BigGuy Corp – Multi-access Print Services Proposal

Made for : Ryan Lockhart, CEO

Prepared by : Ramtin Moghaddam

Table of Contents: **Page:**

Executive Summary

To support BigGuy Corp's operations, our group was tasked with developing a cross-platform printing system that enables communication between Linux and Windows-based systems. The main objective was to make sure Brasco (Windows 10) could submit print jobs to ABC Server (Debian) and BigGuy (Windows server 2019), while also receiving jobs from Linux mint(ABC Client).

We successfully managed to implement the architecture using CUPS (Common UNIX Printing System) and WSL (Windows Subsystem for Linux) to be able to bridge cross-platform compatibility issues furthermore a web-based PDF upload interface was also created to address the limitations of direct printing from Windows 10 to Linux-based CUPS printers.

The solution provides flexible, two-way printing with minimal infrastructure changes and leverages entirely free, open-source technologies meaning no proprietary software or costly upgrades were needed, helping keep operational costs low while allowing more room for future scalability.

In conclusion, our cross-platform print-sharing infrastructure addresses and eliminates the limitations the system had before, significantly improving print reliability across all departments. This helps us support a more diverse OS environment in a cost-effective way. The implemented solution improves operation efficiency and allows for seamless integration for future modifications or infrastructure.

Problem Overview

As been informed, BigGuy Corp has upgraded its systems which led to requiring compatibility between the newly introduced Linux systems (Debian and Linux) and legacy Windows Infrastructure. We first worked with Windows Server 2008 but faced compatibility issues with CUPS and SAMBA, which directed us to decide to upgrade to Windows Server 2016 however due to WSL not being supported on Server 2016, we yet again proceeded to upgrade to Windows Server 2019 to support the modern print subsystem.

The objective was:

. To help Brasco (Windows 10) to be able to print to both Debian and Windows server 2019 systems

. To allow ABC Client (Linux Mint) to send print jobs to both Debian and Windows Server 2019 systems.

. To avoid replacing the current system or a complete network infrastructure overhaul

There were limitations in Windows printing such as the lack of support for CUPS printers and the inability to share CUPS printers directly with Windows forcing us to develop a workaround solution that is stable, scalable, and cost-effective.

Implemented Solution

The core of our solution mainly relies on CUPS and WSL, we:

. Created and configured CUPS-PDF on both Debian and Windows Server 2019

. We shared the printers via HTTP so both clients (Windows 10 and Linux Mint) could access them through CUPS web GUI.

. Created a Flask-based custom webpage on Windows 10 WSL to support manual PDF uploads for printing

. Relocated printed documents from WSL (Linux layer) into Windows Server's filesystem using automated shell scripting

The implemented solution helped us to achieve full print compatibility from Linux Mint and Windows 10 to both printers and persistent sharing through HTTP helping avoid SAMBA issues.

Issues Encountered and Resolution Steps

CUPS-PDF File Upload Attempt Failed: We first tried to modify CUPS's configuration files to accept direct file uploads via the CUPS web interface and when that failed we attempted to change the printer's backend to 'file:/' with the intention of uploading files directly, however, this also failed because CUPS-PDF backend operates on printer-based output, not file-input and CUPS web interface does not support file upload for print jobs.

WSL CUPS Printer Isolation and Flask Upload Workaround: When printers are created inside WSL via CUPS, they are isolated from Windows and do not show up in the Windows print dialog. This prevented us from manually printing PDF files via Windows Applications to CUPS-PDF printer running in WSL, which led us to develop a Flask-based web interface that allows users to upload their own PDF files from the Windows environment into WSL for processing by CUPS. The script basically forwards the uploaded files to CUPS inside WSL, where the user selects which printer to send the PDF documents to (Debian or Windows Server). This workaround helped us address and resolve the missing GUI-level printer integration and helped us develop support for full manual print capability without relying on Windows printer discovery.

IIS PDF Handling Failure: At a point, we considered utilizing IIS (Internet Information Services) to handle PDF uploads however that was quickly discarded due to its lack of support for seamless print integration and limitations since it's not compatible with CUPS printers

Initial OS Compatibility Issues: Samba failed to share Bullzip/CutePDF printers from Windows server 2008/2016 to the Linux client which led us to upgrade to Windows Server 2019 to support WSL as it's not supported on Server 2016.

Technical Documentation

Printer Mapping:

. Linux Mint -> Debian (via CUPS HTTP)

. Linux Mint -> Windows Server 2019 (via WSL+CUPS HTTP)

. Windows 10 -> Debian (via CUPS HTTP)

. Windows 10 -> Windows Server 2019 (via CUPS HTTP)

Tools Used:

. Debian 12 XFCE (ABC Server)

. Linux Mint (ABC Client)

. Windows Server 2019 (BigGuy)

. Windows 10 (Brasco)

. CUPS

. WSL (Ubuntu 20.04)

. Flask (python3)

Implementation:

1. First, we install CUPS and CUPS-PDF on Debian and Linux Mint by running this command:

    . sudo apt update && sudo apt install cups cups-pdf -y

2. We proceed to add the CUPS-PDF printer via CUPS web interface on Debian and Windows Server (WSL):

    . navigate to http://localhost:631 on your web browser

. navigate to Administration -> Add Printer -> Choose CUPS-PDF backend -> Set name and location -> Share printer

3. We move on to sharing printers from Debian and WSL-CUPS:

. Edit /etc/cups/cupsd.conf via nano or vi and set/changed the following parameters:

. Set 'Listen 0.0.0.0:631'

. Set 'Browsing On'

. Set 'WebInterface Yes'

. Added 'Allow all' under <Location /> and <Location /admin>

4. We proceed to installing WSL and Ubuntu on Windows Server and Windows 10 by running the following command on PowerShell as an administrator:

. 'wsl --install -d Ubuntu-20.04'

5. Proceed to install and configure CUPS inside WSL by running the following commands :

. 'sudo apt update && sudo apt install cups cups-pdf -y'

. 'sudo nano /etc/cups/cupsd.conf' just add/modify the same parameters from step 3

6. Create the following script in WSL Windows server to move printed documents to Windows Server Filesystem:

. '#!/bin/bash

mv ~/PDF/* /mnt/c/Users/Public/Desktop/'

. Proceed to add this following line via nano or vi in /etc/cups/cups-pdf.conf:

. 'Out /mnt/c/Users/Public/Desktop/'

7. We move to creating a flask web interface (app.py) inside WSL for Windows 10, use the following script for app.py:

. 'from flask import Flask, request, render_template_string, redirect

```
import os
import subprocess

app = Flask(__name__)
UPLOAD_FOLDER = '/tmp/uploads'
os.makedirs(UPLOAD_FOLDER, exist_ok=True)

def get_printers():
    result = subprocess.run(['lpstat', '-p'], capture_output=True, text=True)
    return [line.split()[1] for line in result.stdout.splitlines() if line.startswith('printer')]

@app.route('/', methods=['GET', 'POST'])
def upload_file():
    printers = get_printers()
    if request.method == 'POST':
        file = request.files['file']
        printer = request.form['printer']
        if file and printer:
            filepath = os.path.join(UPLOAD_FOLDER, file.filename)
            file.save(filepath)
            subprocess.run(['lp', '-d', printer, filepath])
```

```
            return redirect('/')
    return render_template_string('''
        <h1>Upload PDF to Print</h1>
        <form method=post enctype=multipart/form-data>
          <input type=file name=file accept="application/pdf" required>
          <br><br>
          <label for="printer">Select printer:</label>
          <select name="printer" required>
            {% for printer in printers %}
              <option value="{{printer}}">{{printer}}</option>
            {% endfor %}
          </select>
          <br><br>
          <input type=submit value=Print>
        </form>
    ''', printers=printers)

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000)'
```

8. Adding the printers on Linux Mint and Windows 10:

. Access http://[IP of linux or Windows]:631 on local web browser

. Administration -> Add Printer -> Internet Printing Protocol (http)

. Enter http://192.168.100.10:631/printers/WindowsServerPDF

. Repeat for adding printer and this time enter Enter http://192.168.100.20:631/printers/DebianPDF

Benefits and Drawbacks:

Benefits:

Cross-platform compatibility: The implemented solution provides seamless printing between Linux and Windows systems without relying on proprietary software by using universal protocols such as HTTP and open-source tools like CUPS

Cost-effective Solution: All tools/components used for the solution (CUPS,WSL,Flask, and Ubuntu) are free and open-source meaning there is no need for paying for print servers or licensing fees.

Web-based Flexibility: With our created flask-based web interface, it gives the users the ability to manually upload and print PDF files directly to their selected printers from Windows 10, which resolves the limitations in GUI-based printing support.

Centralized Management: With CUPS configured for remote access, this can help the IT staff with monitoring and configuring printers from any machine on the network, improving efficiency

Minimal Infrastructure change: We mostly kept the same machines and only upgrade when necessary (such as windows server 2008 into server 2019)

Drawbacks:

CUPS printer isolation: Printers made and hosted in CUPS do not show up in the Windows print dialog which leads the users to rely on the custom upload interface rather than native print options

Manual setup: The implementation of the solution takes place across all machines with some non-trivial configurations, making the setup quite time-consuming

Performance Limitation: Using multiple layers (WSL, FLASK, and CUPS) led to some latency in job handling when compared to native printing drivers

Web Interface Dependency: If the flask service or interface fails or goes down, or even if WSL stops working, Windows 10 users lose the ability to manually upload and print files