簡單即時伺服器 (SRS) 用於 RTMP 串流

讓我們深入了解如何使用 Simple Realtime Server (SRS) 部署一個邊緣伺服器來進行 RTMP 串流，並專注於理解和有效地配置它，包括 `rtmp.conf` 文件。SRS 是一個強大的高性能媒體伺服器，專為實時串流協議如 RTMP、HLS 和 HTTP-FLV 設計。它特別適合需要可擴展性的情況，例如內容分發網絡（CDNs），其中邊緣伺服器可以有效地將串流分發給許多客戶端，同時從原始伺服器獲取內容。我將逐步引導你完成這個過程，解釋邊緣部署的概念，並澄清如何處理配置文件如 `rtmp.conf`。

什麼是 SRS 和邊緣部署？

SRS 是一個開源伺服器，專為處理實時媒體串流，並專注於簡單和高效。它支持 RTMP（實時消息傳輸協議）進行低延遲的實時串流，以及其他協議如 HLS 和 WebRTC。在 SRS 中，邊緣伺服器作為客戶端（觀看者或發布者）和原始伺服器（串流的來源）之間的中介。邊緣伺服器僅在客戶端請求時從原始伺服器獲取串流，從而減少原始伺服器的負載，並實現可擴展的分發——可以將其視為專為實時串流設計的 CDN 的快取層。

在以下情況下部署邊緣伺服器與 SRS 是理想的：- 為大量觀看者提供服務而不會壓倒原始伺服器。- 允許發布者將串流推送到邊緣，然後將其轉發到原始伺服器。- 通過利用更便宜的邊緣節點來最小化昂貴的原始伺服器的頻寬使用。

逐步部署邊緣伺服器與 SRS 進行 RTMP

以下是如何將 SRS 設置為 RTMP 串流的邊緣伺服器。我假設你在 Linux 系統上工作（例如 Ubuntu），因為 SRS 為這些環境進行了優化。

1. 安裝 SRS　首先，你需要在機器上運行 SRS：- **下載 SRS**：從官方 GitHub 存儲庫獲取最新穩定版本（github.com/ossrs/srs）。截至 2025 年 2 月 26 日，你通常會克隆該存儲庫：`git clone https://github.com/ossrs/srs.git cd srs` - **構建 SRS**：SRS 使用簡單的構建過程，使用 `./configure` 和 `make`：`./configure  make` 這將伺服器編譯到 `objs` 目錄（例如 `objs/srs`）。- **測試二進制文件**：使用默認配置運行它以確保它正常工作：`./objs/srs -c conf/srs.conf` 默認情況下，它在端口 1935 上監聽 RTMP。檢查控制台輸出以確認。

2. 理解邊緣概念　在 SRS 中，邊緣伺服器以“遠程”模式運行，這意味著它不會生成串流，而是在客戶端請求時從原始伺服器拉取串流（用於播放）或將串流推送到原始伺服器（用於發布）。這種按需獲取是邊緣伺服器高效擴展 RTMP 交付的原因。

- **原始伺服器**：串流的來源（例如，編碼器如 OBS 將 RTMP 串流推送到的位置）。
- **邊緣伺服器**：客戶端連接的中繼，僅在需要時從原始伺服器獲取。

對於這個示例，假設你已經在 `192.168.1.100:1935` 運行 SRS 的原始伺服器（將此替換為你的實際原始 IP）。

3. 配置邊緣伺服器　SRS 使用配置文件來定義其行為。默認的 `srs.conf` 是一個很好的起點，但對於邊緣部署，你將創建一個特定的配置——我們稱其為 `edge.conf`。以下是如何設置它：

- **創建 `edge.conf`**：

  ```
  cd conf
  nano edge.conf
  ```

- **添加邊緣配置**：這是 RTMP 邊緣部署的最小 `edge.conf`：

  ```
  listen              1935;
  max_connections     1000;
  srs_log_tank        file;
  srs_log_file        ./objs/edge.log;
  vhost __defaultVhost__ {
      cluster {
          mode        remote;
          origin      192.168.1.100:1935;
      }
  }
  ```

  - `listen 1935`：邊緣監聽 RTMP 連接的端口 1935。
  - `max_connections 1000`：限制並發連接（根據你的伺服器容量進行調整）。
  - `srs_log_file`：記錄到文件以進行調試。
  - `vhost __defaultVhost__`：默認虛擬主機配置。
  - `cluster { mode remote; origin 192.168.1.100:1935; }`：將此伺服器設置為邊緣（`mode remote`）並將其指向原始伺服器。

- **保存並退出**：在 nano 中按 Ctrl+O，Enter，Ctrl+X。

4. 啟動邊緣伺服器　使用你的邊緣配置運行 SRS：

```
./objs/srs -c conf/edge.conf
```

檢查日誌（`./objs/edge.log`）以確認它正在運行並連接到原始伺服器。

5. 測試設置

- **發布串流**：使用 OBS 或 FFmpeg 將 RTMP 串流推送到原始伺服器：

  ```
  ffmpeg -re -i input.mp4 -c copy -f flv rtmp://192.168.1.100/live/livestream
  ```

  這裡，`live` 是應用程序名稱，`livestream` 是串流密鑰。

- **從邊緣播放**：使用 VLC 或其他 RTMP 客戶端從邊緣播放串流：

  ```
  rtmp://<edge-server-ip>/live/livestream
  ```

  將 `<edge-server-ip>` 替換為你的邊緣伺服器的 IP（例如 `192.168.1.101`）。邊緣將從原始伺服器獲取串流並將其提供給你。

6. 探索 `rtmp.conf`　SRS 默認不附帶 `rtmp.conf` 文件，但你可能會在教程或自定義設置中遇到對它的引用。它本質上是 RTMP 特定配置文件的命名約定。例如，SRS 文檔（ossrs.net）提供了一個用於實時 RTMP 串流的範例 `rtmp.conf`：

```
listen              1935;
max_connections     1000;
vhost __defaultVhost__ {
    tcp_nodelay     on;
    min_latency     on;
    play {
        gop_cache   off;
        queue_length 10;
    }
    publish {
        mr          off;
    }
}
```

- **目的**：此配置優化了原始伺服器上的低延遲 RTMP 串流，而不是邊緣。對於邊緣部署，你可以通過添加步驟 3 中的 `cluster` 塊來適應它。
- **關鍵設置**：
  - `tcp_nodelay on`：通過禁用 Nagle 的算法來減少延遲。
  - `min_latency on`：優先考慮低延遲而非緩衝。
  - `gop_cache off`：禁用圖像組的快取以進行實時播放。
  - `mr off`：禁用“合併讀取”以避免發布延遲。

對於邊緣，你可以將其與 `cluster` 設置結合使用，而不是單獨使用它。

進一步說明：邊緣機制和 RTMP

- **邊緣工作原理**：當客戶端請求 `rtmp://<edge-ip>/live/livestream` 時，邊緣會檢查是否擁有該串流。如果沒有，它將從原始伺服器（`192.168.1.100:1935`）拉取並本地快取以供其他客戶端使用。如果發布者將串流推送到邊緣，它將將串流轉發到原始伺服器。

- **RTMP 具體信息**：RTMP 是一個低延遲協議，適合實時串流。SRS 高效處理 RTMP，支持如絕對時間編碼（ATC）等功能，以便在伺服器之間進行同步，但默認情況下在邊緣模式下禁用，除非指定。
- **可擴展性**：添加多個指向同一原始伺服器的邊緣以處理數千個客戶端。SRS 支持故障轉移，通過列出多個原始伺服器（例如 `origin 192.168.1.100:1935 192.168.1.200:1935;`）。

提示和故障排除

- **防火牆**：確保原始伺服器和邊緣伺服器上的端口 1935 打開。
- **日誌**：檢查 `edge.log` 中的錯誤，例如與原始伺服器的連接失敗。
- **延遲**：如果原始伺服器延遲低，邊緣會添加最小延遲（通常 <1s）。
- **多個邊緣**：部署具有相同配置的其他邊緣，根據需要調整 `listen` 端口或 IP。

總結

部署 SRS 邊緣伺服器進行 RTMP 非常簡單，只要你理解原始邊緣關係。`edge.conf` 設置邊緣動態獲取或推送串流，而 `rtmp.conf` 類型的配置可以根據需要進一步微調 RTMP 性能。使用此設置，你已準備好高效地擴展實時串流——無論是為少量觀看者還是全球觀眾。想進一步調整它或將 HLS 與 RTMP 結合使用嗎？隨時告訴我！