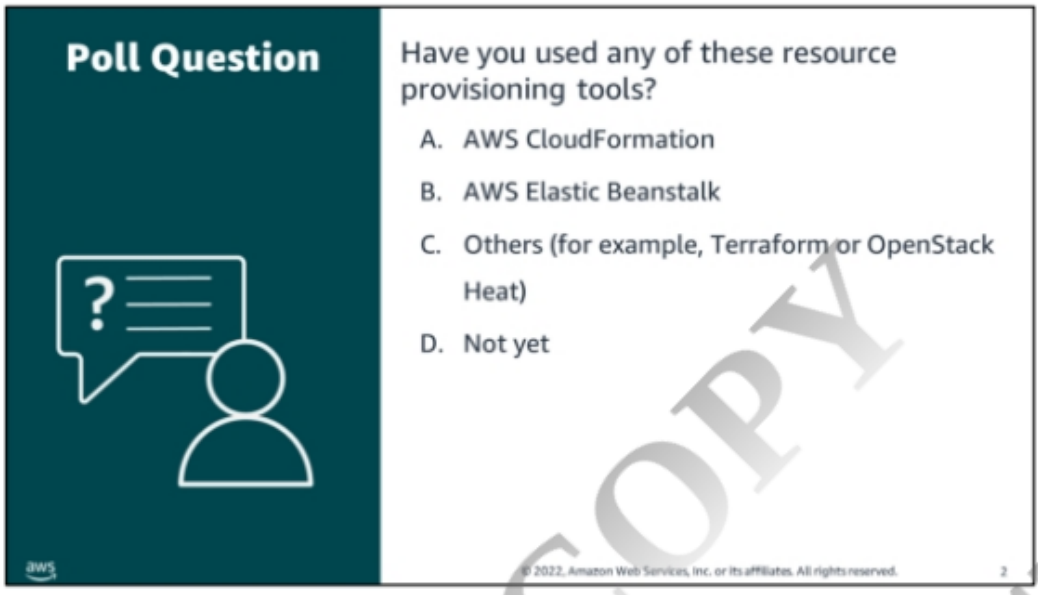
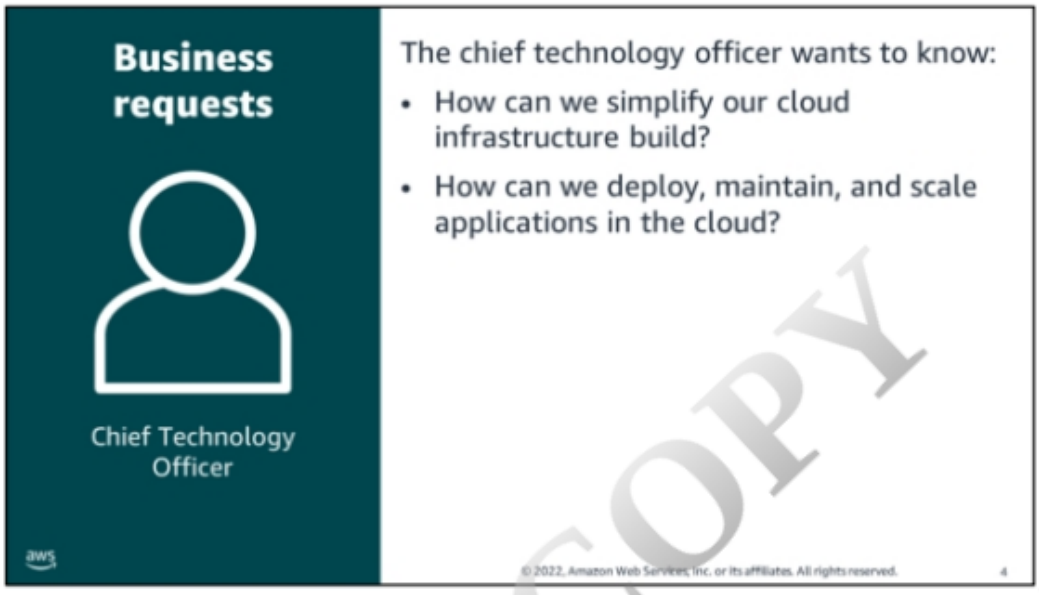
Printed by: 2d35e8483186bd2@placeholder.44518.edu. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.



Module overview

- Business requests
- AWS CloudFormation
- Infrastructure management
- Present solutions
- Knowledge check

 3

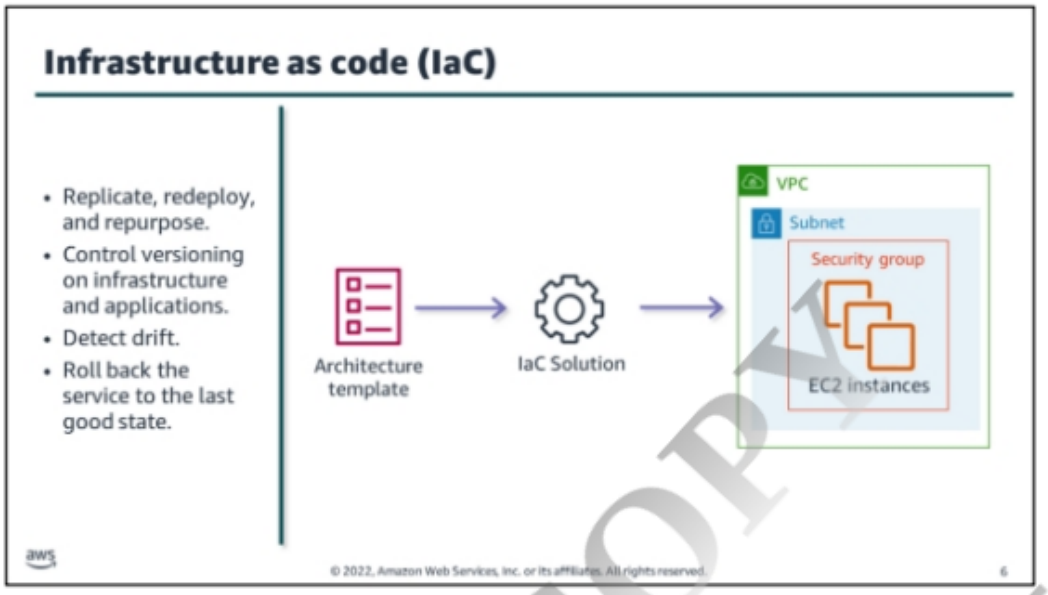


Imagine your chief technology officer meets with you to discuss how to automate deployments and operations in the cloud. Here are some questions they are asking.

At the end of this module, you meet with the chief technology officer and present some solutions.

The chief technology officer asks, "How can we simplify our cloud infrastructure build?"

The infrastructure team would like to create environments that can be easily deployed, updated, and taken down. The company wants you to identify tools that automate infrastructure deployment.



You can simplify the deployment of your AWS resources using infrastructure as code (IaC). With IaC, you use code to define, deploy, configure, update, and remove infrastructure.

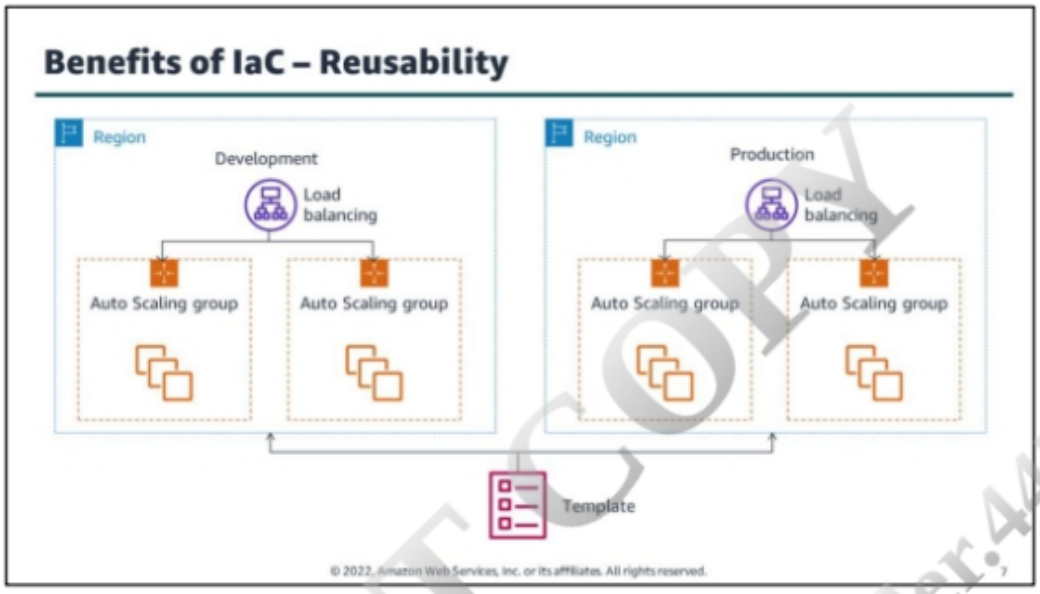
A *template* is a text file that describes and defines the resources to be deployed in your environment. This template is then processed by an engine that provisions the specified resources.

- Define an entire application stack (all resources required for your application) in a JSON or YAML template file. Treat the template as code and manage it using a version control system.
- Define runtime parameters for a template, such as the Amazon Elastic Compute Cloud (Amazon EC2) instance size and Amazon EC2 key pair.
- The IaC solution provisions the resources defined in the template.

IaC has the following benefits:

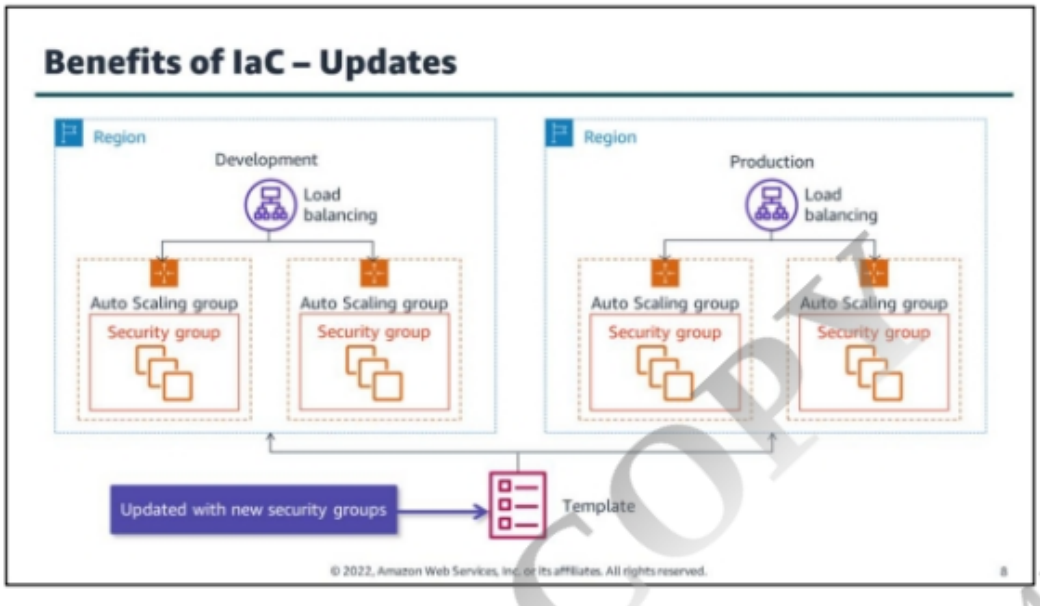
- **Speed and safety** – Your infrastructure is built programmatically, which makes it faster than manual deployment and makes errors less likely.
- **Reusability** – You can organize your infrastructure into reusable modules.
- **Documentation and version control** – Your templates document your deployed resources, and version control provides a history of your infrastructure over time. You can also roll back to a previous working version of your infrastructure in the event of error.
- **Validation** – You perform code review on your templates, which decreases the chances of errors.

To learn more about creating a continuous integration and delivery (CI/CD) pipeline in the AWS Cloud, see "Complete CI/CD with AWS CodeCommit, AWS CodeBuild, AWS CodeDeploy, and AWS CodePipeline" in the *AWS DevOps Blog* (https://aws.amazon.com/blogs/devops/complete-ci-cd-with-aws-codecommit-aws-codebuild-aws-codedeploy-and-aws-codepipeline/).



If you build infrastructure with code, you gain the benefits of repeatability and reusability while building your environments. Build the same complex environments with one template, or a combination of templates. This example uses the architecture template to create identical resources in different AWS Regions. One is the development environment and the other is the production environment.

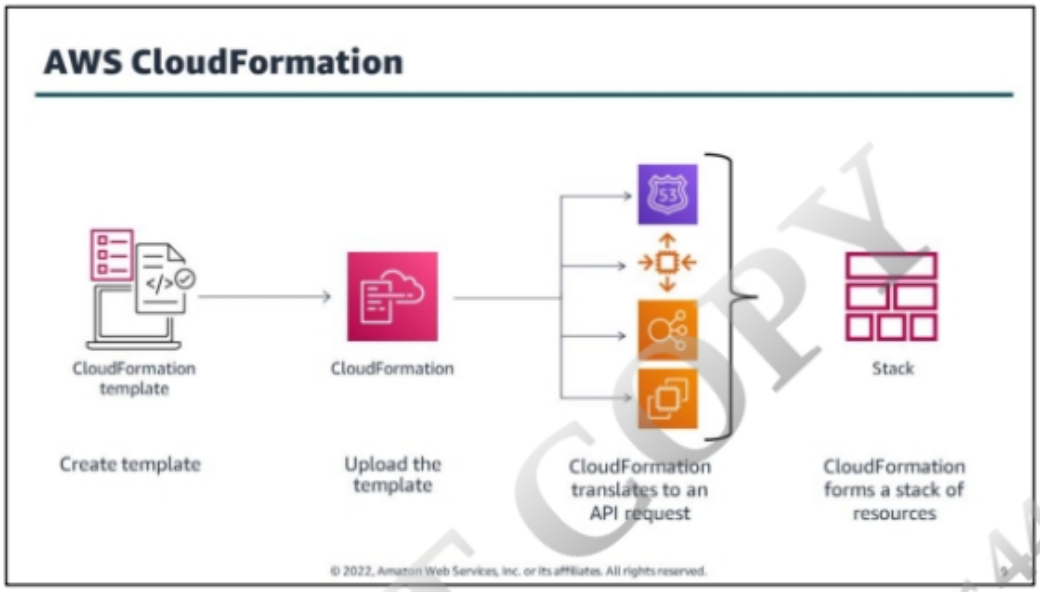
For builds to match a specific context, create environments dependent on conditions. For example, a template can be designed so that different Amazon Machine Images (AMIs) are used in the development or the production environments.



In this scenario, the template has been updated to add new security groups to the instance stacks.

With one change to the template used to launch these environments, all environments add the new security group resource.

This feature makes resource maintenance easier, provides consistency, and reduces effort through parallelization.

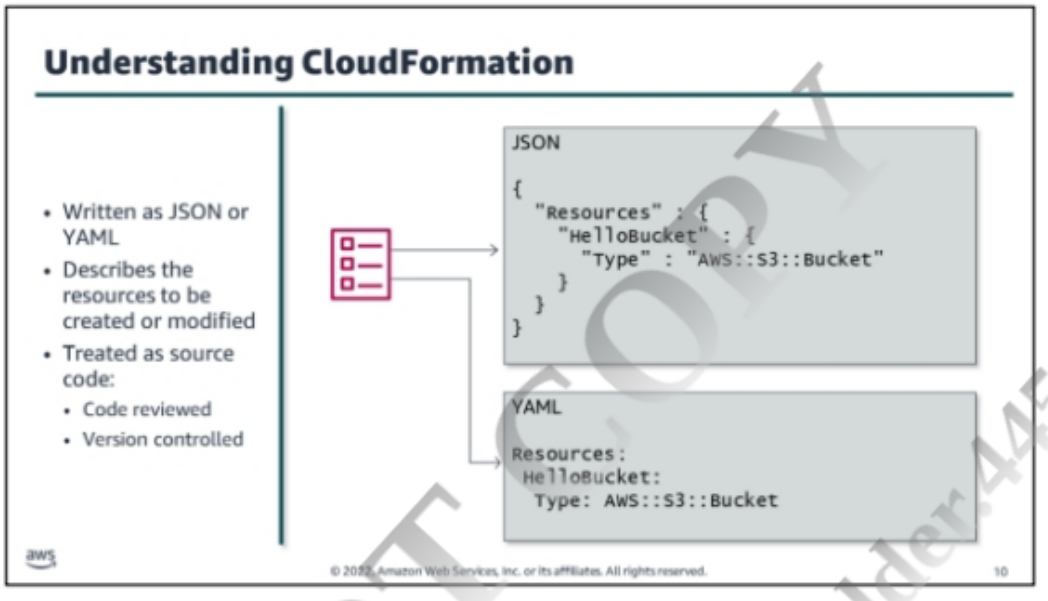


Essentially, CloudFormation is an API wrapper. When you create an EC2 instance in the AWS Management Console, you initiate an API call to the Amazon EC2 service. The information you enter through the wizard is passed on as parameters.

CloudFormation uses those APIs. The resources you define in your CloudFormation template become API calls to AWS services, just like in the AWS Management Console. CloudFormation manages the dependencies and relationships.

Author your CloudFormation template with any code editor, check it in to a version control system such as GitHub or CodeCommit, and review files before deploying.

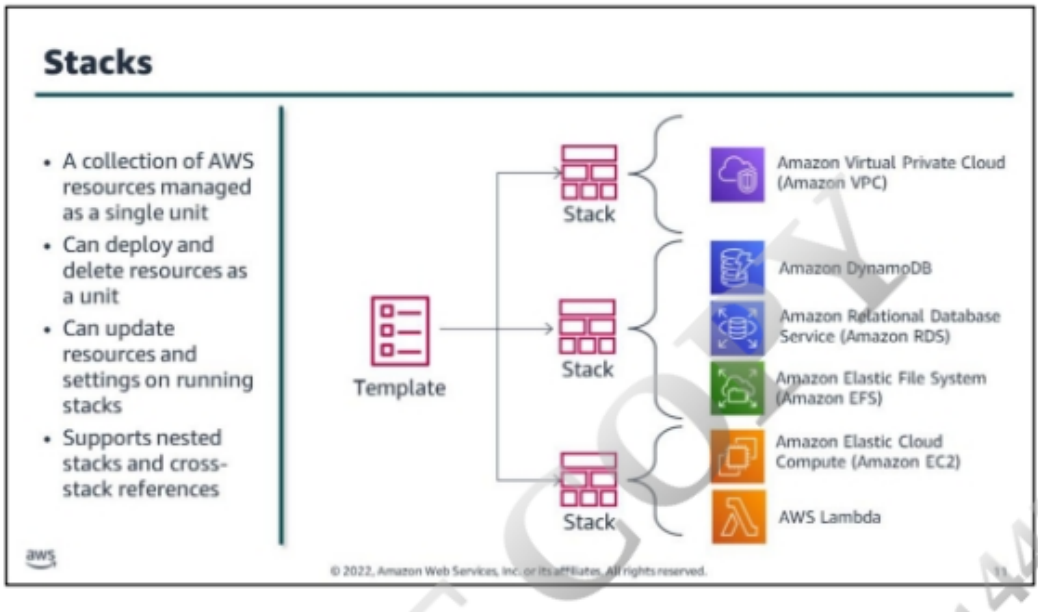
CloudFormation is available in all AWS Regions, and you pay for only the resources you use.



Additional points for a CloudFormation template:

- You can manage it using your choice of version control—for example, Git or Subversion (SVN).
- Define an entire application stack (all resources required for your application) in a JSON template file.
- Define runtime parameters for a template (EC2 instance size, Amazon EC2 key pair, and so on).
- If you created an AWS resource outside CloudFormation management, you can bring this existing resource into CloudFormation management using *resource import*.

YAML-formatted CloudFormation templates follow the same anatomy as existing JSON-formatted templates and support all the same features.



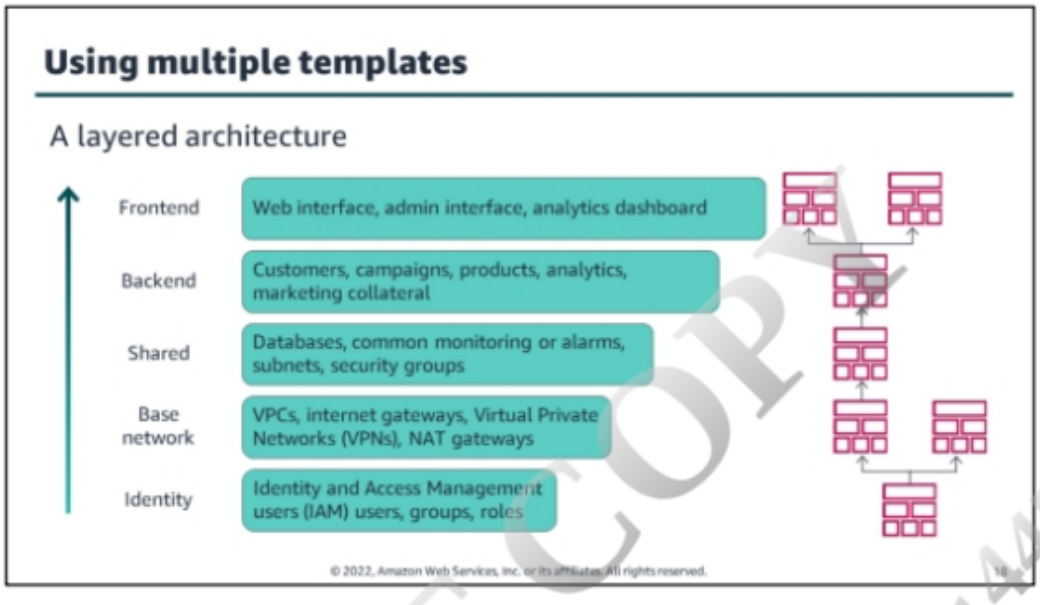
All resources in a stack are defined by the stack's CloudFormation template. You can manage a collection of resources by creating, updating, or deleting stacks. For example, a stack can include all resources required to run a web application, including a web server, database, and networking rules. If you no longer require that web application, you can delete the stack, which deletes all of its related resources.

CloudFormation treats the stack resources as a single unit. They must all be created or deleted successfully for the stack to be created or deleted. If a resource can't be created, CloudFormation rolls the stack back and deletes any resources that were created. If a resource can't be deleted, CloudFormation retains any remaining resources until the entire stack can be successfully deleted.

When you need to make changes to a stack's settings or change its resources, you update the stack instead of deleting it and creating a new stack. To change a running stack, submit the changes that you want to make by providing a modified template, new input parameter values, or both. CloudFormation generates a change set by comparing your stack with the changes you submitted.

You can create cross-stack references so that you can share outputs from one stack with another stack.

For more information about cross-stack references, see "Walkthrough: Refer to resource outputs in another AWS CloudFormation stack" in the *AWS CloudFormation User Guide* (https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/walkthrough-crossstackref.html).

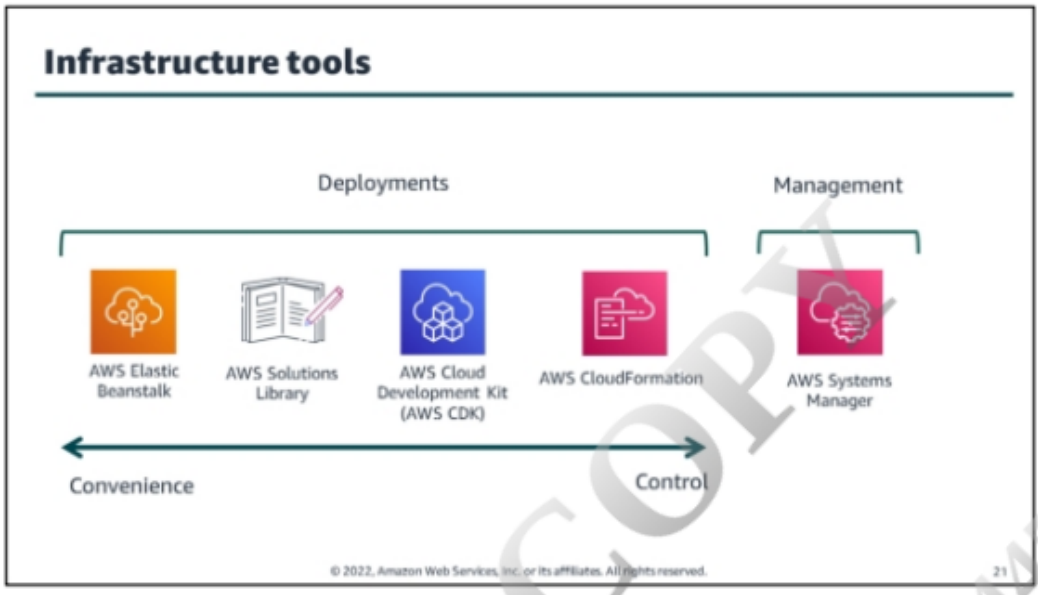


A layered architecture organizes stacks into multiple horizontal layers that build on top of one another. Each layer has a dependency on the layer directly under it. You can have one or more stacks in each layer, but within each layer, your stacks should have AWS resources with similar lifecycles and ownership.

For more information about a layered architecture, see "AWS CloudFormation best practices" in the *AWS CloudFormation User Guide* (https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/best-practices.html).

The chief technology officer asks, "How can we deploy, maintain, and scale applications in the cloud?"

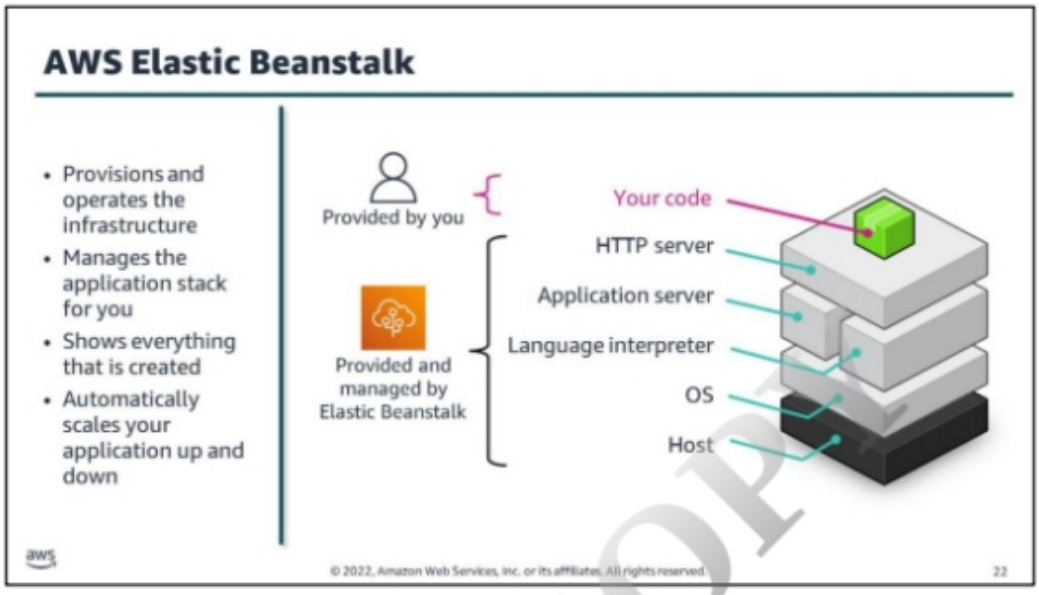
The operations team is looking for tools that can automate the deployment of infrastructure and help them manage resources once deployed.



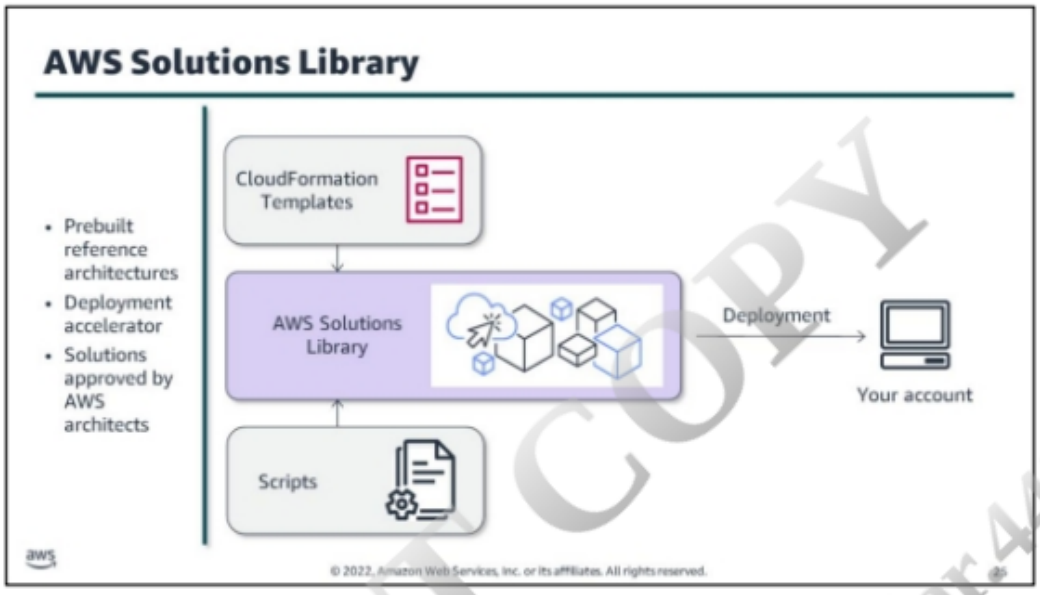
When choosing infrastructure deployment tools, you need to find a balance between convenience and control. Some tools give you complete control and have you choose every component and configuration. Though you can customize your deployment to fit your business needs, this requires greater expertise and more resources to manage and maintain. Other tools are designed for convenience and include preconfigured infrastructure templates for common solutions. Though these tools are easier to use and require less maintenance, you do not always have the ability to customize your infrastructure components. You can automate your infrastructure deployments using the following tools:

- **AWS Elastic Beanstalk** – Elastic Beanstalk integrates with developer tools and provides a one-stop experience to manage the application lifecycle. Elastic Beanstalk provisions and manages application infrastructure to support your application.
- **AWS Solutions Library** – AWS Solutions Library carries solutions built by AWS and AWS Partners for a broad range of industry and technology use cases. These solutions includes the tools you need to get started quickly, such as CloudFormation templates, scripts, and reference architectures.
- **AWS Cloud Development Kit (AWS CDK)** – AWS CDK is an open-source software development framework to model and provision your application resources using common programming languages. AWS CDK simplifies the creation and deployment of CloudFormation templates. It offers infrastructure components and groups of components preconfigured according to best practices. However, you can still customize your components are their settings.
- **AWS CloudFormation** – With CloudFormation, you define every resource and its configuration. You have granular control over every component of your infrastructure.

With **AWS Systems Manager**, you can view and control your infrastructure on AWS. You can automate or schedule a variety of maintenance and deployment tasks.

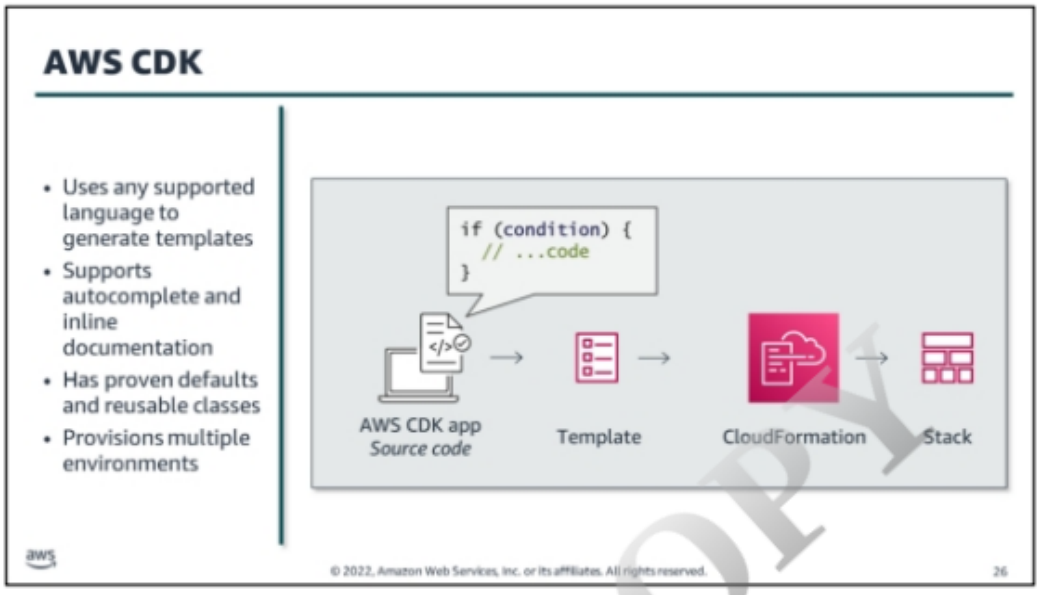


The goal of Elastic Beanstalk is to help developers deploy and maintain scalable web applications and services in the cloud without having to worry about the underlying infrastructure. Elastic Beanstalk configures each EC2 instance in your environment with the components necessary to run applications for the selected application type. You don't have to worry about logging into instances to install and configure your application stack. With Elastic Beanstalk you can provision infrastructure to support common application designs, such as web applications and worker services.



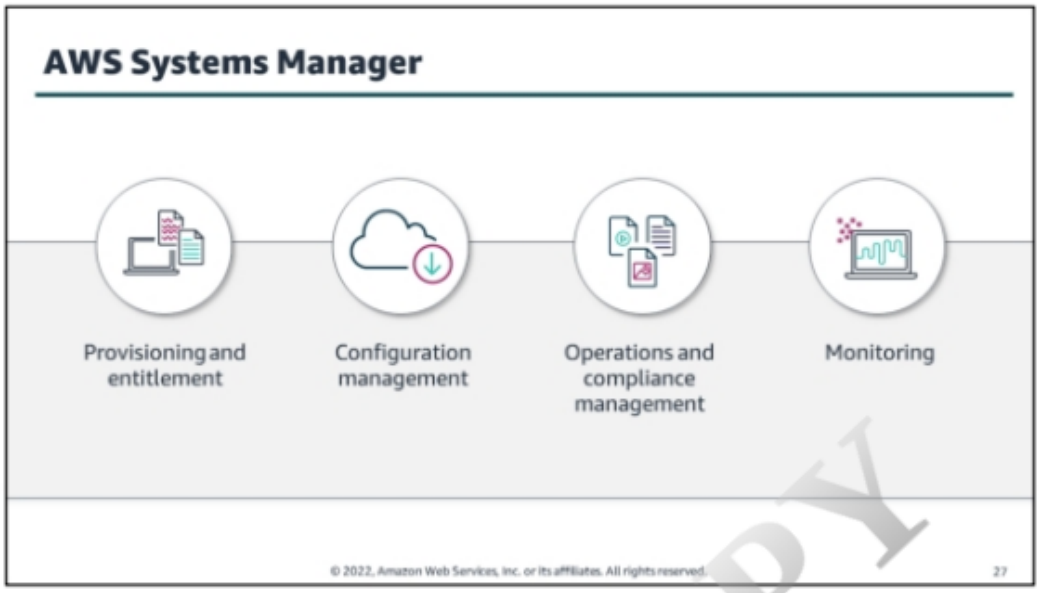
AWS Solutions Library helps you solve common problems and build faster using AWS. Solutions are vetted by AWS architects and are designed to be operationally effective, reliable, secure, and cost efficient. Many AWS solutions come with prebuilt CloudFormation templates. They can also include a detailed architecture, a deployment guide, and instructions for automated and manual deployment. You will be charged for the resources you use to create and run this environment.

For more information, see "AWS Solutions Library" (https://aws.amazon.com/solutions/).



AWS Cloud Development Kit (AWS CDK) is a software development framework that defines your cloud application resources using a declarative model and familiar programming languages. AWS CDK includes a library of customizable constructs, which are building blocks consisting of one or more resources and include common configurations. You can use AWS CDK to generate CloudFormation templates and deploy your infrastructure along with your application runtime assets.

You can use AWS CDK with common programming languages such as Python, JavaScript, TypeScript, Java, or C#.



When designing infrastructure you must plan for its management. This affects your infrastructure deployments since you need to grant the correct security policies to your management tools. You might also need to install management agents on your instances.

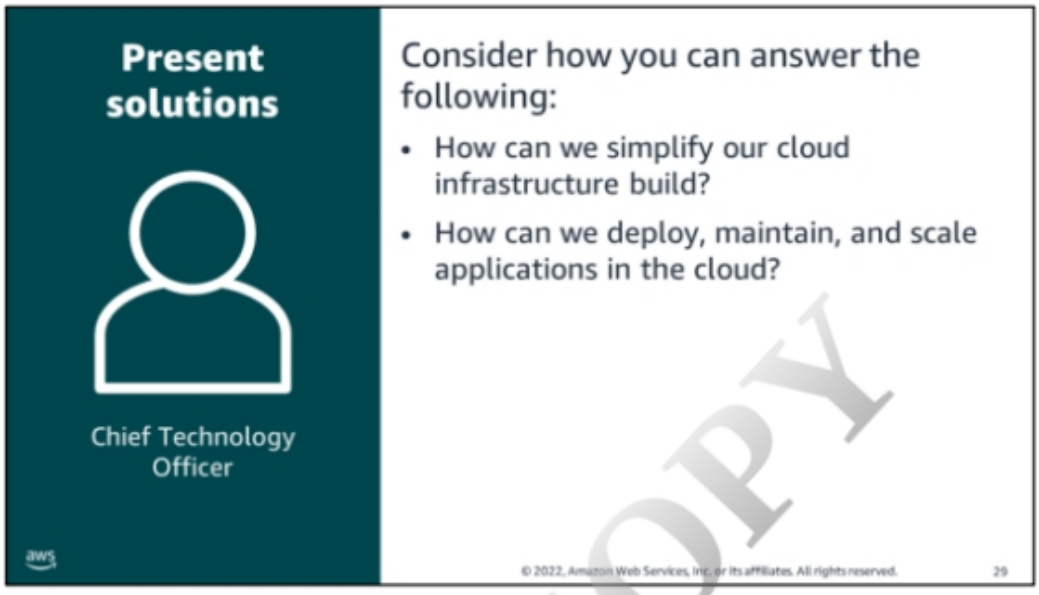
AWS Systems Manager provides a central place to view and manage your AWS resources, so you can have complete visibility and control over your operations.

With Systems Manager, you can:

- Create logical groups of resources such as applications, different layers of an application stack, or development and production environments.
- Select a resource group and view its recent API activity, resource configuration changes, related notifications, operational alerts, software inventory, and patch compliance status.
- Take action on each resource group depending on your operational needs.
- Centralize operational data from multiple AWS services and automate tasks across your AWS resources.

You can open Systems Manager from the Amazon EC2 console. Select the instances you want to manage and define the management tasks you want to perform. Systems Manager is available at no cost to manage your Amazon EC2 and on-premises resources.

For more information about Systems Manager, see "What is AWS Systems Manager?" in the *AWS Systems Manager User Guide* (https://docs.aws.amazon.com/systems-manager/latest/userguide/what-is-systems-manager.html).



Imagine you are now ready to talk to the chief technology officer and present solutions that meet their architectural needs.

Think about how you would answer the questions from the beginning of the lesson.

Your answers should include the following solutions:

- You can simplify your infrastructure with an infrastructure as code (IaC) approach using Amazon CloudFormation.
- You can deploy and maintain your infrastructure using AWS CDK, solutions from the AWS Solutions Library, and AWS Elastic Beanstalk. You can automate infrastructure management using AWS Systems Manager.

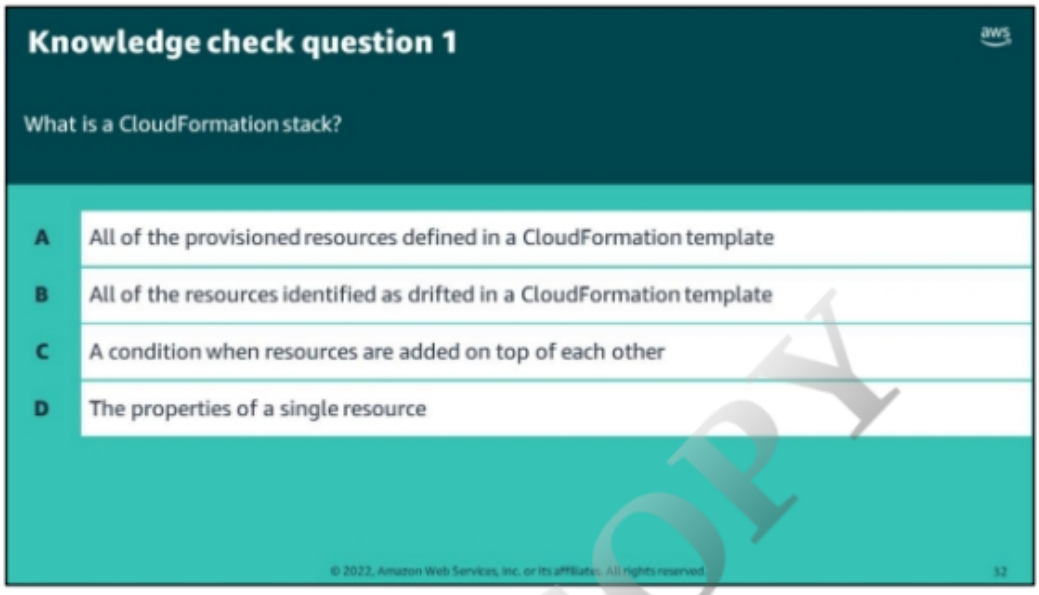
Module review

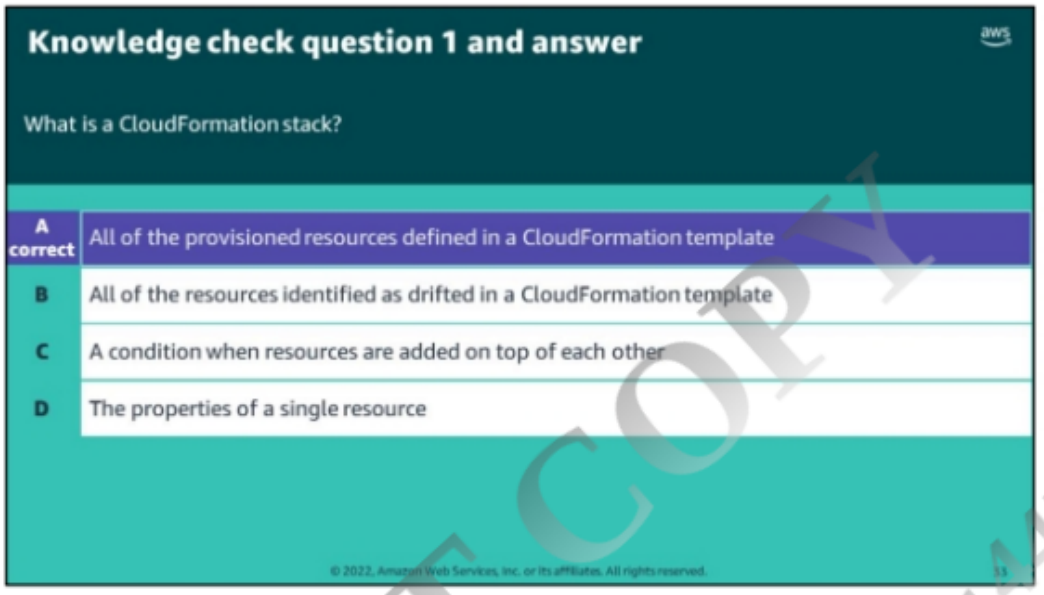
In this module you learned about:

- ✓ Amazon CloudFormation
- ✓ Infrastructure management

Next, you will review:

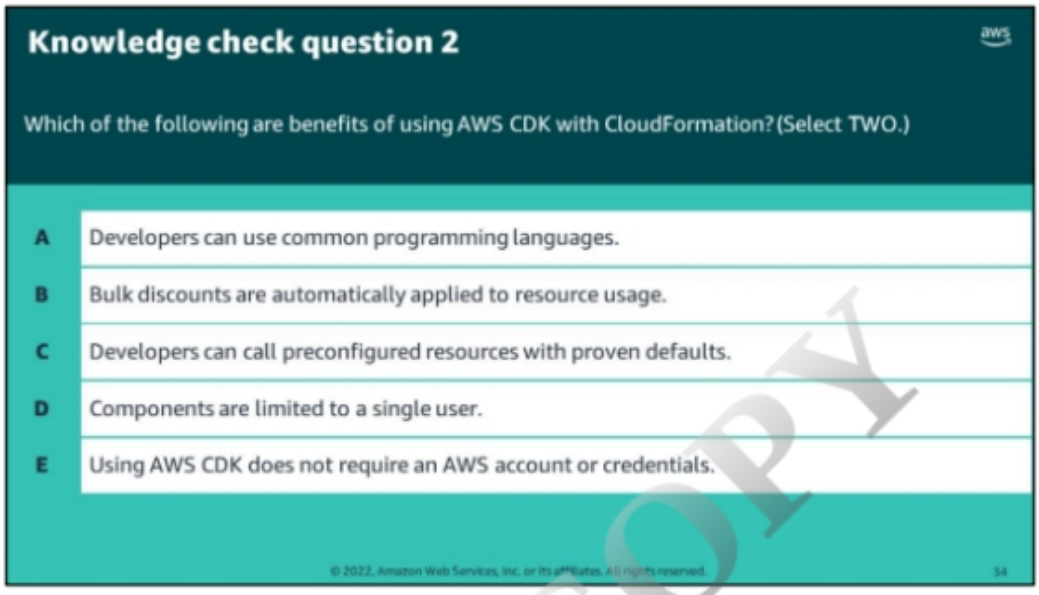
Knowledge check

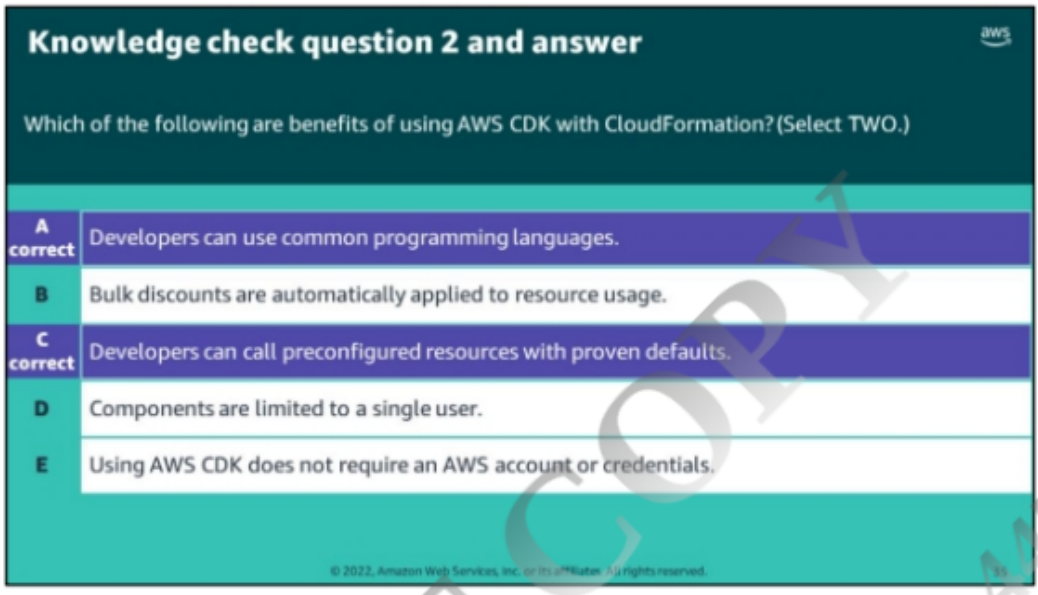




The correct answer is A, all of the provisioned resources defined in a CloudFormation template.

A stack is a collection of AWS resources generated by a CloudFormation template and managed as a single unit.





The correct answers are A and C, developers can use common programming languages and developers can call preconfigured resources with proven defaults.

AWS CDK is a software development framework you can use to define your cloud application resources. It can be used with familiar programming languages such as TypeScript, Python, Java, and .NET. AWS CDK offers different cloud components that include configuration detail, boilerplate, and glue logic for using one or multiple AWS services.

For more information about the features of AWS CDK, see "AWS Cloud Development Kit features" (https://aws.amazon.com/cdk/features/).