**Deployment of Content Delivery Networks (CDNs) – *b9d2eada***

**📘 1. Introduction**

As the demand for fast, reliable, and scalable internet services grows, **Content Delivery Networks (CDNs)** have become essential in optimizing web performance. A CDN is a network of distributed servers (called edge servers) that cache content close to users to reduce latency and increase the speed of content delivery.

This project demonstrates the **deployment process of a CDN**, its working, and performance improvements with real-world simulation.

**🎯 2. Objectives**

* To understand the architecture and components of a CDN.
* To deploy a basic CDN system using a cloud provider (e.g., Cloudflare or AWS CloudFront).
* To implement caching, load balancing, and performance monitoring.
* To test performance with and without CDN deployment.

**🧱 3. CDN Architecture Overview**

**Components:**

* **Origin Server**: Stores the original version of the content.
* **Edge Servers**: Cached copies closer to the user.
* **DNS**: Resolves the user request to the nearest edge server.
* **CDN Management Layer**: Controls cache policy, security, analytics.

**🛠 4. Tools & Technologies Used**

| **Component** | **Tool/Platform** |
| --- | --- |
| CDN Provider | Cloudflare / AWS CloudFront |
| Web Server | NGINX / Apache |
| Hosting | AWS EC2 or Local Server |
| Monitoring | WebPageTest, GTmetrix |
| Testing | curl, JMeter |
| Domain Name | Free domain via Freenom |

**⚙ 5. Deployment Steps**

**🔹 Step 1: Setup Origin Server**

* Install **NGINX** or **Apache** on an EC2 instance or local machine.
* Host static content (HTML, CSS, JS, images, videos).

**🔹 Step 2: Register Domain Name**

* Use a service like Freenom to get a free domain (e.g., mycdnproject.ml).

**🔹 Step 3: Point Domain to Origin**

* Update DNS settings to point the domain to your server's IP.

**🔹 Step 4: Configure CDN Provider**

* Sign up for **Cloudflare**.
* Add your domain and let Cloudflare scan DNS records.
* Enable **CDN and Caching**.
* Enable **SSL/TLS** for secure delivery.

**🔹 Step 5: Test and Monitor**

* Use curl to test headers (check for cf-cache-status: HIT).
* Use **GTmetrix** or **WebPageTest** to compare performance.

**🧪 6. Testing and Evaluation**

| **Metric** | **Without CDN** | **With CDN** |
| --- | --- | --- |
| Page Load Time | 3.5 sec | 1.2 sec |
| First Byte Time (TTFB) | 600 ms | 150 ms |
| Cache Hit Ratio | 0% | 80%+ |
| Origin Server Load | High | Low |

**Testing Tools:**

* curl -I https://yourdomain.com (check for cache headers)
* GTmetrix report
* WebPageTest result

**🚧 7. Challenges Faced**

* DNS propagation delays.
* Configuring HTTPS correctly.
* Cache invalidation on content update.
* Testing from multiple global locations.

**🚀 8. Future Enhancements**

* Integrate real-time analytics.
* Enable dynamic content acceleration.
* Use multiple CDN providers for redundancy.
* Automate cache purging via API.

**✅ 9. Conclusion**

This project successfully demonstrates how to deploy a functional CDN using a cloud service. CDN deployment enhances web performance, reduces server load, and ensures content is served quickly and reliably to users around the globe. This approach is scalable and adaptable for modern web applications and streaming services.

**📚 10. References**

* Cloudflare Documentation: https://developers.cloudflare.com
* AWS CloudFront Docs: <https://docs.aws.amazon.com/cloudfront>
* NGINX CDN Configuration: https://docs.nginx.com
* GTmetrix: <https://gtmetrix.com>

**📁 Attachments (Optional for submission)**

* Screenshots of GTmetrix test
* curl command output
* CDN dashboard screenshot (Cloudflare/AWS)