

# Azure DevOps for IoT and Intercommunications

Ravindersingh Khalsa 201901179  
Jitanshu Shaw 201901292  
Mushir Shaikh 201901475

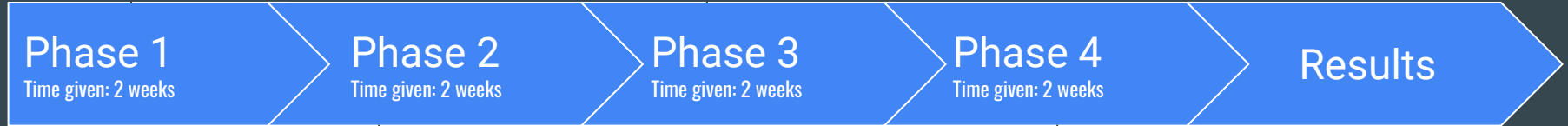


Guided By:- Tapas Kumar Maiti

# Project Timeline

Learning about  
DevOps

Talker-Listener Model  
Installation



Azure IoT

Talker-Listener Model  
Deployment



Phase 1

Phase 2

Phase 3

Phase 4

End

# DevOps

## What is DevOps?

By the name itself it is very clear, it is a collaboration of Development as well as Operations. Basically it is a combination of tools which helps for the automation of the whole infrastructure. It is an implementation of Agile methodology on the development side as well as operations side. Using DevOps, Development and testing in a production similar environment deploy builds frequently validate operation quality continuously.



Phase 1

Phase 2

Phase 3

Phase 4

End

# DevOps

## What is the need for DevOps?

The general market trend can be self explanatory for this. Instead of releasing big chunks of the program features, nowadays companies are trying to see if small features can be delivered to their customers through a series of the releases. This gives the company several benefits such as quicker feedbacks from the consumers, better software quality, etc. which results in high customer satisfaction. Accordingly the companies are required to

- Increase deployment frequency
- In addition, failure rates of new releases can be reduced.
- Quicker mean time to recover from the event of new release crashing.



Phase 1

Phase 2

Phase 3

Phase 4

End

# DevOps

## Phases of DevOps

According to DevOps practices, mentioned in few upcoming slides, is the general logic of the flow where everything tries to get automated for seamless delivery. However, it may vary from organization to organization as per the requirement.

- Plan: Here, the clients and software development team discuss project goals and work towards creating a plan.
- Code: Programmers here first design and then code tech application and use tools like git to store application code.
- Build : There are several tools like Maven and Gradle, take code from different repositories and combine them to build the complete application.



Phase 1

Phase 2

Phase 3

Phase 4

End

# DevOps

## Phases of DevOps

- **Test:** The application is tested using automated testing tools like selenium to ensure software quality. There are some docker containers which provide testing environment to test the build features.
- **Integrate:** When testing is complete, new features are integrated automatically to the already existing code-base.
- **Deploy:** Application is packaged after release and de-ployed from development server to production server



Phase 1

Phase 2

Phase 3

Phase 4

End

# DevOps

## Phases of DevOps

- Operate: Once software is deployed, operations team performs activities such as configuring servers and provisioning them with the required resources.
- Monitor: Monitoring allows IT organization to identify specific releases and understand the impact on end-users.



Phase 1

Phase 2

Phase 3

Phase 4

End

# DevOps

## Usefulness of DevOps

- Companies which follow DevOps, release more products and features within a short amount of time.
- Time taken to create and deliver software is reduced.
- Complexity of maintaining a software is reduced.
- Improved Collaboration between Developers and Operations team.
- Continuous integration and delivery ensure faster time to market.
- DevOps enables IT organizations to meet two critical business goals simultaneously, respond to urgent business needs, and second providing stable, reliable and secure IT services.





Phase 1

Phase 2

Phase 3

Phase 4

End

# Azure IoT Hub

IoT: Internet of Things, think of it as a large ecosystem where devices gather data using sensors, interact using actuators, connect with peer devices and the internet. But, IoT is more than devices, it also comprises of cloud services like processing sensor data, etc. and also edge processing, in which we don't need connectivity and will have to process data locally using AI models trained in the cloud.

So, an IoT application and the connected devices use the managed Azure IoT Hub service, which is hosted in the cloud and serves as a central messaging hub. An IoT hub can be connected to almost any device. Azure IoT Hub provides bidirectional communication between millions of devices, it is multi language, open source sdk. It also provides monitoring for configuration management, it makes easier to scale the system up and down. And lastly it provides end to end security with services like IP whitelisting, blacklisting, shared access policies, and also open to firmware and software updates.



Phase 1

Phase 2

Phase 3

Phase 4

End

# Azure IoT Hub

## Authentication

Azure IoT hub or any IoT hub has an identity registry that stores information about the devices and modules permitted to connect to it. Before a device or module can connect, there must be an entry for that device or module in the IoT hub's identity registry. A device or module authenticates with the IoT hub based on credentials stored in the identity registry.

Azure IoT Hub supports two methods of authentication between the device and the IoT hub. You can use SAS token-based authentication or X.509 certificate authentication.

## Communication

After selecting the authentication method, the internet connection between the IoT device and IoT Hub is secured using the Transport Layer Security (TLS) standard. There are several versions of TLS which Azure IoT supports like TLS 1.0, 1.1, and 1.2.



Phase 1

Phase 2

Phase 3

Phase 4

End

## Talker-Listener Model Implementation

Talker Listener model is based on publish subscribe model where a group of node creates a topic and sends a message. The nodes which are subscribed to the topic received the message immediately. We are implementing the talker listener model by running the image in a docker container in Terminal using the foxy desktop directory of ROS

Phase 1

Phase 2

Phase 3

Phase 4

End

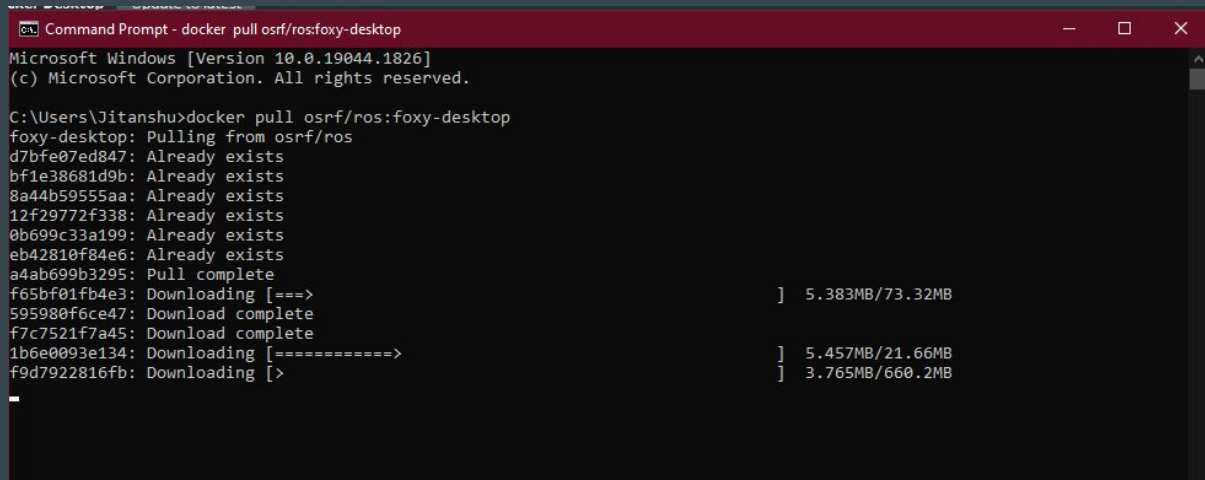
# Talker-Listener Model

## Step by Step Instructions

First step is to retrieve the ROS docker image that has the tag "foxy-desktop."

```
docker pull osrf/ros:foxy-desktop
```

The first step results in



```
Microsoft Windows [Version 10.0.19044.1826]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Jitanshu>docker pull osrf/ros:foxy-desktop
foxy-desktop: Pulling from osrf/ros
d7bfe07ed847: Already exists
bf1e38681d9b: Already exists
8a44b59555aa: Already exists
12f29772f338: Already exists
0b699c33a199: Already exists
eb42810f84e6: Already exists
a4ab699b3295: Pull complete
f65bf01fb4e3: Downloading [====>] 5.383MB/73.32MB
595980f6ce47: Download complete
f7c7521f7a45: Download complete
1b6e0093e134: Downloading [=====>] 5.457MB/21.66MB
f9d7922816fb: Downloading [>] 3.765MB/660.2MB
```

Phase 1

Phase 2

Phase 3

Phase 4

End

# Talker-Listener Model

## Step by Step Instructions

The next step is to run the image in a docker container using Terminal.

After that, ROS2 help command can provide the assistance.

```
root@f722d3318e44: /
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\Jitanshu> docker run -it osrf/ros:foxy-desktop
root@f722d3318e44:/#
```

```
root@f722d3318e44: /
PS C:\Users\Jitanshu> docker run -it osrf/ros:foxy-desktop
root@f722d3318e44:/# ros2 --help
usage: ros2 [-h] Call 'ros2 <command> -h' for more detailed usage. ...

ros2 is an extensible command-line tool for ROS 2.

optional arguments:
  -h, --help            show this help message and exit

Commands:
  action               Various action related sub-commands
  bag                  Various rosbag related sub-commands
  component            Various component related sub-commands
  daemon              Various daemon related sub-commands
  doctor              Check ROS setup and other potential issues
  interface           Show information about ROS interfaces
  launch              Run a launch file
  lifecycle            Various lifecycle related sub-commands
  multicast           Various multicast related sub-commands
  node                Various node related sub-commands
  param               Various param related sub-commands
  pkg                 Various package related sub-commands
  run                 Run a package specific executable
  security            Various security related sub-commands
  service            Various service related sub-commands
  topic              Various topic related sub-commands
  wtf                 Use 'wtf' as alias to 'doctor'

Call 'ros2 <command> -h' for more detailed usage.
root@f722d3318e44:/#
```

Phase 1

Phase 2

Phase 3

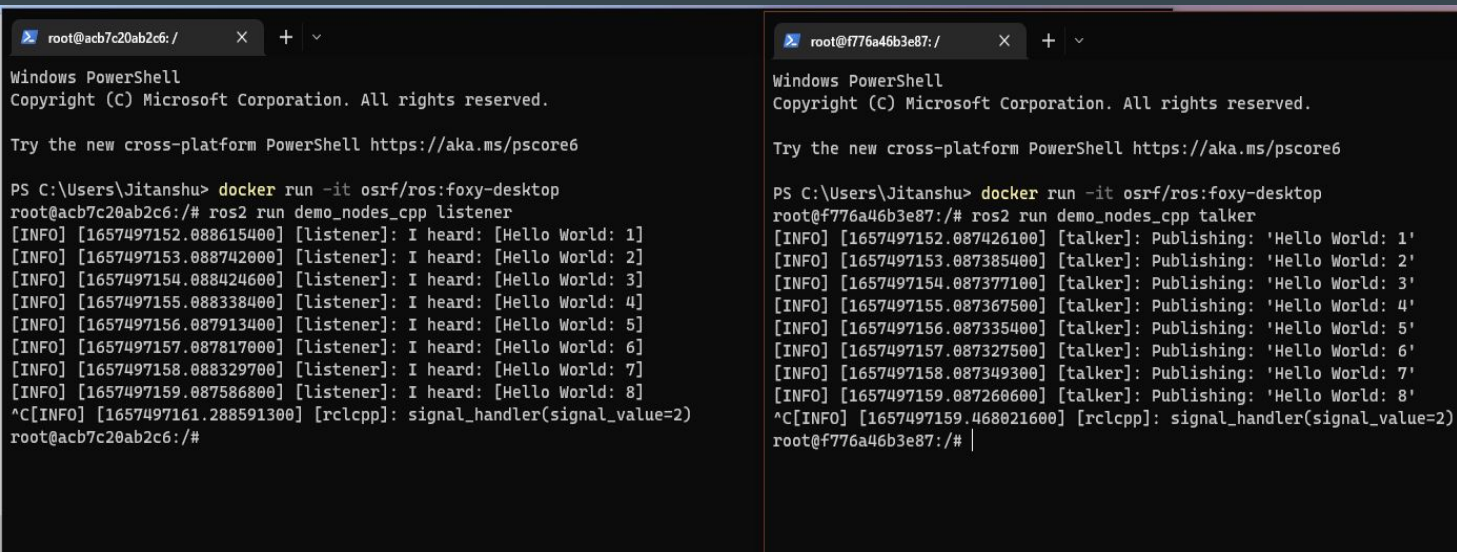
Phase 4

End

# Talker-Listener Model

## Step by Step Instructions

The final step is to run two separate docker containers, one for talker and one for listener. As we are doing a simulation here, we need to open two different terminals. Instead, if the same is done on Azure IoT Hub, we can do the same task on two different machines.



```
root@acb7c20ab2c6: /  
Windows PowerShell  
Copyright (C) Microsoft Corporation. All rights reserved.  
  
Try the new cross-platform PowerShell https://aka.ms/pscore6  
  
PS C:\Users\Jitanshu> docker run -it osrf/ros:foxy-desktop  
root@acb7c20ab2c6:/# ros2 run demo_nodes_cpp listener  
[INFO] [1657497152.088615400] [listener]: I heard: [Hello World: 1]  
[INFO] [1657497153.088742000] [listener]: I heard: [Hello World: 2]  
[INFO] [1657497154.088424600] [listener]: I heard: [Hello World: 3]  
[INFO] [1657497155.088338400] [listener]: I heard: [Hello World: 4]  
[INFO] [1657497156.087913400] [listener]: I heard: [Hello World: 5]  
[INFO] [1657497157.087817000] [listener]: I heard: [Hello World: 6]  
[INFO] [1657497158.088329700] [listener]: I heard: [Hello World: 7]  
[INFO] [1657497159.087586800] [listener]: I heard: [Hello World: 8]  
^C[INFO] [1657497161.288591300] [rclcpp]: signal_handler(signal_value=2)  
root@acb7c20ab2c6:/#
```

```
root@f776a46b3e87: /  
Windows PowerShell  
Copyright (C) Microsoft Corporation. All rights reserved.  
  
Try the new cross-platform PowerShell https://aka.ms/pscore6  
  
PS C:\Users\Jitanshu> docker run -it osrf/ros:foxy-desktop  
root@f776a46b3e87:/# ros2 run demo_nodes_cpp talker  
[INFO] [1657497152.087426100] [talker]: Publishing: 'Hello World: 1'  
[INFO] [1657497153.087385400] [talker]: Publishing: 'Hello World: 2'  
[INFO] [1657497154.087377100] [talker]: Publishing: 'Hello World: 3'  
[INFO] [1657497155.087367500] [talker]: Publishing: 'Hello World: 4'  
[INFO] [1657497156.087335400] [talker]: Publishing: 'Hello World: 5'  
[INFO] [1657497157.087327500] [talker]: Publishing: 'Hello World: 6'  
[INFO] [1657497158.087349300] [talker]: Publishing: 'Hello World: 7'  
[INFO] [1657497159.087260600] [talker]: Publishing: 'Hello World: 8'  
^C[INFO] [1657497159.468021600] [rclcpp]: signal_handler(signal_value=2)  
root@f776a46b3e87:/#
```

Phase 1

Phase 2

Phase 3

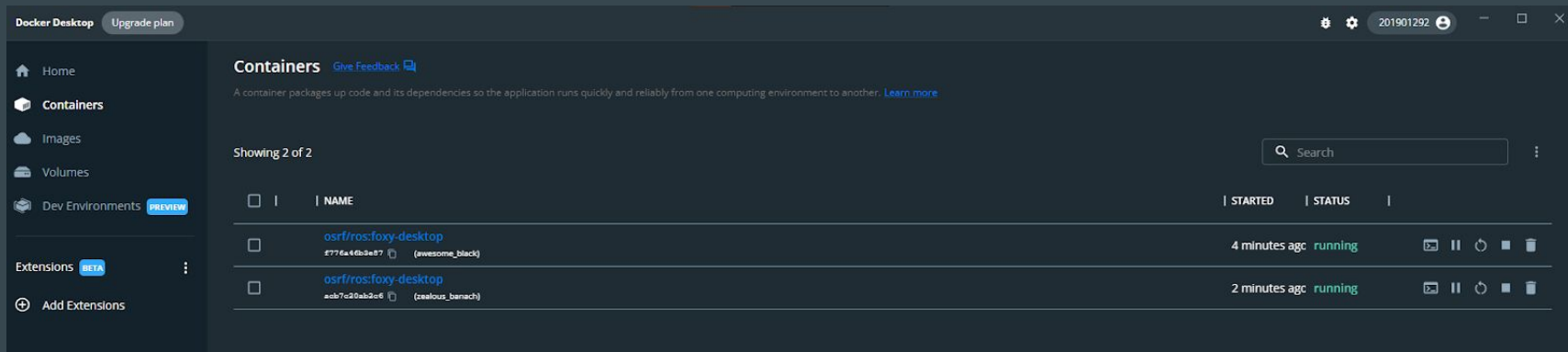
Phase 4

End

# Talker-Listener Model

## Step by Step Instructions

From the below image we can see that two different containers are running, one for listener and other for talker.



Phase 1

Phase 2

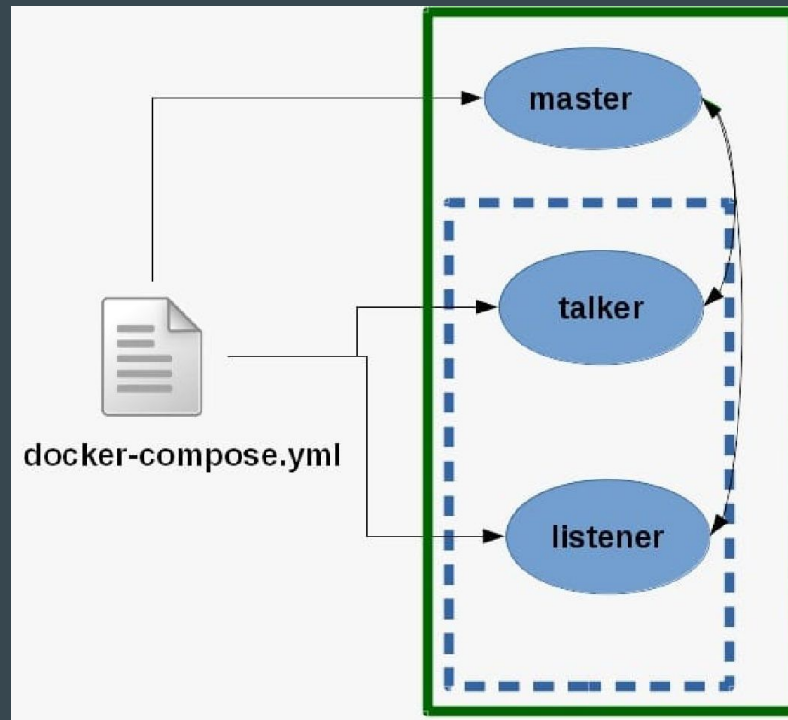
Phase 3

Phase 4

End

# Talker-Listener Model

This flowchart shows how a docker file is communicating with the master node and talker listener node







Phase 1

Phase 2

Phase 3

Phase 4

Results

## Results

From the previous demonstration, we can see that using Azure DevOps(docker) we can operate other devices connected to the Azure IoT hub In a similar way. The above demonstration shows the talker and listener using simulation, but can also be performed using two different devices using Azure IoT, which operates on the cloud. Also, there can be multiple devices connected at the same time. Basically, it is a broadcast sender and broadcast receiver, as we can have as many devices or nodes as listeners, or the other way, we can have as many talkers and one listener at the same time, and the device running smoothly.

# References

- 1) <https://docs.microsoft.com/en-us/azure/iot-hub/iot-concepts-and-iot-hub>
- 2) <https://www.slideshare.net/Simplilearn/introduction-to-devops-devops-tutorial-for-beginners-devops-training-for-beginners-simplilearn>
- 3) <https://www.slideshare.net/HomepreeRloy/devops-foundation>
- 4) <https://www.slideshare.net/Simplilearn/introduction-to-devops-devops-tutorial-for-beginners-devops-training-for-beginners-simplilearn>
- 5) <https://github.com/microsoft/ros-azure-iothub>
- 6) <https://github.com/cocheok/robotics-devops>
- 7) <https://microsoft.github.io/Win-RoS-Landing-Page/>
- 8) <https://cocheok.github.io/robotics-devops/development/>