4Q.**How would you further automate the management of**

**the infrastructure if given unlimited time and resource?**

## Infrastructure automation is motivated by two factors. To begin, system administrators, information technology professionals, and DevOps engineers must automate as many routine tasks as possible. That is why we create tools.

## Second, automation allows for the management of complex, diverse environments as well as rapid scaling.

## As a result, we employ CloudFormation on AWS: Increase the speed of cloud provisioning with infrastructure as code. Tools enable you to securely store and version your application's source code while also automatically building, testing, and deploying it to AWS or your on-premises environment.

## Infrastructure automation tools will be useful whether you use a public cloud, a private cloud, or a hybrid cloud.

## The use of infrastructure automation in IT operations is critical to the successful delivery of services. This article will go over eight ways infrastructure automation improves IT operations.

## 

1. [Start with existing AWS CloudFormation templates](https://aws.amazon.com/blogs/infrastructure-and-automation/best-practices-automating-deployments-with-aws-cloudformation/#existing-templates)

Starting with an existing template can save you hours or days of work when building an automated deployment on AWS infrastructure. Amazon Elastic Cloud Compute (Amazon EC2) instances, Amazon Simple Storage Service (Amazon S3) buckets, Lambda functions, and Amazon RDS databases are all common resources.

1. [Create modular template](https://aws.amazon.com/blogs/infrastructure-and-automation/best-practices-automating-deployments-with-aws-cloudformation/#modular-templates)

To automate a complex deployment, you could use a single AWS CloudFormation template, but it's easier to work with multiple smaller templates. Consider a three-tier web application with the following components: An Amazon RDS database with high availability, an Amazon EC2 Auto Scaling group for the application layer, and load balancers for the web layer.

1. [Use existing repositories as submodules](https://aws.amazon.com/blogs/infrastructure-and-automation/best-practices-automating-deployments-with-aws-cloudformation/#repo-submodules)

Use submodules to take advantage of previously completed work. Submodules not only save time during development, but they also reduce the need for redundant resources to be maintained.

You can import other GitHub repositories as submodules and deploy them from your parent AWS CloudFormation template if you keep your AWS CloudFormation templates in a GitHub repository. Use the git submodule add command to accomplish this.

1. [Use an integrated development environment with linting](https://aws.amazon.com/blogs/infrastructure-and-automation/best-practices-automating-deployments-with-aws-cloudformation/#ide-linting)

When developing for AWS CloudFormation, using an integrated development environment (IDE) can help. An IDE can detect errors in real time and reformat multiple lines of code at the same time. Most IDEs support third-party linting tools to ensure that your code is properly developed.

1. [Use parameters to define paths to your external assets](https://aws.amazon.com/blogs/infrastructure-and-automation/best-practices-automating-deployments-with-aws-cloudformation/#param-assets)

Avoid hardcoded paths to external assets in your AWS CloudFormation templates to ensure their future viability. External assets called by the template can be stored in the same repository as the template, reducing external dependencies. AWS CodeCommit, GitHub, and GitLab are examples of popular repositories.

1. [Use the same names for common parameters](https://aws.amazon.com/blogs/infrastructure-and-automation/best-practices-automating-deployments-with-aws-cloudformation/#same-names)

Avoid hardcoded paths to external assets in your AWS CloudFormation templates to ensure their future viability. External assets called by the template can be stored in the same repository as the template, reducing external dependencies. AWS CodeCommit, GitHub, and GitLab are examples of popular repositories.

Using the parameter name VPCID in all your templates simplifies keeping track of your outputs and parameter values.

Here are some other common parameter names to keep the same used in the project:

VPCCIDR – Classless Inter-Domain Routing (CIDR) block of the VPC into which the Quick Start is deployed

PrivateSubnet1ID – ID of the first Availability Zone’s private subnet

PrivateSubnet2ID – ID of the second Availability Zone’s private subnet

PublicSubnet1ID – ID of the first Availability Zone’s public subnet

PublicSubnet2ID – ID of the second Availability Zone’s public subnet

PublicSubnet3ID – ID of the second Availability Zone’s public subnet

PublicSubnet4ID – ID of the second Availability Zone’s public subnet

1. [Automate AWS CloudFormation testing with TaskCat](https://aws.amazon.com/blogs/infrastructure-and-automation/best-practices-automating-deployments-with-aws-cloudformation/#taskcat)

When you build AWS CloudFormation templates, you must test them. Testing typically involves the following steps:

Upload your templates to an S3 bucket.

Sign in to the AWS Management Console.

Open the AWS CloudFormation console.

Enter the S3 path to your parent template.

Manually enter parameter values.

Launch the stack.

Check if the deployment succeeds or fails.

It's time-consuming to manually deploy these templates across multiple AWS regions. An open-source tool called TaskCat automates this process for you.

1. [Maintain your templates](https://aws.amazon.com/blogs/infrastructure-and-automation/best-practices-automating-deployments-with-aws-cloudformation/#maintain-templates)

Test your AWS CloudFormation templates on a regular basis to ensure that they continue to function properly. As new service functionality becomes available, incorporate it. EC2 instance types, for example, may be deprecated, and Amazon Machine Images (AMIs) may be removed from general availability.