Lambda@Edge

- is a feature of AWS that lets you **run Lambda functions at AWS CloudFront edge locations**, closer to the end users. It's designed to improve performance, reduce latency, and allow custom logic to be executed before the request reaches your origin server or after the response comes back.

🧭 What Is Lambda@Edge?

Lambda@Edge runs your code in response to events generated by **Amazon CloudFront**, which is AWS's content delivery network (CDN). It enables you to **modify requests and responses** as they pass through the CloudFront edge.

🔄 Event Triggers in Lambda@Edge

You can run functions at different points in the request/response cycle:

1. **Viewer Request** – When a user makes a request to CloudFront.
2. **Origin Request** – Before CloudFront forwards the request to your origin (like an S3 bucket or EC2).
3. **Origin Response** – After CloudFront receives the response from the origin.
4. **Viewer Response** – Before the response is sent to the user.

🧠 Key Use Cases

- **Redirect or rewrite URLs** (e.g., www to non-www, or clean URLs).
- **Add or modify HTTP headers** (e.g., security headers or caching policies).

- **A/B testing** by routing different users to different backends.
- **Geo-targeting** content based on the user's location.
- **Authorization and authentication** at the edge.
- **Language or device-based content delivery.**

✅ **Benefits**

Feature	Benefit
🌍 **Low Latency**	Code runs at edge locations close to users
⚙️ **No Servers to Manage**	Like AWS Lambda, it's fully serverless
🔒 **Security**	Code executes in AWS-managed environment
📈 **Scalable**	Automatically scales with global traffic
💡 **Flexible Logic at the Edge**	Customize requests/responses before reaching your servers

---

⚠️ **Considerations**

- **Deployment can take minutes** (because it's replicated globally).
- **Cold starts** can still occur.
- **Limited runtime environment** – supports only Node.js and Python (as of now).
- **Resource limits** – similar to AWS Lambda, with stricter memory and timeout caps.

🧪 **Example**

Let’s say you want to redirect mobile users to a mobile version of your site:

```
exports.handler = (event, context, callback) => {
    const request = event.Records[0].cf.request;
    const userAgent = request.headers['user-agent'][0].value;

    if (userAgent.includes("Mobile")) {
        return callback(null, {
            status: '302',
            statusDescription: 'Found',
            headers: {
                location: [{ key: 'Location', value: 'https://m.example.com' }],
            },
        });
    }

    return callback(null, request);
};
```
