**Q1- Why it is recommended to interact with AWS using CLI during serverless development and deployment. Which AWS service helps you achieve this environment? [ENT]**

**Ans :-** The [AWS Command Line Interface (AWS CLI)](https://aws.amazon.com/cli/) is an open source tool that enables developers to interact with AWS services using a command line shell. In many cases, the AWS CLI increases developer velocity for building cloud resources and enables automating repetitive tasks. It is an important piece of any serverless developer’s toolkit. Follow these instructions to [install and configure the AWS CLI](https://docs.aws.amazon.com/cli/latest/userguide/install-cliv2.html.) on your operating system.

AWS enables you to build infrastructure with code. This provides a single source of truth for AWS resources. It enables development teams to use version control and create deployment pipelines for their cloud infrastructure. [AWS Cloud Formation](https://aws.amazon.com/cloudformation/) provides a common language to model and provision these application resources in your cloud environment.

**Q2- What are the benefits of AWS SAM? [NET]**

**Ans:-**

* Single-deployment configuration. AWS SAM makes it easy to organize related components and resources, and operate on a single stack. ...
* Extension of AWS Cloud Formation.
* Built-in best practices.
* Local debugging and testing.
* Deep integration with development tools.

**Q3-What is S3 archival feature and in which scenarios it is used? How does it reduce the operation cost? [ET]**

Ans:-S3 is an easy-to-use, scalable, and cheap storage service from Amazon. You can use S3 to store any amount of data for a wide range of use cases. Static website hosting, data archival, and software delivery are a few general scenarios where S3 would be a perfect tool.

After you understand how you're charged on Amazon S3, consider the following tips for reducing your billing costs: Clean up incomplete multipart uploads. Delete previous versions of

that you don't need. Review your storage-class transition costs.

**Q4-Is AWS SAM same as normal CLI? Why or why not?[ET]**

**Ans:-**  As a companion to AWS SAM, the AWS SAM CLI is a command line tool that if it does not exist and uses it for any other applications in the same Region building new features and security updates on a regular cadence.

***Q:5—*What does SAM stands for? Why it is named so? [NET]**

**Ans:-**

**AWS Serverless Application Model**

* In Amazon Web Services (AWS), Serverless Application Model (SAM) is an open-source framework for building serverless applications.
* AWS SAM provides shorthand syntax to express functions, APIs, databases, and event source mappings. During deployment, SAM transforms and expands the SAM syntax into AWS CloudFormation syntax. Then, CloudFormation provisions your resources with reliable deployment capabilities.
* [**AWS SAM**](https://docs.aws.amazon.com/serverless-application-model/latest/developerguide/what-is-sam.html), short for Serverless Application Model, is an open source framework you can use to build serverless applications on AWS. It provides a shorthand syntax you can use to describe your application (Lambda functions, API endpoints, DynamoDB tables, and other resources) using a simple YAML template. During deployment, SAM transforms and expands the shorthand SAM syntax into an [AWS CloudFormation](https://aws.amazon.com/cloudformation/) template. Then, CloudFormation provisions your resources in a reliable and repeatable fashion.
* The [**AWS SAM CLI**](https://docs.aws.amazon.com/serverless-application-model/latest/developerguide/serverless-sam-reference.html#serverless-sam-cli), formerly known as SAM Local, is a command-line interface that supports building SAM-based applications. It supports local development and testing, and is also an active open source project. The CLI lets you choose between Python, Node, Java, Go, .NET, and includes a healthy collection of templates to help get you started.

You can use AWS SAM to define your serverless applications. AWS SAM consists of the following components:

**AWS SAM** **Template Specification**

You use this specification to define your serverless application. It provides you with a simple and clean syntax to describe the functions, APIs, permissions, configurations, and events that make up a serverless application. You use an AWS SAM template file to operate on a single, deployable, versioned entity that's your serverless application.

**AWS SAM Command Line Interface (AWS SAM CLI)**

You use this tool to build serverless applications that are defined by AWS SAM templates. The CLI provides commands that enable you to verify that AWS SAM template files are written according to the specification, invoke Lambda functions locally, step-through debug Lambda functions, package and deploy serverless applications to the AWS Cloud, and so on.

**Benefits of using AWS SAM**

Because AWS SAM integrates with other AWS services, creating serverless applications with AWS SAM provides the following benefits:

* **Single-Deployment Configuration**

AWS SAM makes it easy to organize related components and resources, and operate on a single stack. You can use AWS SAM to share configuration (such as memory and timeouts) between resources, and deploy all related resources together as a single, versioned entity.

* **Extension of AWS CloudFormation**

Because AWS SAM is an extension of AWS CloudFormation, you get the reliable deployment capabilities of AWS CloudFormation. You can define resources by using AWS CloudFormation in your AWS SAM template. Also, you can use the full suite of resources, intrinsic functions, and other template features that are available in AWS CloudFormation.

* **Built-In Best Practices**

You can use AWS SAM to define and deploy your infrastructure as config. This makes it possible for you to use and enforce best practices such as code reviews. Also, with a few lines of configuration, you can enable safe deployments through CodeDeploy, and can enable tracing by using AWS X-Ray.

* **Local Debugging and Testing**

The AWS SAM CLI lets you locally build, test, and debug serverless applications that are defined by AWS SAM templates. The CLI provides a Lambda-like execution environment locally. It helps you catch issues upfront by providing parity with the actual Lambda execution environment. To step through and debug your code to understand what the code is doing, you can use AWS SAM with AWS toolkits like the [AWS Toolkit for JetBrains](https://docs.aws.amazon.com/toolkit-for-jetbrains/latest/userguide/), [AWS Toolkit for PyCharm](https://aws.amazon.com/pycharm/), [AWS Toolkit for IntelliJ](https://aws.amazon.com/intellij/), and [AWS Toolkit for Visual Studio Code](https://aws.amazon.com/visualstudiocode/). This tightens the feedback loop by making it possible for you to find and troubleshoot issues that you might run into in the cloud.

* **Deep Integration with Development Tools**

You can use AWS SAM with a suite of AWS tools for building serverless applications. You can discover new applications in the [AWS Serverless Application Repository](https://docs.aws.amazon.com/serverlessrepo/latest/devguide/). For authoring, testing, and debugging AWS SAM–based serverless applications, you can use the [AWS Cloud9 IDE](https://docs.aws.amazon.com/cloud9/latest/user-guide/). To build a deployment pipeline for your serverless applications, you can use [CodeBuild](https://docs.aws.amazon.com/codebuild/latest/userguide/), [CodeDeploy](https://docs.aws.amazon.com/codedeploy/latest/userguide/), and [CodePipeline](https://docs.aws.amazon.com/codepipeline/latest/userguide/). You can also use [AWS CodeStar](https://docs.aws.amazon.com/codestar/latest/userguide/) to get started with a project structure, code repository, and a CI/CD pipeline that's automatically configured for you. To deploy your serverless application, you can use the [Jenkins plugin](https://plugins.jenkins.io/aws-sam/). You can use the [Stackery.io toolkit](https://www.stackery.io/product/aws-sam/) to build production-ready applications.

***Q:6—* What command do you run with SAM to initialise any project with SAM template? Mention any two disadvantages of SAM.**

**Ans:-**

**Sam init**

* Initializes a serverless application with an AWS SAM template. The template provides a folder structure for your AWS Lambda functions, and is connected to event sources such as APIs, Amazon Simple Storage Service (Amazon S3) buckets, or Amazon DynamoDB tables. This application includes everything that you need to get started and to eventually extend it into a production-scale application.
* With AWS SAM, you can initialize your application using one of two modes:

1) interactive Workflow

2) Providing All Required Parameters

**Interactive Workflow**

Through the interactive initialize workflow, you can input

1) Your project name, preferred runtime, and template file

2) The location of a custom template

**Providing Parameters**:

* Provide all required parameters.
* If you provide a subset of required parameters, you are prompted for the additional required information.

**Disadvantage of SAM**

1. The API Gateway configuration gets extensively based on the Swagger. It is irrelevant to the users since they don't need to get stuck with the Swagger.
2. The event source and the feature set might get limited since it was just released.
3. It is a serverless web application that is still new and might have some bugs that the current users should get worried about.
4. The users will no longer manage to reference event sources resources since they are the properties functions.

***Q:7—* What are the steps to integrate AWS SAM in a Spring project? Is it possible to interact with all the serverless services using console? [NENT]**

**Ans:-**

1. Step 1: Download a sample **AWS SAM** application.
2. Step 2: Build your application.
3. Step 3: **Deploy** your application to the **AWS** Cloud.
4. Step 4: (Optional) Test your application locally.
5. Troubleshooting.
6. Clean up.
7. Conclusion.

Simplify how you build, deploy, deliver, and share serverless applications on AWS .To get started with building SAM-based applications, use the AWS SAM CLI. SAM and deploy all related resources together as a single, versioned entity. SAM integrates with a suite of AWS serverless tools. Sign In to the Console .

***Q:8—* What does the following command do: sam local start-lambda? [NET]**

**ANS:-**

**sam local start-lambda**

* Enables you to programmatically invoke your Lambda function locally by using the AWS CLI or SDKs. This command starts a local endpoint that emulates AWS Lambda.
* By default when you use this command, the AWS SAM CLI assumes that your current working directory is your project's root directory. The AWS SAM CLI first tries to locate a template file built using the [sam build](https://docs.aws.amazon.com/serverless-application-model/latest/developerguide/sam-cli-command-reference-sam-build.html) command, located in the .aws-sam subfolder, and named template.yaml or template.yml. Next, the AWS SAM CLI tries to locate a template file named template.yaml or template.yml in the current working directory. If you specify the --template option, AWS SAM CLI's default behavior is overridden, and will load just that AWS SAM template and the local resources it points to.
* You can run your automated tests against this local Lambda endpoint. When you send an invoke to this endpoint using the AWS CLI or SDK, it locally executes the Lambda function that's specified in the request.

**Example**

*# SETUP*

*# ------*

*# Start the local Lambda endpoint by running this command in the directory that contains your AWS SAM template.*

sam local start-lambda

*# USING AWS CLI*

*# -------------*

*# Then, you can invoke your Lambda function locally using the AWS CLI*