What is AWS SAM and how does it relate to CloudFormation?
Follow-up: When would you choose SAM over CDK or plain CloudFormation for serverless applications?

🟡 **What is AWS SAM?**

AWS SAM stands for **Serverless Application Model**.
It's a **tool** that helps you build and deploy **serverless applications** on AWS **easily and quickly**.

☁️ **Serverless means:**

- You don't manage servers.
- You use AWS services like **Lambda**, **API Gateway**, and **DynamoDB**.
- AWS automatically handles the scaling and infrastructure.

🔧 **What Does SAM Do?**

🧩 Imagine CloudFormation is like writing code in assembly (low-level, long, complex).

👉 **SAM is like Python** — short, clean, and easy.

With just **a few lines of YAML**, you can:

- Define a **Lambda function**
- Attach an **API Gateway**
- Connect to **DynamoDB**
- Set **permissions** — all together!

🔄 **How SAM Works**

1. You write a simple YAML file (template.yaml)
2. You use the **SAM CLI** to:
   - Build the code (sam build)
   - Deploy to AWS (sam deploy)
   - Test locally (sam local start-api)
3. Behind the scenes, SAM converts your YAML into **CloudFormation**.

  - So you still get rollback, versioning, audit logs — all benefits of infrastructure-as-code.

🎯 When Should You Use SAM?

Use Case	Use SAM	Use CDK	Use CloudFormation
Simple Serverless App (Lambda + API)	✅ Best	✅ Possible	✅ Possible
You like YAML	✅ Yes	❌ No	✅ Yes
Need Local Testing	✅ Yes (SAM CLI)	🔶 Some support	❌ None
Complex Logic / Reuse	❌ No	✅ Best	🔶 Verbose
Large Infra Teams	🔶 OK	✅ Best	✅ Yes

🔍 Real-World Example: Hello World API

📁 Project Structure

cpp

```
sam-app/
├── template.yaml
└── hello_world/
    ├── app.py
    └── requirements.txt
```

🧾 template.yaml

yaml

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31

Resources:
  HelloWorldFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: hello_world/
      Handler: app.lambda_handler
      Runtime: python3.9
      Events:
        HelloApi:
          Type: Api
          Properties:
            Path: /hello
            Method: GET
```

🐍 app.py

python

```
def lambda_handler(event, context):
    return {
      "statusCode": 200,
      "body": "Hello from SAM!"
  }
```

🚀 Deploy & Test

```
bash
------
sam build
sam deploy --guided  # You’ll get a public URL
```

🔗 You get:

https://<your-api>.execute-api.us-east-1.amazonaws.com/Prod/hello

🧪 Test Locally:

```
bash
------
sam local start-api
# Visit: http://localhost:3000/hello
```

AWS SAM is great for **building small, serverless apps using Lambda and API Gateway**. It saves time **by reducing YAML**, allows **local testing with SAM CLI**, and still uses CloudFormation underneath. I used SAM to build a REST API for a startup in days, tested locally, and deployed with a single command — it was fast, reliable, and serverless from day one.

What is AWS SAM primarily used for?

A. Automating EC2 deployments
B. Building and deploying serverless applications on AWS
C. Creating Kubernetes clusters
D. Visualizing infrastructure
Answer: B

What does AWS SAM transform into before actual resource provisioning?

A. Docker Compose
B. Terraform scripts
C. Native CloudFormation templates
D. AWS Lambda runtime
Answer: C

---

4. Which AWS service does not directly benefit from AWS SAM’s high-level abstractions?

A. Lambda
B. API Gateway
C. DynamoDB
D. EC2
Answer: D