



NUBOMEDIA: The First Open Source WebRTC PaaS

ACM Multimedia 2017 – Open Source Competition
25th October 2017 (Mountain View, CA, USA)

Boni García
Universidad Rey Juan Carlos (Spain)
boni.garcia@urjc.es

Table of contents

1. Introduction
2. NUBOMEDIA overview
3. Demo
4. Conclusions

Table of contents

1. Introduction

- Context
- Problem at hand
- Our proposal: NUBOMEDIA
- References

2. NUBOMEDIA overview

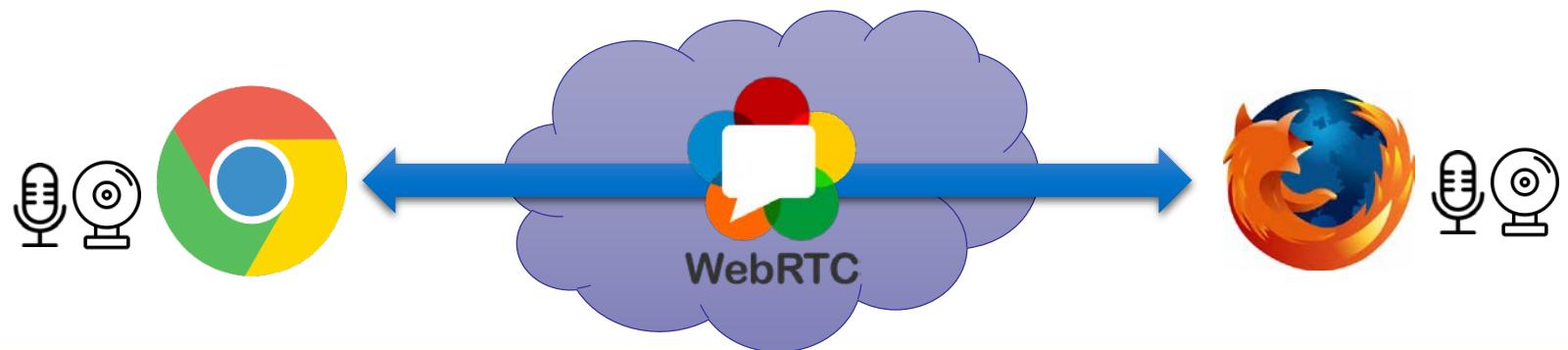
3. Demo

4. Conclusions

1. Introduction

Context

- **WebRTC** is the umbrella term for a number of technologies aimed to bring high-quality *Real Time Communications* to the Web
 - W3C (JavaScript APIs): *getUserMedia*, *PeerConnection*, *DataChannels*
 - IETF (protocol stack): ICE, SDP, TURN, STUN, ...



1. Introduction

Problem at hand

- Multimedia applications and services are becoming the main force of the Internet
 - For example WebRTC, but also Video Content Analysis (VCA) or Augmented Reality (AR)
- Deploying these types of technologies in common clouds infrastructures is complex and cannot be achieved easily

1. Introduction

Our proposal: NUBOMEDIA



- NUBOMEDIA is an open source **PaaS** (Platform as a Service)
- NUBOMEDIA exposes to developers the ability of **deploying and leveraging** applications with media capabilities:
 - WebRTC, media recording, group communications, VCA, AR, etc.

1. Introduction

Our proposal: NUBOMEDIA



- NUBOMEDIA has been conceived for simplifying the way developers use to deal with multimedia applications
 - From the developer's perspective, NUBOMEDIA capabilities are accessed through a set of **APIs and SDKs**
 - NUBOMEDIA applications can be deployed using the **NUBOMEDIA PaaS Manager**

1. Introduction

References

- Home page
<http://www.nubomedia.eu/>
- Developers guide
<http://nubomedia.readthedocs.io/>
- GitHub organization
<https://github.com/nubomedia/>
- Support for developers
<https://groups.google.com/forum/#!forum/nubomedia-dev>



Table of contents

1. Introduction

2. NUBOMEDIA overview

- Architecture
- Media API
- Room API
- PaaS Manager

3. Demo

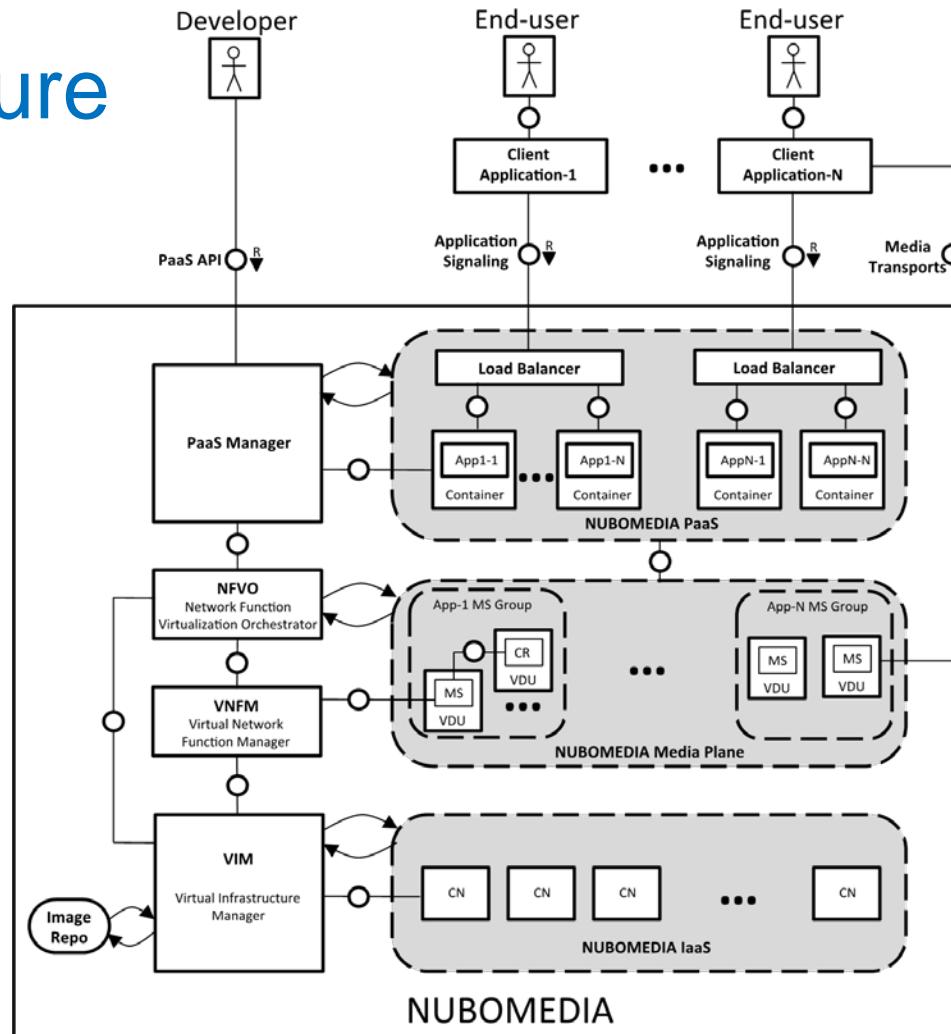
4. Conclusions

2. NUBOMEDIA overview

Architecture



OPEN BATON



KURENTO

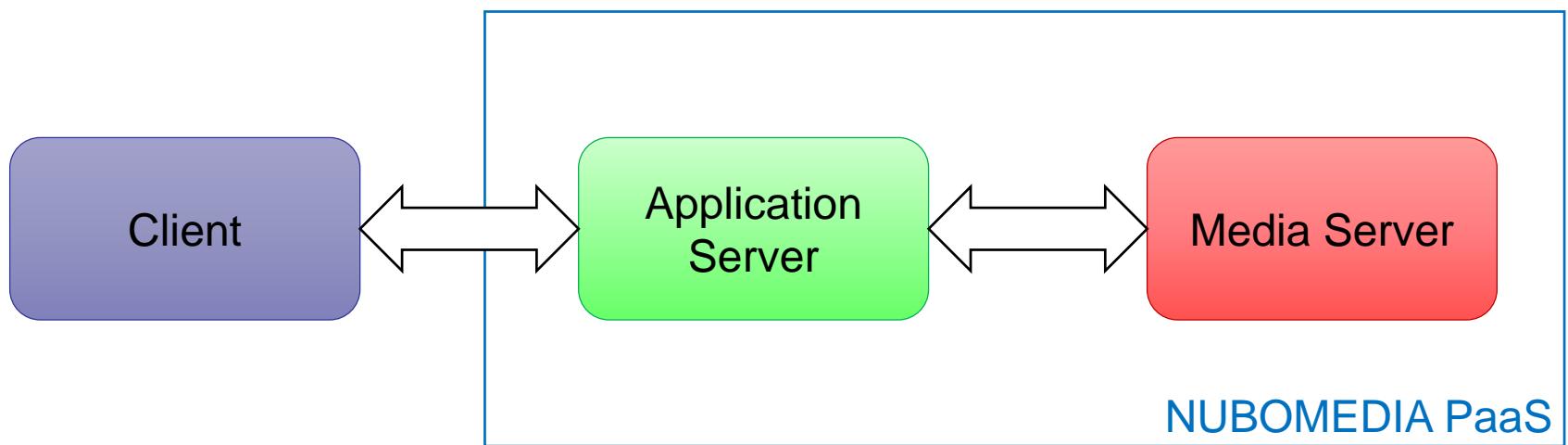


OPENSIFT

2. NUBOMEDIA overview

Architecture

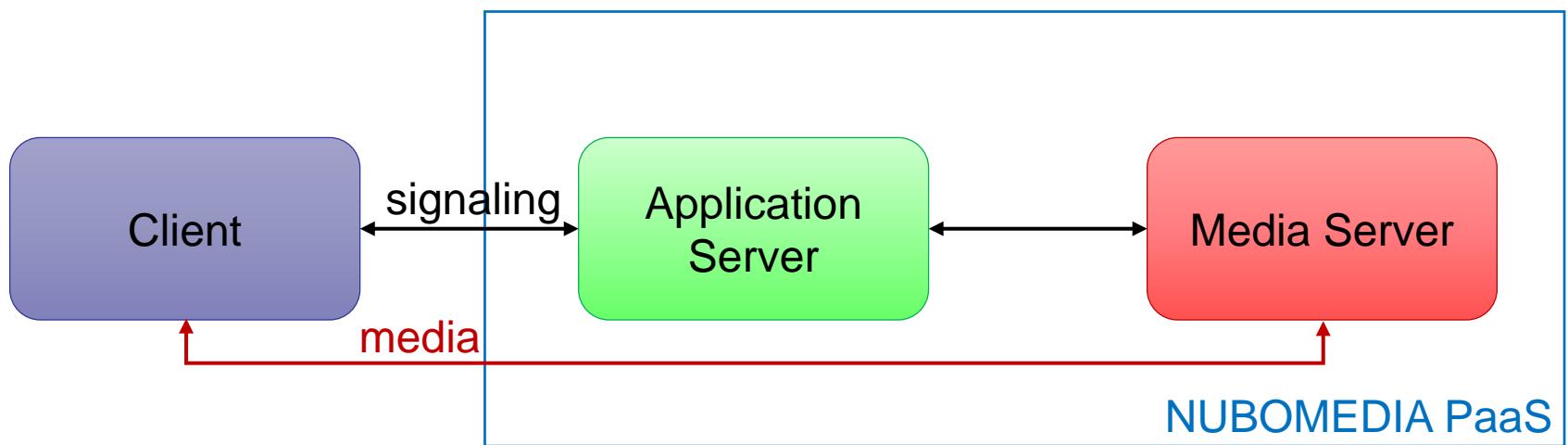
- A NUBOMEDIA application follows a three-tier model (inspired in the Web)



2. NUBOMEDIA overview

Architecture

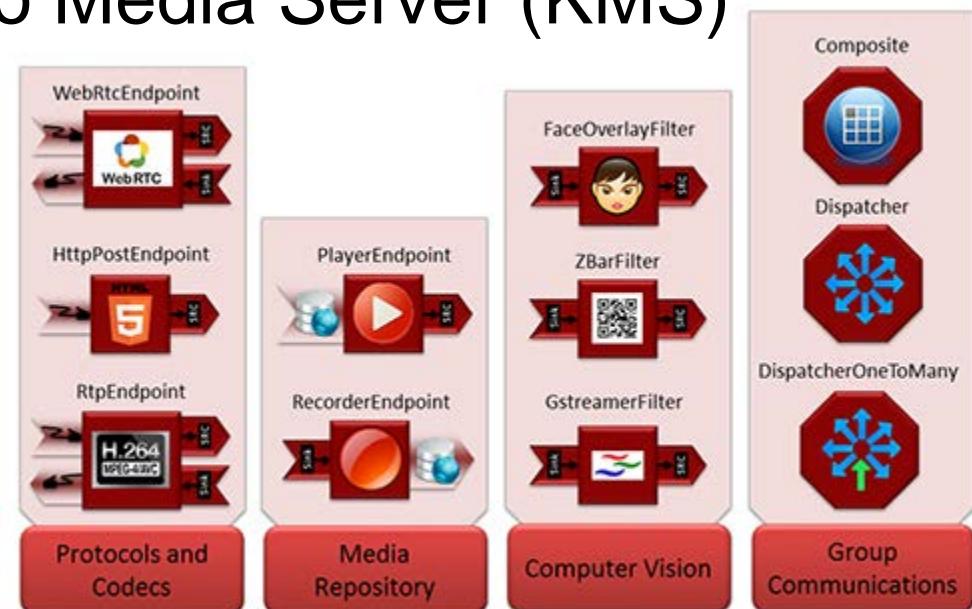
- Like every application with media capabilities, it is important to distinguish between the **media** and **signaling** plane



2. NUBOMEDIA overview

Media API

- NUBOMEDIA Media API allows Java developers consume the media services provided by Kurento Media Server (KMS)
- Concepts:
 - Media Element
 - Media Pipeline



2. NUBOMEDIA overview

Media API

- KMS instances are provided elastically by NUBOMEDIA
 - The number of available KMS instances depends on the PaaS Manager configuration
- Each KMS has a total amount of available points to create Media Pipelines and Media Elements
 - The total points depends on the number of VCPUs of the KMS
 - The type of the instance can be selected on the PaaS Manager configuration

Instance type	# VCPUs	KMS points
Medium	2	200
Large	4	400

2. NUBOMEDIA overview

Media API

- Each KMS is controlled by an instance of `KurentoClient`

```
<dependency>
  <groupId>org.kurento</groupId>
  <artifactId>kurento-client</artifactId>
</dependency>

<dependency>
  <groupId>de.fhg.fokus.nubomedia</groupId>
  <artifactId>nubomedia-media-client</artifactId>
</dependency>
```

Dependencies
(Maven)

- With each media session an instance of `KurentoClient` should be created

```
KurentoClient kurentoClient = KurentoClient.create();
```

- The number of available points per KMS decreases with each Media Element creation (scaling in/out)

2. NUBOMEDIA overview

Media API

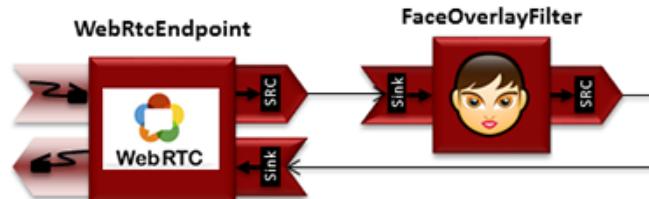
- Example: [nubomedia-magic-mirror](#)

```
// One KurentoClient instance per session
KurentoClient kurentoClient = KurentoClient.create();

// Media logic (pipeline and media elements connectivity)
MediaPipeline mediaPipeline = kurentoClient.createMediaPipeline();

WebRtcEndpoint webRtcEndpoint = new WebRtcEndpoint.Builder(mediaPipeline).build();
FaceOverlayFilter faceOverlayFilter = new FaceOverlayFilter.Builder(mediaPipeline).build();
faceOverlayFilter.setOverlaidImage("http://files.kurento.org/img/mario-wings.png", -0.35F,
    -1.2F, 1.6F, 1.6F);

webRtcEndpoint.connect(faceOverlayFilter);
faceOverlayFilter.connect(webRtcEndpoint);
```



NUBOMEDIA Tutorial: WebRTC Magic Mirror

This application shows a WebRtcEndpoint connected to itself (loopback) with a FaceOverlay filter in the middle (take a look to the Media Pipeline). To run this demo follow these steps:

1. Open this page with a browser compatible with WebRTC and click on Start button.
2. Grant the access to the camera and microphone. After the SDPICE candidates negotiation the loopback should start.
3. Click on Stop to finish the communication.

Local stream

Remote stream

Console

```

> ICE Candidate received
I Received message: {"id": "iceCandidate", "candidate": {"candidate": "candidate: # 1 TCP: 847249910 80.96.122.194"
> ICE candidate received
> Remote URL: 'blob:https://magic-mirror20351478233927019961541054960142882140.paa.s.nubomedia.eu/c43be32...
  
```

2. NUBOMEDIA overview

Room API

- The Room API is a high-level communications library that provides capabilities for managing multi-conference WebRTC sessions. It has the following components:
 - Room Server: a container-based implementation of the server, uses JSON-RPC over WebSockets for communications with the clients
 - Room JavaScript Client: module implementing a Room client for Web applications
 - Room Client: a client library for Java web applications or Android clients

2. NUBOMEDIA overview

Room API

- Example: [nubomedia-room-tutorial](#)

Server-side:
KurentoClient
management

```
<dependency>
    <groupId>org.kurento</groupId>
    <artifactId>kurento-room-server</artifactId>
</dependency>
<dependency>
    <groupId>org.kurento</groupId>
    <artifactId>kurento-room-client-js</artifactId>
</dependency>
```

Dependencies
(Maven)

```
public class SingleKmsManager extends KmsManager {

    @Override
    public KurentoClient getKurentoClient(KurentoClientSessionInfo sessionInfo) throws RoomException {
        return KurentoClient.create();
    }

    @Override
    public boolean destroyWhenUnused() {
        return true;
    }
}
```

2. NUBOMEDIA overview

Room API

- Example: [nubomedia-room-tutorial](#)

Client-side
room
management

```
var kurento = KurentoRoom(wsUri, function (error, kurento) {
    if (error) return console.log(error);

    room = kurento.Room({
        room: $scope.roomName,
        user: $scope.userName,
        updateSpeakerInterval: $scope.updateSpeakerInterval,
        thresholdSpeaker: $scope.thresholdSpeaker
    });

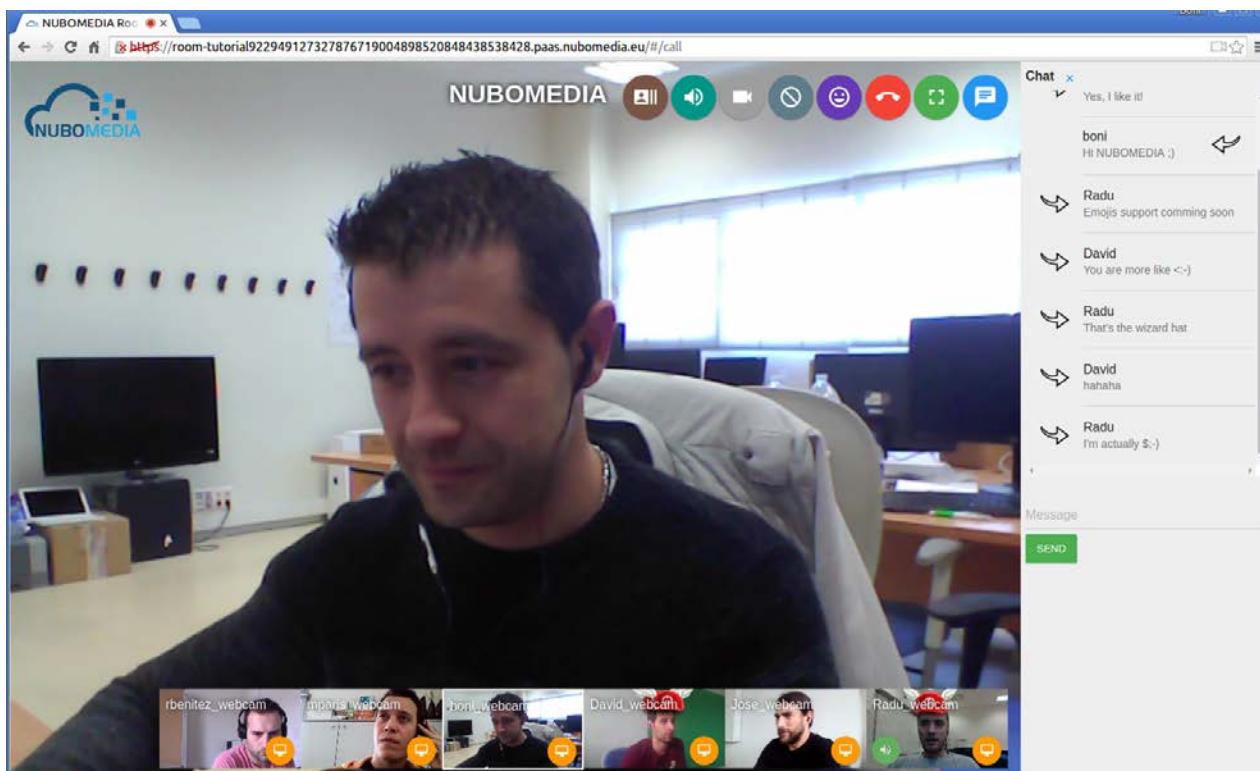
    var localStream = kurento.Stream(room, {audio: true, video: true, data: true});

    localStream.addEventListener("access-accepted", function () {
        room.addEventListener("room-connected", function (roomEvent) {
            var streams = roomEvent.streams;
            localStream.publish();
            ServiceRoom.setLocalStream(localStream.getWebRtcPeer());
            for (var i = 0; i < streams.length; i++) {
                ServiceParticipant.addParticipant(streams[i]);
            }
        });
        // ...
    });
});
```

2. NUBOMEDIA overview

Room API

- Example: [nubimedia-room-tutorial](https://room-tutorial922949127327876719004898520848438538428.paas.nubimedia.eu/#/call)



2. NUBOMEDIA overview

PaaS Manager

- NUBOMEDIA PaaS manager which controls the way in which the NUBOMEDIA applications are built and deployed
- Internally, the NUBOMEDIA PaaS uses **Docker containers** to deploy applications
- We need to include a **Dockerfile** in GitHub repository to be deployed on NUBOMEDIA



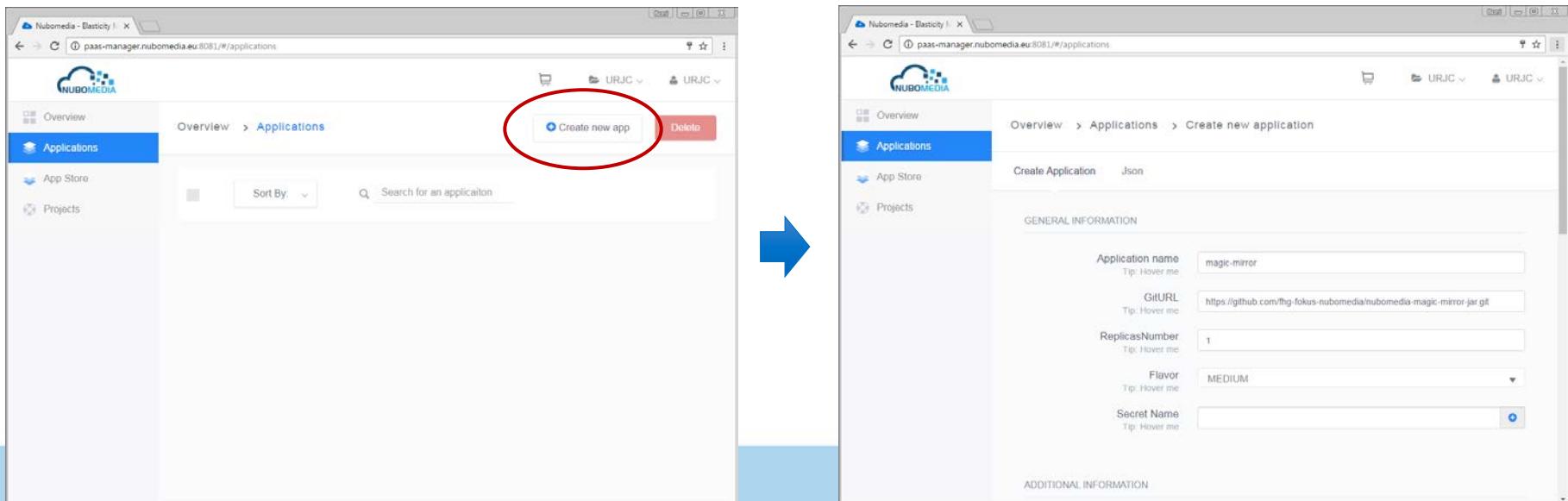
```
FROM nubomedia/apps-baseimage:src
MAINTAINER Nubomedia
ADD . /home/nubomedia
ENTRYPOINT cd /home/nubomedia && mvn spring-boot:run
```

<https://docs.docker.com/engine/reference/builder/>

2. NUBOMEDIA overview

PaaS Manager

- The capabilities provided by the PaaS Manager can be used by developers using the **PaaS GUI**
- NUBOMEDIA apps are deployed using GitHub repositories and a set of configuration parameters



2. NUBOMEDIA overview

PaaS Manager

- Most important configuration values:

The screenshot shows the 'GENERAL INFORMATION' section of the PaaS Manager. It includes fields for Application name (magic-mirror), GitURL (https://github.com/nubomedia/nubomedia-benchmark), ReplicasNumber (1), Flavor (MEDIUM), and Secret Name (empty). A 'GitHub URL repository' callout points to the GitURL field. A 'KMS host type:' callout points to the Flavor dropdown, listing Medium (2 VCPUs, 200 points) and Large (4 VCPUs, 400 points). A 'Number of KMSs' callout points to the ReplicasNumber field.

GENERAL INFORMATION

Application name
Tip: Hover me

magic-mirror

GitURL
Tip: Hover me

https://github.com/nubomedia/nubomedia-benchmark

ReplicasNumber
Tip: Hover me

1

Flavor
Tip: Hover me

MEDIUM

Secret Name
Tip: Hover me

KMS host type:
- Medium = 2 VCPUs (200 points)
- Large = 4 VCPUs (400 points)

Number of KMSs

ADDITIONAL INFORMATION

In/OUT scale

Enable

ScaleInOut
Tip: Hover me

0

Scale_out_threshold
Tip: Hover me

0

Table of contents

1. Introduction
2. NUBOMEDIA overview
- 3. Demo**
4. Conclusions

Table of contents

1. Introduction
2. NUBOMEDIA overview
3. Demo
4. Conclusions

4. Conclusions

- NUBOMEDIA is a PaaS platform enabling the convergence of WebRTC and advanced media processing
- It can be used by developers for saving tons of effort when creating applications with advance media capabilities
- Possible improvement: scheduling and placement algorithms for sessions based on policies beyond the points mechanisms



Thank you!
QA

Boni García
Universidad Rey Juan Carlos (Spain)
boni.garcia@urjc.es