Objective: Create an AWS CDK application in TypeScript that deploys a static website using appropriate AWS services for hosting and content delivery.

Requirements:

1	**Infrastructure as Code**	• Use AWS CDK to define and deploy the infrastructure needed to host and deliver a static website. • Ensure that relevant security features have been considered. • Ensure that relevant tests and code quality tools are used.
2	**Website content**	• Include a simple HTML file (e.g., index.html) as the website content in your project directory. • Ensure this HTML file is deployed as part of the CDK deployment process.
3	**Deployment instructions**	• Provide clear and concise instructions in a README.md file on how to deploy your CDK application. • Include steps to validate that the website is accessible.
4	**Future improvements**	• Describe potential improvements you might make in the future, focusing on the following aspects: • Security: Measures to enhance the security of the infrastructure and the static website. • Observability: Methods to improve monitoring, logging, and overall observability of the deployed resources. • Cost optimization: Strategies to reduce costs associated with hosting and delivering the static website.

Solution:

To Create an AWS CDK application in TypeScript that deploys a static

website using appropriate AWS services for hosting and content delivery follow the below step by step guide to achieve the goal.

1. Infrastructure as Code (IaC)	**Step: Install AWS CDK** Make sure you have Node.js installed, then install the AWS CDK globally: **npm install -g aws-cdk**
2.Create a New CDK Project & Initialize with directory	Create a new directory for CDK project and initialize it **mkdir static-website-cdk** **cd static-website-cdk** **cdk init app --language typescript**
3.Add Dependencies / Packages	Install the required AWS CDK packages / plugins for S3 and AWS CloudFront service **npm install @aws-cdk/aws-s3 @aws-cdk/aws-s3-assets @aws-cdk/aws-cloudfront @aws-cdk/aws-cloudfront-origins**
4.Define the CDK Stack	**Edit lib/static-website-cdk-stack.ts to define the infrastructure:**
5.Create the Website Content	<! -- website/index.html --> <!DOCTYPE html> <html lang="en"> <head>

	<meta charset="UTF-8"> <meta name="viewport" content="width=device-width, initial-scale=1.0"> <title>Static Website</title> </head> <body> Welcome to My Static Website! </body> </html> Create a directory named website in the root of your CDK project and add a simple index.html file:
<u>Deployment Instructions Step by Step guide</u>	**Bootstrap CDK Environment** bootstrapped your environment **cdk bootstrap**
1.Deploy the Stack	Deploy your CDK stack with the below command **cdk deploy**
2.Verify URL / Validate the Deployment	Once done with deployment, check the output for the URL of your static website. Open this URL in a web browser to verify /validate that the website is accessible.
3. Future Improvements	**Security** • **Enable HTTPS**: Use an ACM (AWS Certificate Manager) certificate with CloudFront to ensure that the site is served over HTTPS.

	• **Implement WAF**: Use AWS Web Application Firewall (WAF) to protect your application from common web exploits. **Observability** • **Enable CloudWatch Logging**: Configure CloudFront to send logs to CloudWatch for monitoring and analysis. • **Use AWS X-Ray**: For more advanced observability, integrate AWS X-Ray to trace requests through your application. **Cost Optimization** • **Optimize S3 Storage**: Use S3 storage classes like Intelligent-Tiering or Glacier for infrequently accessed data. • **Minimize CloudFront Requests**: Optimize caching policies and object invalidation to reduce unnecessary CloudFront requests.
Access Your Website	After deployment, check the output for the URL of your static website and open it in your browser.
README.md	**Create a README.md file with deployment instructions:** Add all the required steps for requirement and deployment **# Static Website CDK** **## Deployment Instructions**

Code:

```
import * as cdk from '@aws-cdk/core';
import * as s3 from '@aws-cdk/aws-s3';
import * as s3deploy from '@aws-cdk/aws-s3-deployment';
import * as cloudfront from '@aws-cdk/aws-cloudfront';
import * as iam from '@aws-cdk/aws-iam';

export class StaticWebsiteStack extends cdk.Stack {
  constructor(scope: cdk.Construct, id: string, props?: cdk.StackProps) {
    super(scope, id, props);

    // Create an S3 bucket for the static website
    const websiteBucket = new s3.Bucket(this, 'WebsiteBucket', {
      websiteIndexDocument: 'index.html',
      websiteErrorDocument: 'error.html',
      publicReadAccess: true,
      removalPolicy: cdk.RemovalPolicy.DESTROY, // Only for dev/test environments
    });

    // Deploy static website content to S3 bucket
    new s3deploy.BucketDeployment(this, 'DeployWebsite', {
      sources: [s3deploy.Source.asset('./website-content')],
      destinationBucket: websiteBucket,
    });
```

```
    // Create a CloudFront distribution
    const distribution = new cloudfront.CloudFrontWebDistribution(this, 'WebsiteDistribution', {
      originConfigs: [
        {
          s3OriginSource: {
            s3BucketSource: websiteBucket,
          },
          behaviors: [{ isDefaultBehavior: true }],
        },
      ],
      defaultRootObject: 'index.html',
    });

    // Output the website URL
    new cdk.CfnOutput(this, 'WebsiteURL', {
      value: distribution.distributionDomainName,
      description: 'The URL of the static website',
    });
  }
}
```