**Recommended Approach: Generate an HTML Report with Cron, Display with PHP**

This approach has two parts:

1. **The Bash Script:** We'll modify it to output a self-contained HTML file instead of text with terminal codes.
2. **The Cron Job:** We'll set up a scheduled task to run this script automatically (e.g., every 5 minutes) to keep the report fresh.
3. **The PHP Page:** This will be a very simple page that just includes and displays the generated HTML report file.

This is the most professional and performant way to do it. Your web page will load instantly because it's just reading a static file, not executing a dozen commands on the server for every visitor.

**Step 1: Modify the Bash Script to Output HTML**

Save this modified script as generate\_report.sh. I have fixed the undefined $script\_name variable and converted all the terminal formatting to HTML and CSS.

**generate\_report.sh**

Generated bash

#!/bin/bash

# Define the output file path. Ensure this directory is writable by the user running the script.

# This should be inside your web root.

OUTPUT\_DIR="/var/www/html/server\_performance"

OUTPUT\_FILE="${OUTPUT\_DIR}/report.html"

# Create the directory if it doesn't exist

mkdir -p "$OUTPUT\_DIR"

# Start generating the HTML file. This uses a HEREDOC to write a block of text.

cat > "$OUTPUT\_FILE" << EOF

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta http-equiv="refresh" content="300"> <!-- Optional: Auto-refresh the page every 5 minutes -->

<title>Server Performance Report</title>

<style>

body { font-family: -apple-system, BlinkMacSystemFont, "Segoe UI", "Roboto", "Helvetica Neue", Arial, sans-serif; background-color: #f4f4f9; color: #333; margin: 2em; }

.container { background-color: #fff; padding: 2em; border-radius: 8px; box-shadow: 0 4px 8px rgba(0,0,0,0.1); max-width: 900px; margin: auto; }

h1 { color: #2c3e50; text-align: center; }

h2 { color: #3498db; border-bottom: 2px solid #ecf0f1; padding-bottom: 10px; margin-top: 30px; }

.green { color: #2ecc71; font-weight: bold; }

.data-grid { display: grid; grid-template-columns: repeat(auto-fit, minmax(200px, 1fr)); gap: 1em; margin-top: 1em; }

.data-item { background-color: #fdfdfd; padding: 15px; border-radius: 5px; border: 1px solid #ecf0f1; }

.data-item strong { display: block; margin-bottom: 5px; color: #7f8c8d; }

pre { background-color: #2c3e50; color: #ecf0f1; padding: 1em; border-radius: 5px; white-space: pre-wrap; word-break: break-all; font-family: "Menlo", "Monaco", "Consolas", monospace; }

.footer { text-align: center; margin-top: 2em; font-size: 0.9em; color: #95a5a6; }

</style>

</head>

<body>

<div class="container">

<h1>Server Performance Report</h1>

EOF

# --- Helper function to append content to the report file ---

# We use 'cat >>' to append to the file created above.

# The 'cat << EOF' syntax is called a "Here Document".

print\_header() {

cat >> "$OUTPUT\_FILE" << EOF

<h2>$1</h2>

EOF

}

# ------------------------ IP Address ------------------------

print\_header "IP Address"

IP\_ADDR=$(ip -4 addr show $(ip route get 1 | awk '{print $5}' | head -1) | grep -oP '(?<=inet\s)\d+(\.\d+){3}')

cat >> "$OUTPUT\_FILE" << EOF

<div class="data-item"><strong>Primary IP:</strong> <span class="green">${IP\_ADDR:-Not Found}</span></div>

EOF

# ------------------------ Hostname ------------------------

print\_header "Hostname"

if [ -f /etc/hostname ]; then

hostname=$(cat /etc/hostname)

else

hostname=$(uname -n)

fi

cat >> "$OUTPUT\_FILE" << EOF

<div class="data-item"><strong>Hostname:</strong> <span class="green">${hostname}</span></div>

EOF

# ------------------------ CPU Uptime ------------------------

print\_header "Uptime"

system\_uptime=$(cut -d' ' -f1 /proc/uptime)

total\_seconds=${system\_uptime%.\*}

days=$((total\_seconds / 86400))

hours=$(( (total\_seconds % 86400) / 3600 ))

minutes=$(( (total\_seconds % 3600) / 60 ))

cat >> "$OUTPUT\_FILE" << EOF

<div class="data-item"><strong>System Uptime:</strong> <span class="green">${days} days, ${hours} hours, ${minutes} minutes</span></div>

EOF

# ------------------------ Memory Usage -----------------------

print\_header "Memory Usage"

read total\_memory available\_memory <<< $(awk '/MemTotal/ {t=$2} /MemAvailable/ {a=$2} END {print t, a}' /proc/meminfo)

used\_memory=$((total\_memory - available\_memory))

total\_memory\_gb=$(awk -v t=$total\_memory 'BEGIN { printf("%.2f", t / 1048576) }')

used\_memory\_gb=$(awk -v u=$used\_memory 'BEGIN { printf("%.2f", u / 1048576) }')

used\_memory\_percent=$(awk -v u=$used\_memory -v t=$total\_memory 'BEGIN { printf("%.1f", (u / t) \* 100) }')

cat >> "$OUTPUT\_FILE" << EOF

<div class="data-grid">

<div class="data-item"><strong>Total Memory:</strong> <span class="green">${total\_memory\_gb} GB</span></div>

<div class="data-item"><strong>Used Memory:</strong> <span class="green">${used\_memory\_gb} GB (${used\_memory\_percent}%)</span></div>

</div>

EOF

# ------------------------ CPU Usage & Load ------------------------

print\_header "CPU Usage & Load"

cpu\_idle=$(top -bn1 | grep "Cpu(s)" | sed 's/.\*, \*\([0-9.]\*\)%\* id.\*/\1/')

cpu\_usage=$(awk -v idle="$cpu\_idle" 'BEGIN { printf("%.1f", 100 - idle) }')

load\_avg=$(cut -d' ' -f1,2,3 /proc/loadavg)

cat >> "$OUTPUT\_FILE" << EOF

<div class="data-grid">

<div class="data-item"><strong>Current Usage:</strong> <span class="green">${cpu\_usage}%</span></div>

<div class="data-item"><strong>Load Average (1, 5, 15m):</strong> <span class="green">${load\_avg}</span></div>

</div>

EOF

# ------------------------ Disk Usage ------------------------

print\_header "Disk Usage"

# We'll use preformatted text for the `df` output.

cat >> "$OUTPUT\_FILE" << EOF

<pre>

$(df -h --output=source,size,used,avail,pcent,target | grep -vE '^Filesystem|tmpfs|cdrom')

</pre>

EOF

# ------------------------ Web Server Response Time ------------------------

print\_header "Local Web Server Response"

# Note: This requires curl and a web server running on localhost

response\_time=$(curl -o /dev/null -s -w '%{time\_total}\n' http://localhost || echo "N/A")

cat >> "$OUTPUT\_FILE" << EOF

<div class="data-item"><strong>Response Time:</strong> <span class="green">${response\_time}s</span></div>

EOF

# ------------------------ Top Processes ------------------------

print\_header "Top 5 Processes (by CPU)"

cat >> "$OUTPUT\_FILE" << EOF

<pre>

<strong>COMMAND %CPU</strong>

$(ps -eo comm,%cpu --sort=-%cpu | head -n 6 | tail -n 5)

</pre>

EOF

print\_header "Top 5 Processes (by Memory)"

cat >> "$OUTPUT\_FILE" << EOF

<pre>

<strong>COMMAND %MEM</strong>

$(ps -eo comm,%mem --sort=-%mem | head -n 6 | tail -n 5)

</pre>

EOF

# --- Finalize the HTML file ---

LAST\_UPDATED=$(date)

cat >> "$OUTPUT\_FILE" << EOF

<div class="footer">

<p>Report generated on: ${LAST\_UPDATED}</p>

</div>

</div>

</body>

</html>

EOF

# This message will appear in the terminal when you run the script manually

echo "Performance report saved to: ${OUTPUT\_FILE}"

**Step 2: Set Up a Cron Job**

1. **Make the script executable:**

Generated sh

chmod +x /path/to/your/generate\_report.sh

IGNORE\_WHEN\_COPYING\_START

content\_copy download

Use code [with caution](https://support.google.com/legal/answer/13505487). Sh

IGNORE\_WHEN\_COPYING\_END

(Replace /path/to/your/ with the actual path).

1. **Open the crontab editor:**

Generated sh

crontab -e

IGNORE\_WHEN\_COPYING\_START

content\_copy download

Use code [with caution](https://support.google.com/legal/answer/13505487). Sh

IGNORE\_WHEN\_COPYING\_END

1. **Add a new line to run the script.** To run it every 5 minutes, add this line at the bottom:

Generated code

\*/5 \* \* \* \* /path/to/your/generate\_report.sh > /dev/null 2>&1

IGNORE\_WHEN\_COPYING\_START

content\_copy download

Use code [with caution](https://support.google.com/legal/answer/13505487).

IGNORE\_WHEN\_COPYING\_END

* + \*/5 \* \* \* \* means "At every 5th minute".
  + /path/to/your/generate\_report.sh is the full path to your script.
  + > /dev/null 2>&1 prevents cron from emailing you the output of the script every time it runs.

**Step 3: Create the PHP Viewer Page**

Now, create a PHP file in your web directory, for example /var/www/html/performance.php. This file will be very simple.

**performance.php**

Generated php

<?php

// Define the path to the report file generated by the bash script.

$reportFile = \_\_DIR\_\_ . '/server\_performance/report.html';

// Check if the report file exists and is readable.

if (file\_exists($reportFile) && is\_readable($reportFile)) {

// Set the content type header to HTML

header('Content-Type: text/html; charset=utf-8');

// Output the content of the file directly to the browser.

// readfile() is efficient for this as it writes the file's content

// directly to the output buffer.

readfile($reportFile);

} else {

// Display a user-friendly error message if the file is missing.

header('HTTP/1.1 500 Internal Server Error');

echo "<h1>Error: Performance Report Not Found</h1>";

echo "<p>The performance report file could not be found. Please ensure the generation script has been run successfully.</p>";

echo "<p>Expected location: " . htmlspecialchars($reportFile) . "</p>";

}

IGNORE\_WHEN\_COPYING\_START

content\_copy download

Use code [with caution](https://support.google.com/legal/answer/13505487). PHP

IGNORE\_WHEN\_COPYING\_END

Now, when you visit http://your-server-ip/performance.php in your browser, you will see a nicely formatted, modern-looking report page with all the server information, updated automatically every 5 minutes.