Yes, you absolutely can use a Raspberry Pi 4B to display a video stream from your Reolink doorbell camera (or any of your Reolink cameras) on an HDMI monitor. Reolink cameras generally support **RTSP (Real-Time Streaming Protocol)**, which is the key to this.

Here's how you can approach it:

**1. Find the RTSP URL for your Reolink Camera**

Reolink cameras typically have a standard RTSP URL format. It usually looks like this:

rtsp://<username>:<password>@<IP\_address>:554/h264Preview\_01\_main

Or for the sub-stream (lower resolution, less bandwidth):

rtsp://<username>:<password>@<IP\_address>:554/h264Preview\_01\_sub

**Here's what you need to do to get the correct URL:**

* **Enable RTSP on your Reolink camera:**
  + Log in to your Reolink camera's web interface (using its fixed IP address in a web browser).
  + Navigate to **Network Settings** -> **Advanced** -> **Port Settings**.
  + Ensure **RTSP** is enabled and note the port number (default is 554).
* **Find your camera's username and password:** This is the username and password you set up for accessing the camera, not necessarily your Reolink app login.
* **Get the camera's IP address:** You already mentioned you have fixed IP addresses for your cameras.
* **Determine the stream type:** "main" for the primary, higher-resolution stream, or "sub" for the secondary, lower-resolution stream. For simple display, the "sub" stream might be sufficient and use less resources on the Pi.

**Example RTSP URL:** If your camera's IP is 192.168.1.100, username is admin, and password is yourpassword, the URL might be:

rtsp://admin:yourpassword@192.168.1.100:554/h264Preview\_01\_main

**Important Note:** Some Reolink cameras might have slightly different URL paths or require the channel number. If the above format doesn't work, check Reolink's support documentation for your specific doorbell model.

**2. Software to Display the Stream on Raspberry Pi**

Once you have the RTSP URL, you can use various media players on your Raspberry Pi to display it.

**a) VLC Media Player (Recommended for ease of use and features)**

VLC is a versatile media player that can handle RTSP streams. It's generally straightforward to install and use.

* **Installation:**

Bash

sudo apt update

sudo apt install vlc

* **Playing the stream (from the terminal):**

Bash

cvlc rtsp://<username>:<password>@<IP\_address>:554/h264Preview\_01\_main --fullscreen --no-audio --network-caching=500

* + cvlc: The command-line version of VLC.
  + --fullscreen: Displays the video in full-screen mode.
  + --no-audio: If you don't need audio, this can reduce resource usage.
  + --network-caching=500: Adjusts the network caching (in milliseconds). You might need to experiment with this value if you experience stuttering.

You can also use the graphical VLC if you have a desktop environment installed on your Pi. Open VLC, go to "Media" -> "Open Network Stream" and paste the RTSP URL.

**b) FFmpeg (More lightweight, but command-line only)**

FFmpeg is a powerful command-line tool for video and audio processing. It's often more resource-efficient than VLC for simply displaying a stream.

* **Installation:**

Bash

sudo apt update

sudo apt install ffmpeg

* **Playing the stream (using ffplay):**

Bash

ffplay -i "rtsp://<username>:<password>@<IP\_address>:554/h264Preview\_01\_main" -fs -loglevel quiet -framedrop -probesize 32 -analyzeduration 1000000 -rtsp\_transport tcp

* + ffplay: The player component of FFmpeg.
  + -i: Specifies the input URL.
  + -fs: Fullscreen.
  + -loglevel quiet: Suppresses most output messages.
  + -framedrop: Drops frames if the system can't keep up, prioritizing real-time playback.
  + -probesize and -analyzeduration: Can help with initial stream detection.
  + -rtsp\_transport tcp: Forces TCP for the RTSP connection, which can be more reliable than UDP.

**c) Omxplayer (Hardware-accelerated, but deprecated on newer Raspberry Pi OS versions)**

Omxplayer used to be the go-to for hardware-accelerated video playback on older Raspberry Pi OS versions (like Raspbian Buster). However, with newer versions (like Bookworm) and Raspberry Pi 4B, its hardware acceleration is deprecated or unsupported. If you're running an older OS, it might be an option.

* **Playing the stream:**

Bash

omxplayer "rtsp://<username>:<password>@<IP\_address>:554/h264Preview\_01\_main" --no-keys --loop

* + --no-keys: Prevents omxplayer from responding to keyboard input, which is good for a dedicated display.
  + --loop: Will restart the stream if it ends (useful for continuous monitoring).

**Recommendation:** Start with **VLC** (cvlc) first, as it's often the easiest to get working. If you encounter performance issues, try experimenting with ffplay and its various options.

**3. Setting up for Autostart (Optional, but recommended for a dedicated display)**

You'll likely want the stream to start automatically when the Raspberry Pi boots up. You can achieve this using a systemd service or by adding the command to a startup script.

**Using a systemd service (Recommended for robustness):**

1. **Create a service file:**

Bash

sudo nano /etc/systemd/system/camera-stream.service

1. **Add the following content (adjust the ExecStart line for your chosen player and RTSP URL):**

Ini, TOML

[Unit]

Description=Reolink Camera Stream

After=network-online.target

[Service]

ExecStart=/usr/bin/cvlc --fullscreen --no-audio --network-caching=500 rtsp://<username>:<password>@<IP\_address>:554/h264Preview\_01\_main

User=pi

Restart=always

RestartSec=5

StandardOutput=journal

StandardError=journal

[Install]

WantedBy=graphical.target

* + **Important:** Replace <username>, <password>, and <IP\_address> with your actual camera details.
  + If using ffplay, the ExecStart line would be: ExecStart=/usr/bin/ffplay -i "rtsp://<username>:<password>@<IP\_address>:554/h264Preview\_01\_main" -fs -loglevel quiet -framedrop -probesize 32 -analyzeduration 1000000 -rtsp\_transport tcp
  + User=pi: Runs the command as the pi user.
  + Restart=always: Ensures the stream restarts if it crashes.
  + RestartSec=5: Waits 5 seconds before attempting a restart.
  + WantedBy=graphical.target: Ensures it starts after the graphical environment is ready (if you're using one). If you're running in a pure command-line environment, you might use multi-user.target instead and ensure you're on a TTY that supports video output.

1. **Save and exit** (Ctrl+X, then Y, then Enter).
2. **Reload systemd and enable the service:**

Bash

sudo systemctl daemon-reload

sudo systemctl enable camera-stream.service

sudo systemctl start camera-stream.service

Now, your Raspberry Pi should automatically display the camera stream on the HDMI monitor when it boots up.

**Important Considerations:**

* **Raspberry Pi OS Version:** Newer versions of Raspberry Pi OS (like Bookworm) have moved away from omxplayer for hardware acceleration. VLC and FFmpeg will still work, but performance might vary. Ensure your OS is up-to-date.
* **Hardware Acceleration:** For smooth playback, especially for higher resolution streams, hardware acceleration is crucial. VLC and FFmpeg can leverage the Pi's GPU to varying degrees.
* **Network Stability:** A stable network connection to your Reolink cameras is essential for uninterrupted streaming.
* **CPU Usage:** Monitor your Raspberry Pi's CPU usage (using htop or top) to ensure it's not being overloaded, especially if you try to display multiple streams (which is a more advanced topic). For a single stream, a Pi 4B should handle it well.
* **Screen Blanking:** You might want to disable screen blanking on your Raspberry Pi so the monitor doesn't turn off. You can usually do this in raspi-config under "Display Options" or by editing /etc/xdg/lxsession/LXDE-pi/autostart (if using the desktop).

This setup should get you well on your way to having a dedicated display for your Reolink doorbell camera!