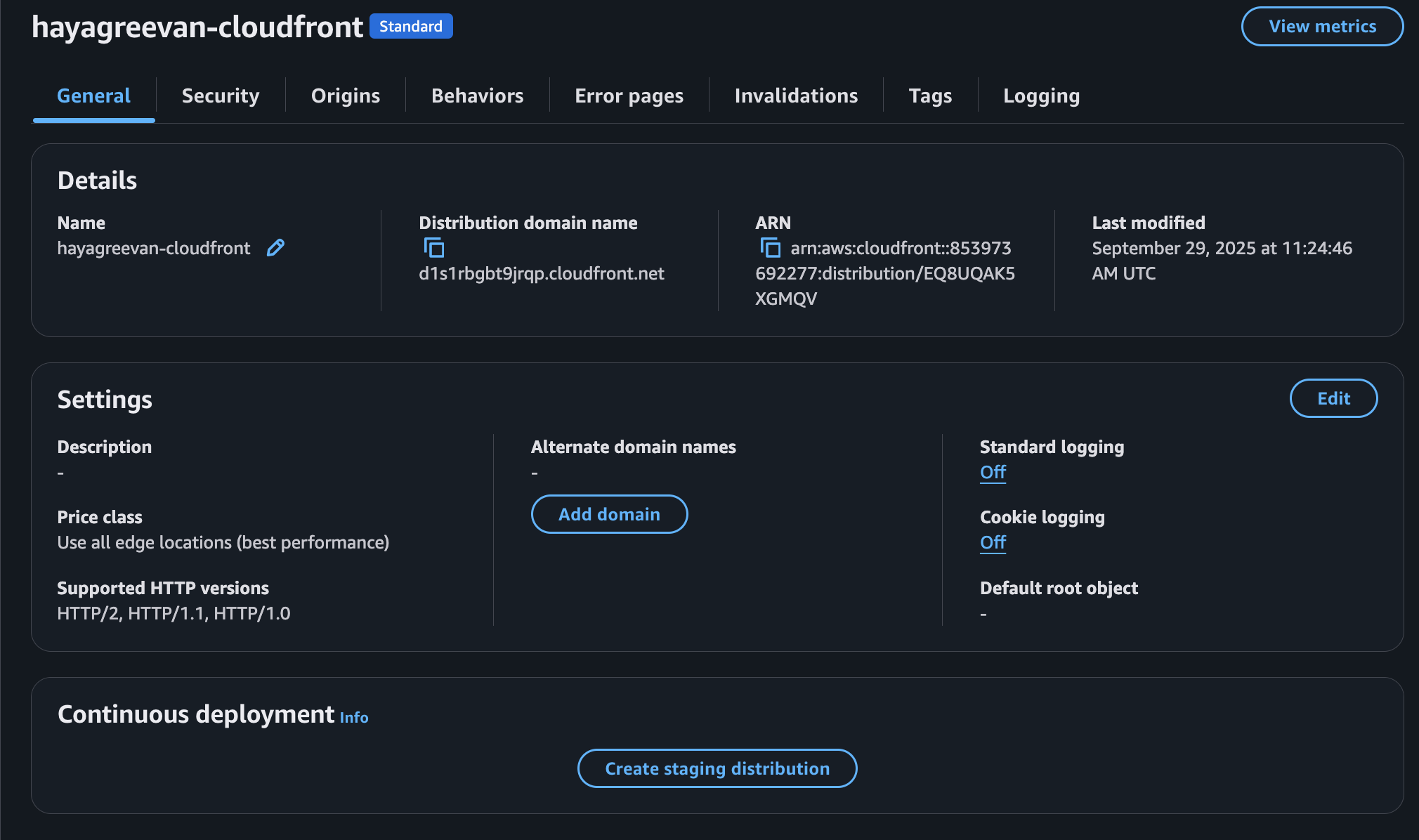
|  |  |  |
| --- | --- | --- |
| Domains & Content Delivery | * Route53 Overview * Records types and it use cases * CloudFront Overview * Integrations with S3 & ELB | **Section 10 - (103 - 120)**  **Section 15 - ( 165 - 170 )** |

1. You are an AWS cloud engineer optimizing static content delivery for a global media streaming company. Your goal is to enhance performance and reduce latency by leveraging Amazon CloudFront as a CDN.

* Configure a CloudFront distribution to serve static assets from an S3 bucket or EC2 instance. Ensure S3 or Ec2 should be accessible only via CloudFront.
  + **Custom Domains**
    - Create custom domain for your CDN.
  + **Test Delivery**:
    - Test the CloudFront URL using tools like VPNs to confirm content is served from the nearest edge location.
  + **Optimize Caching**:
    - Set cache behaviors with appropriate TTLs for different content types (e.g., longer for images, shorter for frequently updated files).



**Questions**

* How does CloudFront improve performance compared to serving content directly from EC2?
* What measures ensure the CloudFront distribution is secure and uses HTTPS?
* How would you monitor and optimize CloudFront performance?
* How would you troubleshoot user-reported slow loading times?
* CloudFront should be accessible only via specific geolocation. Eg: India, Singapore
* Explore on cache invalidation and implement cache strategy.

**Bonus Challenge:**

* **Enable HTTPS** for your CloudFront
  + Secure content delivery with SSL/TLS.
  + Use AWS Certificate Manager (ACM) to request a certificate and associate it with the distribution.

1. As an AWS Cloud Engineer managing a global e-commerce platform, you need to ensure low latency, high availability, and fault tolerance using AWS Route 53 for DNS management and traffic routing.

Requirements:

* **DNS Management**: Configure Route 53 hosted zones to manage domain and subdomain records (e.g., A/AAAA records for EC2 or load balancers).
* **Routing Policies**:
  + **Latency-Based**: Route users to the nearest AWS region for the lowest latency.
  + **Weighted**: Distribute traffic across multiple resources with adjustable weights.
  + **Failover**: Redirect traffic to a backup resource if the primary fails using health checks.
  + **Geolocation**: Direct users to region-specific endpoints (e.g., EU users to EU servers).
* **Health Checks**: Monitor backend services and reroute traffic from unhealthy resources.

**Questions**:

* How would you configure a Route 53 hosted zone and add DNS records for your services?
* How would you set up latency-based routing to improve user experience globally?
* How would you implement and adjust weighted routing policies to balance traffic?
* How would you configure failover routing to ensure service availability during outages?
* How would you set up health checks in Route 53 to detect failures and reroute traffic?

