A Comparative Study of Deployment Tools from IOT Perspective

Raghav Maheshwari1, K. Rajalakshmi
Jaypee Institute of Information Technology
raghavddn16@gmai.com, k.rajalakshmi@jiit.ac.in

Abstract: Now days Internet of Things has become one of the most trending technology in the field of Computer Science. Many businesses are accepting the various practices of Internet of Things to bring reliability, stability and security in their development and production phases of products or services. Deploying their application to a large scale is the key challenge faced by many users. Choosing the right tool according to the needs is a decision one should make taking various parameters into consideration. This paper studies various deployment tools, based on various researches and practices, on platforms like Raspberry pi board, Linux devices and so on and tried to give a uniformed comparative study among them on various useful and important parameters.

Keywords: Dev-ops, Internet of Things, IOT, Deployment Tools, Deployment, Development, Production, Docker, Mender, AWS Greengrass, Kaa, Android Things, Raspberry Pi, Comparison

I Introduction

The Internet of Things (IOT) is considered to be a boom for Computer Science and Technology. Heterogeneous devices, like mobile phones, laptops, cameras, sensors, embedded devices and others, become part of a mesh network, thus enabling them to interact and communicate among them and use the shared resources. [11] Inspired from this, many started practices like continuous development, continuous testing, continuous integration and continuous deployment, commonly known as Dev-Ops. [3]

The practices involved users to maintain a stream of incessant flow of data, from the development phase till the deployment phase. Though there are some individual problems and errors in each of the phases, but it is the Phase of Continuous Deployment which involves a deep research to counter the problem. Continuous Deployment Cycle incorporates automated flow of data from one device to another, through some tools, as shown in Figure 1.

Various companies have business which requires them to choose some tools for the deployment of their application to remote devices. There is not only one tool, and all the tools don't possess the same features. To calculate their needs and then choose a deployment tool suitable for them, much man working hours are required. The idea of lessening such problems and having a comparative study of some such major tools, paved way to the research and practices, explained in later sections.

Deployment Tools, namely Docker, Mender, AWS Greengrass, Kaa, and Android Things, were chosen and were studied to present a comparison among them, to increase the users' convenience.

II Objectives

To present a comparison among the deployment tools, it is a necessary step to properly install them on some device and generate some application which could be deployed to other remote devices. Thus the main objectives of the paper are:

A. Installation of Tools - First and foremost step of the research is the proper installation of the deployment tools. These tools are Docker, Mender, AWS Greengrass, Kaa and Android Things. The proper installation is essential as it forms the basis of the research. The installation of the tools is explained in later sections respectively.
B. Application Development - For capturing the real-time data, this paper focused on developing an android application which uses the camera functionality of the mobile device to capture the image and send it to the system. The application was developed keeping in mind the deployment process, thus all the tools used the same.
C. Identification of the Component - The components or measures of the comparative study among different tools of deployment could be divided broadly into two categories: Functional and Non Functional. Functional issues could include Security, Privacy, Internet Layer, Languages used and others. Non Functional measures include Scalability, Elasticity, and Raspberry Pi support and so on.
D. Comparative Study - The final step is to list down a detailed comparative study among different tools of deployment, on different functional and non-functional measures working on similar kind of data. This gave us a clear view of pros and cons of the tools in comparison with one another.

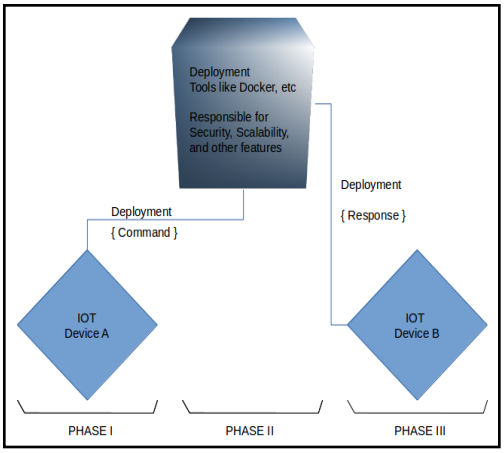

Figure 1. Phases in the Deployment Cycle

III Layers of Internet of things

It is very important to understand the seed of deployment tools. These tools work for the enhanced experience of Internet if Things. IOT is not just devices, but consists of different layers. [1] As shown in Figure 2, they are:

A. *End Systems* - At the very bottom layer comes the End Systems. These are the equipment or the hardware used in any IOT based project. It could be sensors like temperature, heat, humidity or could be Cameras, Mobile Devices, Capturing Equipment, RFID tags, and other wired or wireless devices, or a combination of both.

   Its key feature is to collect information from its surroundings and pass on the information. For this paper mobile devices and cameras have been used to capture the information. Being responsive in nature, they are able to get live data as per the need.
B. *Connections* - The Connections used by the End Systems to pass on the information further could be a normal Router WIFI, or a subnet of the net. It could use LTE/4G, WAN or maybe LANs too, depending on the need. Its key function is to provide a connection among the End Systems and the Network Layer.
C. *Network Layer* - Next comes the Network Layer. Network Layer contains the IPv4/IPv6, and so on, which acts as a unique identity for different components of the system, to ease the process of the communication. The

sending and receiving of the data is done through these addresses, like the MAC addresses or 6LoWPAN and others. It is responsible for providing addresses to different components for making the communication possible.

This paper focuses more on IPv4 for addressing in the project. In other words, it could be said that doorway helps in to and fro movement of the data. It provides network connectivity to the data, which is essential for any IOT system.

D. *TCP / UDP* - The connections are established either by *TCP or UDP* protocol. TCP or Transmission Control Protocol is connection oriented – once a connection is established, data can be sent bidirectional. UDP or User Datagram Protocol is a simpler, connectionless Internet protocol.

E. *Middle Ware* - For deployments and over the air transmission, comes the Middle Ware. Tools like Mender, Docker, AWS Greengrass, Jenkins, Apache, Kaa and so on are built solely for deployments of data from system to cloud or from one system to another.

Its main function is to securely deploy the data to its destination. The destination may be cloud of any other system. The various deployment tools falls under this layer of the IOT Architecture.

F. *Application Layer* - The final layer id the Application Layer, where the data rests on reaching. It could be Cloud Servers, Web Hosts, and Android Platform and so on. Data Analytics or Visualization can be under on this layer. Applications are controlled by users and are delivery point of a particular service.

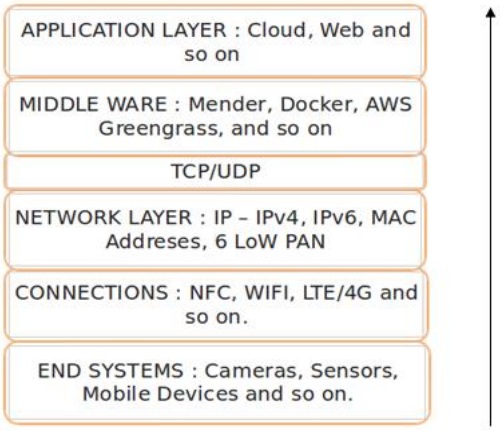

Figure 2. IOT Architecture

IV Getting started with Docker

Docker is a platform to develop, deploy and run applications with containerization technology. An image is launched by running a container. [6] A container is an executable package that provides a ready to use environment on any machine. [2] The following shows the steps to install Docker successfully for efficient working.

A. Installing Docker

1) Install Docker from the terminal using the convenience script, as a standard command, as in Figure 3.

```
curl -fsSL get.docker.com -o get-docker.sh
sudo sh get-docker.sh
```

Figure 3. Standard Command for the Convenience script [2]

2) To use as a non- root user [2]
sudo usermod -aG docker your-user (1)

B. Building Images of Application

Here take the application ready to be deployed. After creating the script for the application in the conventional way, one needs to build it.

1) Build the image by getting into the directory of the image : [2]
$ docker build -t <image name> . (2)

2) To run the image : [2]
docker run -d <image name> (3)

3) To save the image in .tar format : [2]
$ docker save -o <filename>.tar <imagename> (4)

C. Pushing and Pulling the Image

1) Create a Local registry, the standard command is:

```
$ docker run -d -e
http://192.168.1.33:5001
--restart=always --name registry
```

Figure 4. Local Registry script standard command [2]

2) Tag the image: [2]
$ docker tag <imagename> 192.168.5.45:5010/newname (5)

3) Pushing the image into the registry : [2]
$ docker push 192.168.5.45:5010/newname (6)

4) Pulling the image on from the registry : [2]
$ docker pull 192.168.5.45:5010/newname (7)

To felicitate remote pulling of the image push the image to the docker hub instead of local registry.

Thus the Docker is ready and up for working

V Getting started with Mender

Mender is an open source, over the air deployment tool for Linux devices. As a pre requisite Docker Engine is required for Mender to work. [4] For the correct installation of Mende, the following steps can be followed.

A. Installing Mender

1) Mender integration environment is initially downloaded :

```
git clone –b 1.6.0
https://github.com/mendersoftware/inte
gration.git integration-1.6.0
cd integration-1.6.0
```

Figure 5. Mender initial script standard command [4]

2) Create the initial user and allow the secure connection afterwards : [4]
sudo ./demo exec mender-useradm /usr/bin/useradm create-user --username=newname@emailaddress.com --password=newpassword (8)

B. Deploying to Devices

Open the Mender UI by navigating to https://localhost/ in the same browser as you accepted the certificate. One can also download the demo images for testing purposes.

There are two types of images: [4]

-Disk images (*.sdimg): they are used to establish the device memory for devices without Mender running already.
-Mender Artifacts (*.mender): they are used to push images to the Mender server so that new root file systems could be deployed to devices in which Mender is already running.
The images could be deployed to devices like Raspberry pi and Beagle Bone, and its progress can be tracked on the UI itself.

C. Maintaining the Environment

1) Stopping the Mender Services [4]
./stop (9)
2) Starting the Mender Services
./start (10)
3) Wrapper script to bring up the environment
./up (11)

Thus the Mender is ready and up for running.

VI Getting started with AWS Greengrass

AWS Greengrass comes under the umbrella of AWS IOT Console, to spread the capabilities to local devices, making a secure network and medium for information exchange. [7]

To run Greengrass on a Raspberry pi board, firstly the P2 SSH option should be enabled. After creating a group and a user, one need to make sure that the Linux Kernel is up to date and all the dependencies are met. To check them, the standard commands are:

```
cd /home/pi/Downloads
git clone https://github.com/aws-samples/aws-greengrass-samples.git
cd aws-greengrass-samples
cd greengrass-dependency-checker-GGCv1.6.0
sudo modprobe configs
sudo ./check_ggc_dependencies | more
```

Figure 6. Greengrass Dependency script standard command [7]

A. Installing Greengrass Core Daemon

1) After signing into the AWS Management Console, browse to Greengrass and select Create an Greengrass Group.
2) Follow the steps, by giving the Group name, Core name, Easy Group Creation and save the Certificates on the system. Also download the Software Configuration for Raspbian on Linux.
3) 3) Now send the zipped downloaded files to the raspberry pi board (preferred). To transfer the compressed files from your computer to a Raspberry Pi AWS Greengrass core device, open a terminal window on your computer and run the following standard commands:

```
cd path-to-downloaded-files
sudo scp greengrass-OS-architecture-1.6.0.tar.gz pi@IP-address:/home/pi
sudo scp GUID-setup.tar.gz pi@IP-address:/home/pi
```

Figure 7. System Script standard command [7]

On the Raspberry pi board:

```
cd path-to-compressed-files
sudo tar -xzvf greengrass-OS-architecture-
1.6.0.tar.gz -C /
sudo tar -xzvf GUID-setup.tar.gz -C
/greengrass
```

Figure 8. Raspberry Pi Board System Script standard command [7]

4) Install the root- CA certificate and run the daemon through the standard command : [7]
cd /greengrass/ggc/core/
sudo ./greengrassd start (12)

B. Interacting with Devices

1) After choosing Add Device on the Group Configuration page, give the name, like Publisher or Subscriber, and also download the security certificates. The devices are added to the Greengrass Group.
2) On the group configuration page, select the source after Add Subscription. Choose Select, Devices, and choose Publisher. Similarly, for a target, choose Subscriber. Finally give a topic name for the deployment.
3) Choose Deploy option to deploy the updated sample group configuration. The working could be seen by opening two terminal windows as Publisher and Subscriber respectively and giving the respective standard commands, as shown in Figure 9 and Figure 10.

```
python basicDiscovery.py --endpoint
AWS_IOT_ENDPOINT --rootCA root-ca-
cert.pem --cert publisher.cert.pem --key
publisher.private.key --thingName
HelloWorld_Publisher --topic
'hello/world/pubsub' --mode publish --
message 'Hello, World! Sent from
HelloWorld_Publisher'
```

Figure 9. Publisher script standard command [7]

```
python basicDiscovery.py --endpoint
AWS_IOT_ENDPOINT --rootCA root-ca-
cert.pem --cert subscriber.cert.pem --key
subscriber.private.key --thingName
HelloWorld_Subscriber --topic
'hello/world/pubsub' --mode subscribe
```

Figure 10. Subscriber script standard command [7]

Lambda functions could also be created for further functionality and triggered for more accurate communications and data processing.

Thus the AWS Greengrass is set up and ready for working.

VII Getting started with KAA

Kaa is an open source IOT platform. Its architecture consists of a Kaa Server, Kaa extensions and Endpoint SDK, along with Apache zookeeper. To set up Kaa, this paper chose virtual sandbox, which could run on any system. [10] Kaa installation is done on a virtual machine following the steps given below.

A. Installing Kaa Sandbox

1) Download Oracle Virtual Box (preferred) on the system, and then download the Sandbox.ova image.
2) Once installed, adjust the memory and processors to be distributed to the virtual box.
3) Once all done, it opens an URL, directing us to the Kaa UI page.
4) Go to Sandbox Management Page and specify the real IP of the system.

B. Launching Kaa Application

1) After installation, Kaa provides sample applications for test running. Using the sandbox, one can even download the source codes of several applications or make a new one (as made earlier). To work with user defined instances, download an SDK Library and deploy it to the required endpoint.
2) On the Administrative UI, login and go to the Add Application and enter the required information. Create file data-scheme.json and configuration-scheme.json to give the required information.
3) Following this, upload the file, choose the tenant developer and finally click on Add Scheme from the options. After giving the appropriate description, add the scheme.
4) To generate the SDK, on the UI, go to Generate SDK option, and create the profile. Fill out the required details and choose the corresponding option to generate one.
5) For the Client Side Application, write the code either in C/C++/Java after installing the respective dependencies. For the Java application :

   Save the application in the demo_app directory and then build the java application: [10]

   java –cp *.jar *.java (13)

   Launch the application to get started it working: [10]

   java java –cp '.:./*' Filename (14)

Other configuration changes could also be done, like end points and so on as per our need.

Thus Kaa is ready and up for working.

VIII Getting started with Android Things

Android Things is an IOT platform based on Android, from Google. As the name suggests, application developed in android can be deployed and Android Studio can be used for debugging mechanisms. [12]

A. Things SDK

With additional APIs from Support Library, Android Things encompasses the utility and functionality of the core Android frameworks. This let users to make possible the communication and message passing among different types of hardware devices. Some features of such apps are:

1. It has increased access to hardware resources and drivers.
2. The starting up and memory requirements are not optimized by system apps.
3. Automatic launching of apps on startup of the device.

B. Console

For user convenience and higher working experience, Android Things presents an UI, known an Console. It enables users to install or update, push or pull images on connected devices. It increases the user functionality the ease at which one can deploy applications.

C. Launching

To get started, create the application to be deployed using the new project wizard:

1. Choose Android Things as the form factor.
2. Select Android Oreo (8.1) or higher. This enables user to incorporates other APIs and dependencies.
3. Give a name to the empty new activity, like New Activity or Home Activity.
4. Android Things add the dependency to the Things Support Library to the application build gradle file: [12]

   dependencies {

```
    …
    compileOnly
        'com.google.android.things:androidthings:+'
}
```
(15)

5. Connect to Raspberry pi or other hardware through TTL Cable.

Android Things is set up and ready for working.

IX Application

An Android Application is written that uses the Camera property of the mobile devices to capture the image of the surrounding (preferably a person). The image is then transmitted by a local node js server to the localhost, thus showing the image on the browser. Figure 11 and Figure 12 shows the snippet of the Android Application and Node js server respectively.

```
@app.route('/',methods=['GET','POST'])

def index(): #storing mechanism
        request_data = request.get_json()
        nm = request_data['name']
        #print len(nm)
        image_64_decode=base64.b64decode(nm)

        filepath = os.path.join(path_to_folder,"a"+".PNG")

        if not os.path.exists(path_to_folder):
                os.makedirs(path_to_folder)
        file1 = open(filepath, "wb")

        file1.write(image_64_decode)

        file1.close()
        return "File created"


if(__name__=="__main__"): #local host address
        app.run(debug=True,host='192.168.43.26',port=5021)
```

Figure 11. Snippet for the Android Application

```
    <configuration>
      <option name="BUILD_FOLDER_PATH" value="$MODULE_DIR$/build" />
      <option name="BUILDABLE" value="false" />
    </configuration>
  </facet>
</component>
<component name="NewModuleRootManager" LANGUAGE_LEVEL="JDK_1_7" inherit-compiler-outp
  <exclude-output />
  <content url="file://$MODULE_DIR$">
    <excludeFolder url="file://$MODULE_DIR$/.gradle" />
  </content>
  <orderEntry type="jdk" jdkName="1.8" jdkType="JavaSDK" />
  <orderEntry type="sourceFolder" forTests="false" />
</component>
```

Figure 12. Snippet for the Node.js Server

X Comparative Study

Table 1 shows the comparative study among four deployment tools on various functional and non-functional parameters, based on researches and practical applications.

A. Comparison Table

The comparison table for the five tools could be summarized as follows:

TABLE 1 COMPARATIVE STUDY

	DOCKER	MENDER	AWS GREEN GRASS	KAA	ANDROID THINGS
BASE	Lightweight as above Host OS	Need Docker Engine as a prerequisite	Use of Lambda Functions and Triggers	Use of Sandbox or Virtual Machine	Android platform
CONCEPT	Containerization	Uses Docker Theory of Containerization	Uses Pub-Sub and OPC-UA Protocol	Kaa Clusters	Google's Developers Kit (Things SDK)
SECURITY	Good	Good	Good	Good	Good
SCALABILITY	High	High	High	High	High
RPI SUPPORT	Yes	Yes (better in later versions)	Yes	Yes	Yes (Just for Development)
DEVELOP-MENT LEVEL	Good	Good (needs errors resolving)	Good (A bit confusing)	Good	Good
PRODUCTI-ON LEVEL	Better	Good	Good	Good (need changes)	Good
PAID VERSION AVAILABLE	Yes	Yes	Yes	Yes	Yes
OTA UPDATES	Yes	Yes	Yes	Yes	Yes
DOCUMENTATION	Available	Available	Available	Available	Available
JAVA	Yes	Yes	Yes	Yes	Yes (Android)
PYTHON	Yes	Yes	Yes	No	No
C/C++	Yes	Yes	No	Yes	No
INTERNET	Could run below or above the Internet	Internet connection is required for the setup and updates	Devices should be sharing the same network.	Needs an Internet connection to connect to the cloud, gateways otherwise.	Wired connection for initial testing, internet otherwise.
CLOUD SUPPORT	Docker Hub	Mender Server	AWS IOT Console	Kaa Cloud	Google Store
REQUIREMENTS	Docker Engine	Docker Engine and Internet	AWS Core and Device SDKs	Kaa Sandbox	Google SDK and Android Studio

B. Analysis

Docker gives us the easiest way to transfer application to remote systems security. It uses the concept of containerization, which is the application is packed with the dependencies, just like a ship container. One can make multiple images of the same container and distribute them to various locations. One can create a local registry, or make an account on the Docker Hub for pushing and pulling images. The building process is simple and easy to handle, but some concerns occur over the security [8] automation of the deployments.

Mender incorporates Docker theory of containerization and Docker Engine, the latter being a prerequisite for working with Mender. It provides sample images and user friendly UI for better working experience. Deploying to Raspberry pi boards have errors in earlier versions, with its control measures released with the subsequent versions.

AWS Greengrass, generally comes under the shade of AWS IOT Platform, gives a wide range of functionalities, but tends to take a confusing road for new users. With the concept of device shadows, it helps interacting within a local network. Lambda functions, user modified programming modules, increase functionality. It is more onto the side of messages communication via OPC-UA, with Machine Learning models to work with.

Kaa provides a platform for device communication with the Cloud via protocols like MQTT. It also provides gateway architecture in cases of Internet failure. It helps doing data analytics and data visualization over the data collected. Though it gives a good UI interaction, it sometimes get confusing working with it. It provides Link Encryption type security with real time computations.

Android Things, by Google, brings on Android Platform as its basis. One should have a good knowledge of Android and Android Studio, before starting with it. It supports Raspberry pi board, but only for development level, as of now. Debugging of errors becomes simpler with Android Things. It also provides its own kit, for initial understanding of its working.

XI Conclusion

The Dev-Ops and IOT are now days becoming a very closely related fields. Deployment tools are quite responsible for the smooth relationship between them. Based on researches and practical applications, this paper tries to find out various pros and cons of different tools, and bringing them together in a comparative study.

Docker shows very promising results for the deployment of various applications. The brand name of Amazon Web Services and Google behind tools Greengrass and Android Things respectively, gives an edged value to them. Mender and Kaa also showed good results and scope of improvement. Hence, one needs to select the tool depending on the needs and necessity of the products or services. This paper also paves way for further researches on similar topics.

References

1. Weyrich, M., & Ebert, C. (2016). Reference architectures for the internet of things. IEEE Software, 33(1), 112-116.Di Liu, Lubin Zhao, The Research and Implementation of Cloud Computing Platform based on Docker, International Center for Wavelet Analysis and Its Applications, School of Information and Software Engineering, ACM Journal 2015.
2. Documentation on Docker - https://www.docker.com/
3. Callanan, M., & Spillane, A. (2016). DevOps: making it easy to do the right thing. IEEE Software, 33(3), 53-59.Huang Xuyong, "Basic Research of Wireless Sensor Networks and Applications in Power System", Huazhong University of Science & Technology, Wuhan, 2008.
4. Documentation on Mender - https://mender.io/
5. Kersten, M. (2018). A Cambrian Explosion of DevOps Tools. IEEE Software, 35(2), 14-17.
6. Pahl, C. (2015). Containerization and the paas cloud. IEEE Cloud Computing, 2(3), 24-31.
7. Documentation on AWS GreenGrass - https://aws.amazon.com/greengrass/
8. Combe, T., Martin, A., & Di Pietro, R. (2016). To Docker or not to Docker: A security perspective. IEEE Cloud Computing, 3(5), 54-62.
9. Xavier M G, Neves M V, Rossi F D,et al. Performance evaluation of container-based virtualization for high performance computing environments[C]//Parallel, Distributed and Network-Based Processing (PDP), 2013 21st Euromicro International Conference on. IEEE, 2013: 233-240
10. Documentation on Kaa - https://www.kaaproject.org/
11. S. Haller, S. Karnouskos, and C. Schroth, "The Internet of Things in an Enterprise Context," in Future Internet – FIS 2008 Lecture Notes in Computer Science Vol. 5468, 2009, pp 14-28.
12. Documentation on Android Things - https://developer.android.com/things/get-started/
13. A. C. Sarma, and J. Girão, "Identities in the Future Internet of Things," in Wireless Personal Communications 49.3 , 2009, pp. 353-363.