7.0-jar-with-dependencies.jar
    ▶ craftbukkit-1.8.8.jar - /home/scheema/reflibs
  ▼ lib
      commons-codec-1.10.jar
      jackson-core-asl-1.9.13.jar
      jackson-mapper-asl-1.9.13.jar
      java-sdk-2.7.0-jar-with-dependencies.jar
  ▼ META-INF
      MANIFEST.MF
```

Review the source code, file structure and files plugin.xml, Dialog-Test.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)*



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

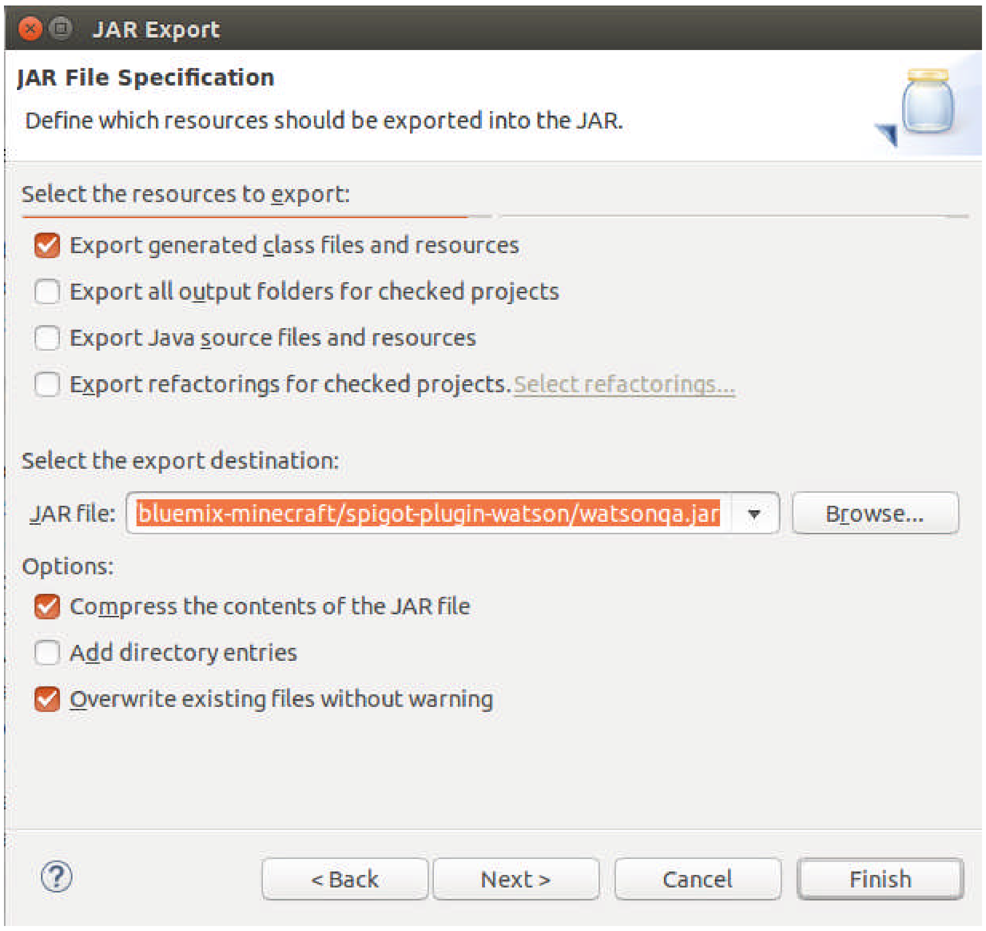
Options:

[✓] Compress the contents of the JAR file

[ ] Add directory entries

[✓] Overwrite existing files without warning

m)


JAR Export

JAR File Specification

Define which resources should be exported into the JAR.

Select the resources to export:

- [✓] 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: bluemix-minecraft/spigot-plugin-watson/watsonqa.jar

Options:

- [✓] Compress the contents of the JAR file
- [ ] Add directory entries
- [✓] Overwrite existing files without warning

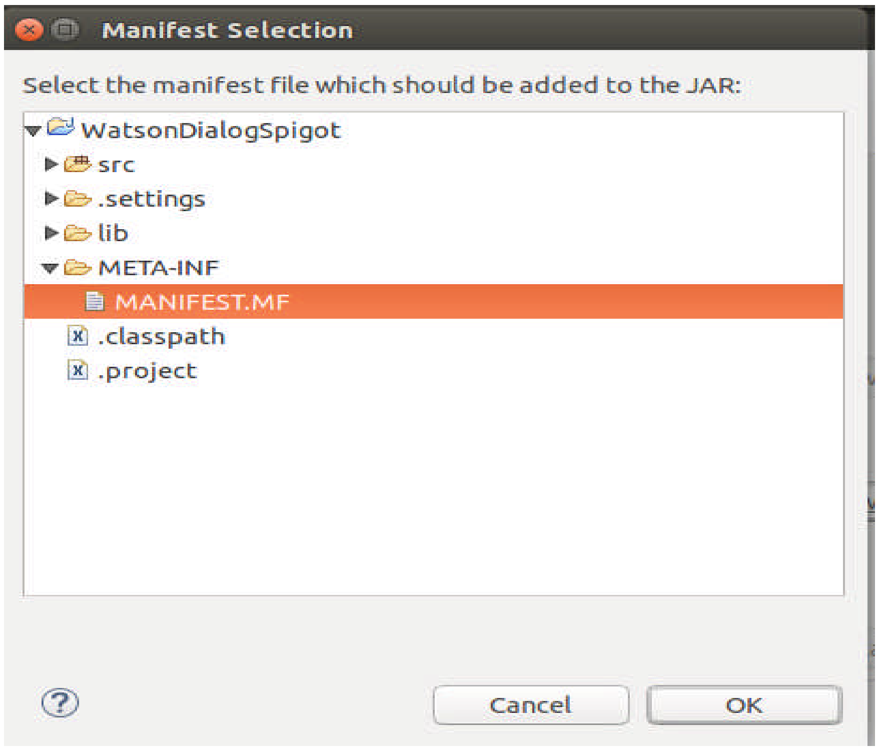
Check per the above figure.

Click Next

***Select the Manifest File** that is under the project at path*

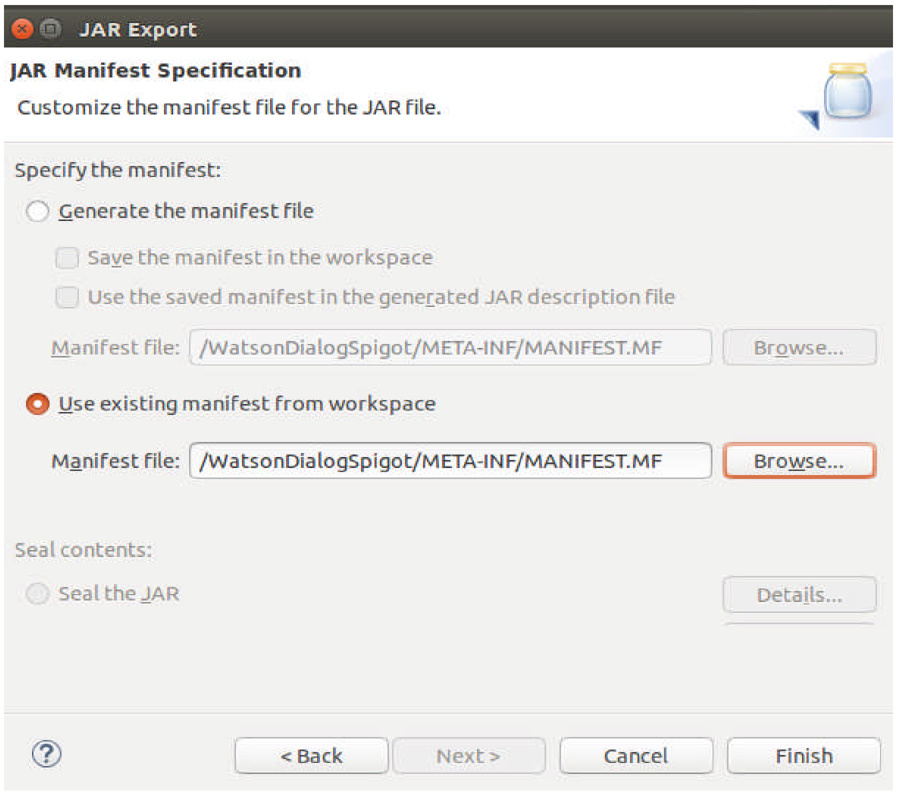
WatsonDialogSpigot -> META-INF -> MANIFEST>MF

The following figures can be used as the reference.


Manifest Selection

Select the manifest file which should be added to the JAR:

- WatsonDialogSpigot
  - src
  - .settings
  - lib
  - META-INF
    - MANIFEST.MF
  - .classpath
  - .project


JAR Export

JAR Manifest Specification

Customize the manifest file for the JAR file.

Specify the manifest:

( ) Generate the manifest file

[ ] Save the manifest in the workspace

[ ] Use the saved manifest in the generated JAR description file

Manifest file: /WatsonDialogSpigot/META-INF/MANIFEST.MF

[✓ Use existing manifest from workspace]

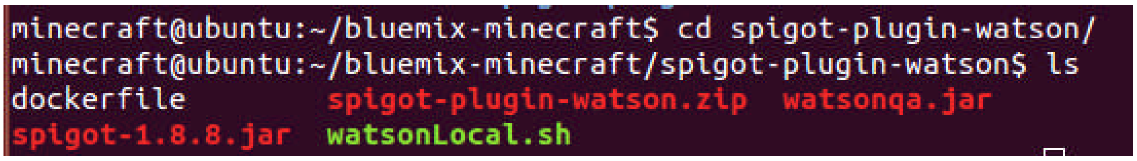
Manifest file: /WatsonDialogSpigot/META-INF/MANIFEST.MF

Seal contents:

( ) Seal the JAR

Click Finish

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



```
minecraft@ubuntu:~/bluemix-minecraft$ cd spigot-plugin-watson/
minecraft@ubuntu:~/bluemix-minecraft/spigot-plugin-watson$ ls
dockerfile        spigot-plugin-watson.zip  watsonqa.jar
spigot-1.8.8.jar  watsonLocal.sh
```

<u>Time to Test the Watson QA service</u>

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 the terminal, you will see a directory ~/bluemix-minecraft/watsonspigotbluemix:

```
mkdir $HOME/watsonspigotbluemix
```

Overwrite watsonqa.jar in ~/ bluemix-minecraft/watsonspigot from the folder ~/bluemix-minecraft/watsonspigot:

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

Note: We are basically replacing downloaded watsonqa.jar with the one created in the lab.

The following steps are needed to run to deploy in Bluemix

2. 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>
```

3. Build a new Docker image:

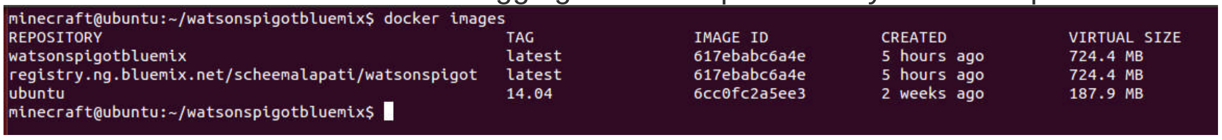
```
docker build -t watsonspigotbluemix .
```

4. Tag the created image:

```
docker tag watsonspigotbluemix registry.ng.bluemix.net/<namespace>/watsonspigot
```

Note: replace namespace with your namespace name

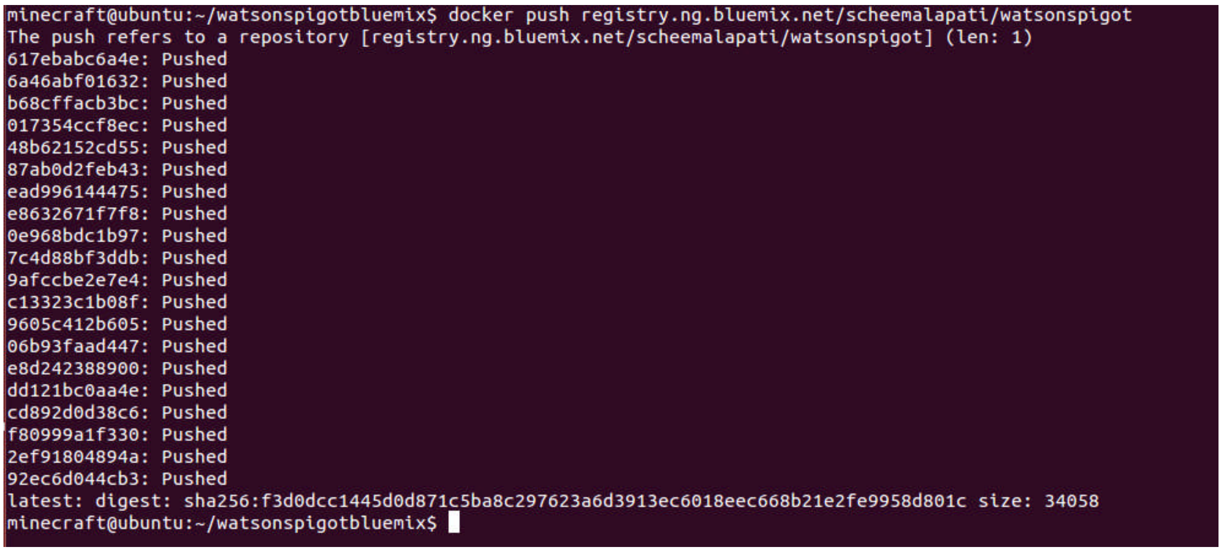
The screen will look like this after tagging. Please replace with your namespace



```
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$
```

5. Push the Docker image to Bluemix:

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

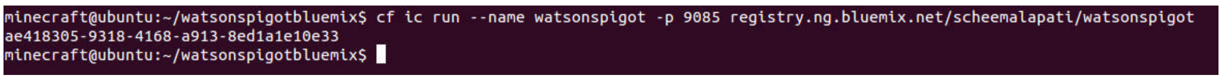


```
minecraft@ubuntu:~/watsonspigotbluemix$ docker push registry.ng.bluemix.net/scheemalapati/watsonspigot
The push refers to a repository [registry.ng.bluemix.net/scheemalapati/watsonspigot] (len: 1)
617ebabc6a4e: Pushed
6a46abf01632: Pushed
b68cffacb3bc: Pushed
017354ccf8ec: Pushed
48b62152cd55: Pushed
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$
```

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

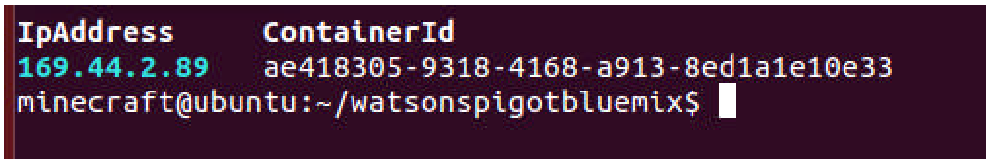
```
cf ic run --name=watsonspigot  -p 9085 registry.ng.bluemix.net/<
replace your name space > /watsonspigot
```

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$
```

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

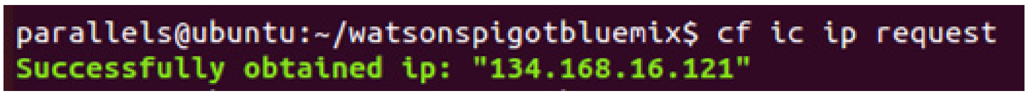


```
IpAddress      ContainerId
169.44.2.89    ae418305-9318-4168-a913-8ed1a1e10e33
minecraft@ubuntu:~/watsonspigotbluemix$
```

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

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

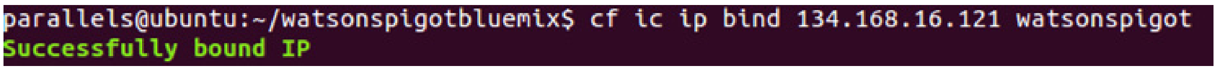
```
cf ic ip request
```



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

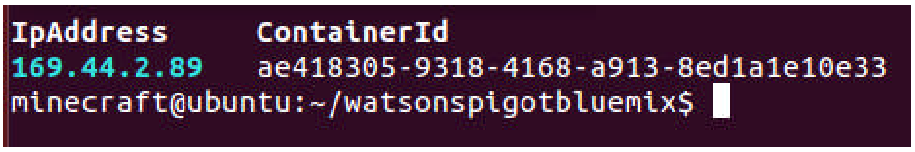
9. Bind the returned IP address to the Bluemix Docker container you ran in step 10: (note – run only if Step 10 not satisfied)

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



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

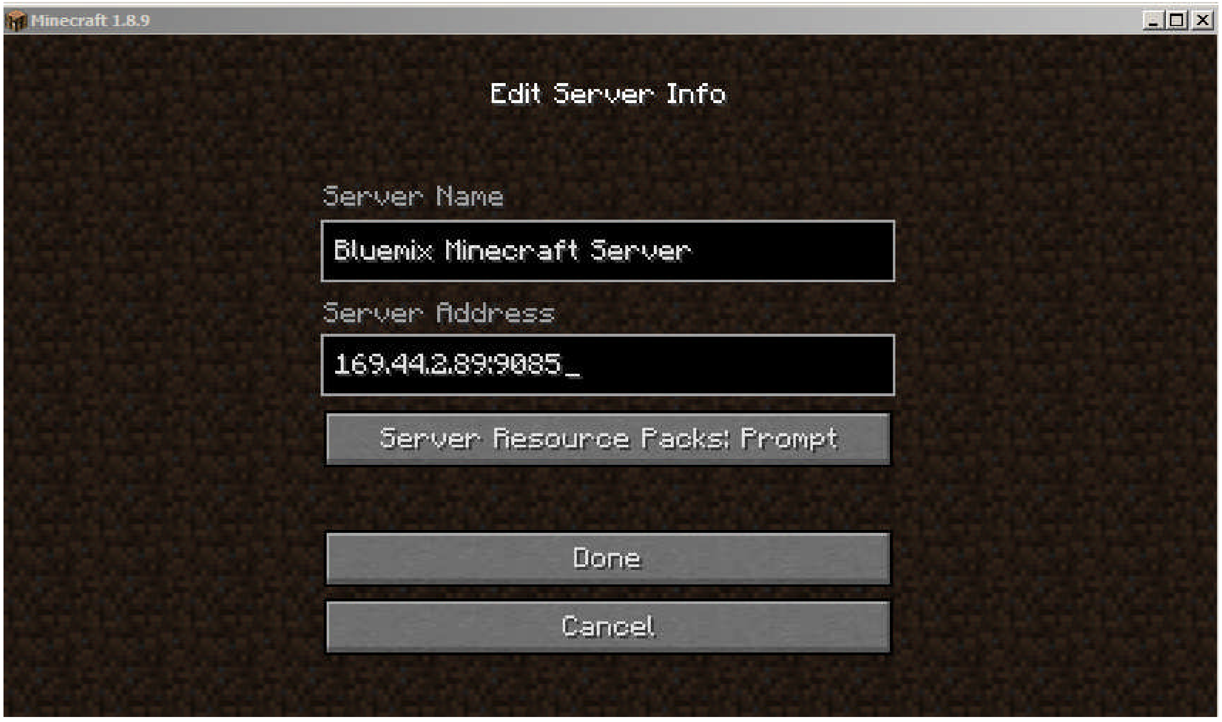
10. Verify that the IP address is bound to the Bluemix Docker container:



```
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

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



Minecraft 1.8.9

Edit Server Info

Server Name

Bluemix Minecraft Server

Server Address

169.44.2.89:9085_

12. 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