Experiment 5.1

Dockerize a React Application with Multi-Stage Build

CODE:

HTML CODE:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Dockerize React App – Multi-Stage Build Demo</title>
  <style>
    /* Global & Reset */
    :root {
      --color-bg: #0d1117;
      --color-card-bg: #161b22;
      --color-border: #30363d;
      --color-primary: #58a6ff; /* Blue */
      --color-success: #3fb950; /* Green */
      --color-error: #f85149; /* Red */
      --color-text: #e6edf3;
      --color-terminal-text: #c9d1d9;
    }

    body {
      font-family: 'Fira Code', 'Courier New', monospace;
      background: var(--color-bg);
      color: var(--color-text);
      display: flex;
      justify-content: center;
      align-items: center;
      min-height: 100vh;
      margin: 0;
      padding: 20px;
      box-sizing: border-box;
    }

    .container {
      text-align: center;
      width: 100%;
      max-width: 900px;
    }

    h1 {
      color: var(--color-primary);
      font-size: 2em;
      margin-bottom: 30px;
    }

    /* Stages Layout */
    .stages {
      display: flex;
      justify-content: space-between;
      gap: 20px;
      margin-bottom: 30px;
      flex-wrap: wrap;
    }

    /* Stage Card Styling */
```

```
.stage {
    flex: 1;
    min-width: 350px;
    background: var(--color-card-bg);
    border: 1px solid var(--color-border);
    border-radius: 12px;
    padding: 20px;
    transition: transform 0.3s ease, box-shadow 0.3s ease;
    box-shadow: 0 4px 15px rgba(0, 0, 0, 0.2);
    position: relative; /* For the completion ribbon */
}

.stage:hover {
    transform: translateY(-5px);
    box-shadow: 0 8px 25px rgba(0, 0, 0, 0.4);
}

.stage h2 {
    font-size: 1.4em;
    color: var(--color-primary);
    border-bottom: 1px dashed var(--color-border);
    padding-bottom: 10px;
    margin-top: 0;
}

/* Dockerfile Code Block */
.stage p {
    background: rgba(0, 0, 0, 0.3);
    border-radius: 6px;
    padding: 15px;
    margin: 15px 0;
    text-align: left;
    white-space: pre-wrap;
    font-size: 0.85em;
    line-height: 1.6;
    color: #d2a8ff; /* Lavender for code */
    height: 150px; /* Consistent height */
    overflow-y: auto;
}

/* Button Styling */
button {
    margin-top: 15px;
    background: var(--color-success);
    color: white;
    border: none;
    padding: 12px 25px;
    border-radius: 8px;
    cursor: pointer;
    font-weight: bold;
    letter-spacing: 0.5px;
    transition: background 0.2s ease, transform 0.1s;
}

button:hover:not(:disabled) {
    background: #2ea043;
    transform: translateY(-1px);
}

button:disabled {
    background: #30363d;
    cursor: not-allowed;
    opacity: 0.7;
}
```

```
/* Terminal Output Area */
.terminal {
    background: #000000; /* True black for terminal background */
    border: 1px solid var(--color-primary); /* Blue border highlight */
    border-radius: 12px;
    padding: 20px;
    text-align: left;
    min-height: 250px;
    box-shadow: 0 0 10px rgba(88, 166, 255, 0.2);
    transition: border-color 0.5s ease;
}

.output {
    white-space: pre-line;
    font-size: 0.9em;
    color: var(--color-terminal-text);
    height: 180px;
    overflow-y: auto;
    margin-bottom: 10px;
    padding-right: 10px;
}

/* Specific terminal colors for log steps */
.output .success { color: var(--color-success); }
.output .info { color: var(--color-primary); }
.output .step { color: #ffa657; }
.output .command { color: #d2a8ff; }
.output .error { color: var(--color-error); }

/* Progress Bar */
.progress-bar-container {
    height: 12px;
    background: var(--color-border);
    border-radius: 6px;
    overflow: hidden;
    margin-top: 10px;
}

.progress {
    height: 100%;
    width: 0%;
    background: linear-gradient(90deg, #3fb950, var(--color-primary));
    transition: width 0.4s ease;
}

/* Application Output (Simulated App) */
.app-output {
    background: #f0f8ff; /* Light, contrasting background */
    color: #282c34;
    border: 2px solid var(--color-success);
    border-radius: 10px;
    padding: 40px 20px;
    margin-top: 20px;
    text-align: center;
    font-size: 1.2em;
    font-family: sans-serif;
    box-shadow: 0 0 20px rgba(63, 185, 80, 0.5);
    display: none; /* Initially hidden */
}
.app-output h3 {
    color: #61dafb;
    font-size: 2em;
    margin-bottom: 5px;
}
```

```
      /* Responsive Layout */
      @media (max-width: 800px) {
        .stages {
          flex-direction: column;
          align-items: center;
        }
        .stage {
          width: 100%;
          min-width: unset;
        }
      }
    </style>
</head>
<body>
    <div class="container">
      <h1>🚀 Docker Multi-Stage Build: React App Demo</h1>

      <div class="stages">
        <div class="stage" id="stage1">
          <h2>Stage 1: Builder Image</h2>
          <p>
            <span class="command">FROM node:18-alpine <span class="info">AS builder</span></span><br>
            <span class="command">WORKDIR /app</span><br>
            <span class="command">COPY package*.json ./</span><br>
            <span class="command">RUN npm install</span><br>
            <span class="command">COPY . .</span><br>
            <span class="command">RUN npm run build</span>
          </p>
          <button id="btn-stage-1" onclick="simulateStage(1)">▶️ Run Builder Stage</button>
        </div>

        <div class="stage" id="stage2">
          <h2>Stage 2: Production Image</h2>
          <p>
            <span class="command">FROM nginx:alpine</span><br>
            <span class="command">COPY <span class="info">--from=builder</span> /app/build
/usr/share/nginx/html</span><br>
            <span class="command">EXPOSE 80</span><br>
            <span class="command">CMD ["nginx", "-g", "daemon off;"]</span>
          </p>
          <button id="btn-stage-2" onclick="simulateStage(2)" disabled>▶️ Run Production Stage</button>
        </div>
      </div>

      <div class="terminal" id="terminal">
        <div class="output" id="output">
          <span class="info">💻 Waiting to start build... Click "Run Builder Stage" to begin.</span>
        </div>
        <div class="progress-bar-container">
          <div class="progress" id="progress"></div>
        </div>
      </div>

      <div class="app-output" id="app-output">
        <h3>React App Running!</h3>
        <p>Served by NGINX on Port 80 inside the Docker Container.</p>
        <p>Image Size: **~20MB** | Build Time: **Fast** | Base Image: **nginx:alpine**</p>
        <p class="success">This demonstrates the success of a lightweight multi-stage build!</p>
      </div>
    </div>

    <script>
      // DOM Elements
      const outputElement = document.getElementById("output");
```

```
        const progressBar = document.getElementById("progress");
        const btnStage1 = document.getElementById("btn-stage-1");
        const btnStage2 = document.getElementById("btn-stage-2");
        const appOutput = document.getElementById("app-output");
        const terminalContainer = document.getElementById("terminal");

        let isBuilding = false;

        // Define the output lines with CSS classes for visual effect
        const STAGE_LINES = {
            1: [
                '<span class="step">[1/6] FROM node:18-alpine AS builder</span>',
                '<span class="step">[2/6] Setting up WORKDIR /app...</span>',
                '<span class="step">[3/6] COPY package*.json...</span>',
                '<span class="step">[4/6] RUN npm install: <span class="info">Downloading dependencies
(52MB)...</span></span>',
                '<span class="step">[5/6] COPY all source files...</span>',
                '<span class="step">[6/6] RUN npm run build: <span class="info">Compiling React static
assets...</span></span>',
                '<span class="success">✅ Builder stage finished. Intermediate image size: ~450MB.</span>',
                '<span class="info">➡️ Stage 2 is ready to start building the production image.</span>'
            ],
            2: [
                '<span class="step">[1/4] FROM nginx:alpine</span>',
                '<span class="step">[2/4] COPY <span class="info">--from=builder</span> /app/build
/usr/share/nginx/html: <span class="success">Copying 450KB static files...</span></span>',
                '<span class="step">[3/4] EXPOSE 80...</span>',
                '<span class="step">[4/4] CMD ["nginx", "-g", "daemon off;"]...</span>',
                '<span class="success">🚀 Final image built! Total size: ~20MB.</span>',
                '<span class="info">🎉 Starting container: <span class="command">docker run -p 8080:80 final-
app...</span></span>'
            ]
        };

        // Function to start the simulation for a specific stage
        function simulateStage(stage) {
            if (isBuilding) return;
            isBuilding = true;
            appOutput.style.display = 'none'; // Hide app output if re-running
            terminalContainer.style.borderColor = 'var(--color-primary)';

            const lines = STAGE_LINES[stage];
            if (!lines) return;

            // Disable buttons during the build
            btnStage1.disabled = true;
            btnStage2.disabled = true;

            outputElement.innerHTML = `<span class="info">>>>> docker build --tag final-app:${stage === 1 ? 'builder' :
'latest'} .</span>\n`;
            progressBar.style.width = "0%";
            outputElement.scrollTop = outputElement.scrollHeight;

            simulateBuild(lines, stage);
        }

        // Function to run the line-by-line simulation
        function simulateBuild(lines, stage) {
            let i = 0;
            const total = lines.length;
            const interval = 800;

            if (window.buildInterval) {
                clearInterval(window.buildInterval);
```

```
        }

        window.buildInterval = setInterval(() => {
            if (i < total) {
                outputElement.innerHTML += `<span class="command">> Step ${i + 1}/${total}:</span> ${lines[i]}\n`;
                outputElement.scrollTop = outputElement.scrollHeight;

                // Update progress bar width
                progressBar.style.width = ((i + 1) / total * 100) + "%";
                i++;
            } else {
                clearInterval(window.buildInterval);
                isBuilding = false;

                // Re-enable/toggle buttons after completion
                if (stage === 1) {
                    btnStage1.disabled = false;
                    btnStage2.disabled = false;
                    btnStage1.textContent = "✅ Builder Complete";
                    btnStage2.textContent = "▶️ Run Production Stage";

                } else if (stage === 2) {
                    btnStage1.textContent = "▶️ Run Builder Stage";
                    btnStage2.textContent = "✅ Build Complete & Running";
                    btnStage1.disabled = false;

                    // New: Show the final application output
                    setTimeout(() => {
                        terminalContainer.style.borderColor = 'var(--color-success)';
                        appOutput.style.display = 'block';
                        outputElement.innerHTML += '<span class="success">✅ Container running on http://localhost:8080.
See the successful application display below!</span>\n';
                        outputElement.scrollTop = outputElement.scrollHeight;
                    }, 500);
                }
            }
        }, interval);
    }

    // Initial setup
    document.addEventListener('DOMContentLoaded', () => {
        btnStage2.disabled = true;
    });
  </script>
</body>
</html>
```

OUTPUTS

EXPERIMENT 5.1

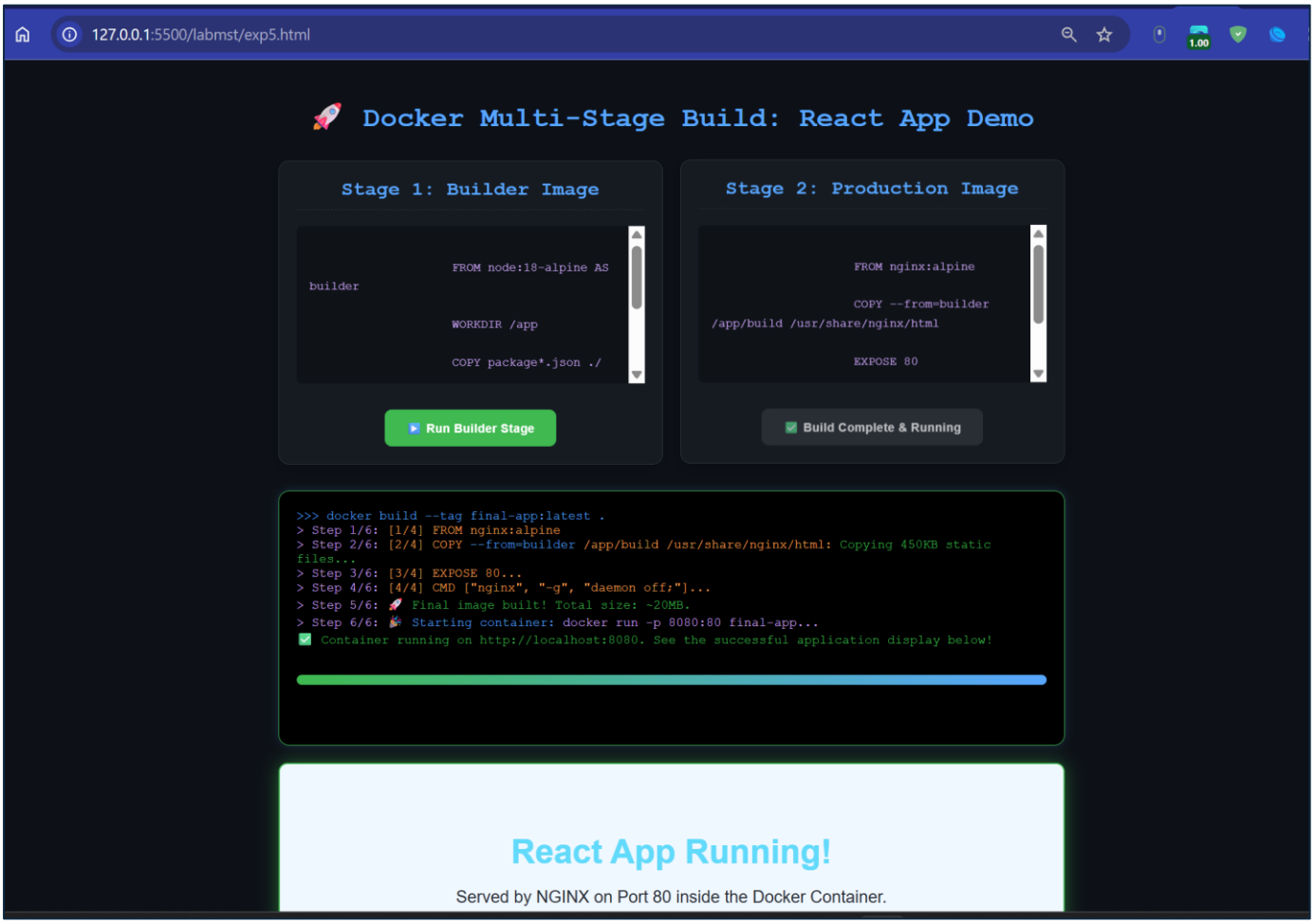

Figure 1: OUTPUT OF EXPERIMENT 5.1