



Joe Baron, Hisham Baz, Tim Bixler, Biff Gaut,  
Kevin E. Kelly, Sean Senior, John Stamper

# AWS Certified Solutions Architect

## OFFICIAL STUDY GUIDE

**ASSOCIATE EXAM**

Covers exam objectives, including designing highly available, cost efficient, fault tolerant, scalable systems, implementation and deployment, data security, troubleshooting, and much more...

Includes interactive online learning environment and study tools with:

- + 2 custom practice exams
- + More than 100 electronic flashcards
- + Searchable key term glossary



columnar storage technology that improves I/O efficiency and parallelizing queries across multiple nodes, Amazon Redshift is able to deliver fast query performance. The Amazon Redshift architecture allows organizations to automate most of the common administrative tasks associated with provisioning, configuring, and monitoring a cloud data warehouse.

## Amazon ElastiCache

*Amazon ElastiCache* is a web service that simplifies deployment, operation, and scaling of an in-memory cache in the cloud. The service improves the performance of web applications by allowing organizations to retrieve information from fast, managed, in-memory caches, instead of relying entirely on slower, disk-based databases. As of this writing, Amazon ElastiCache supports Memcached and Redis cache engines.

## Management Tools

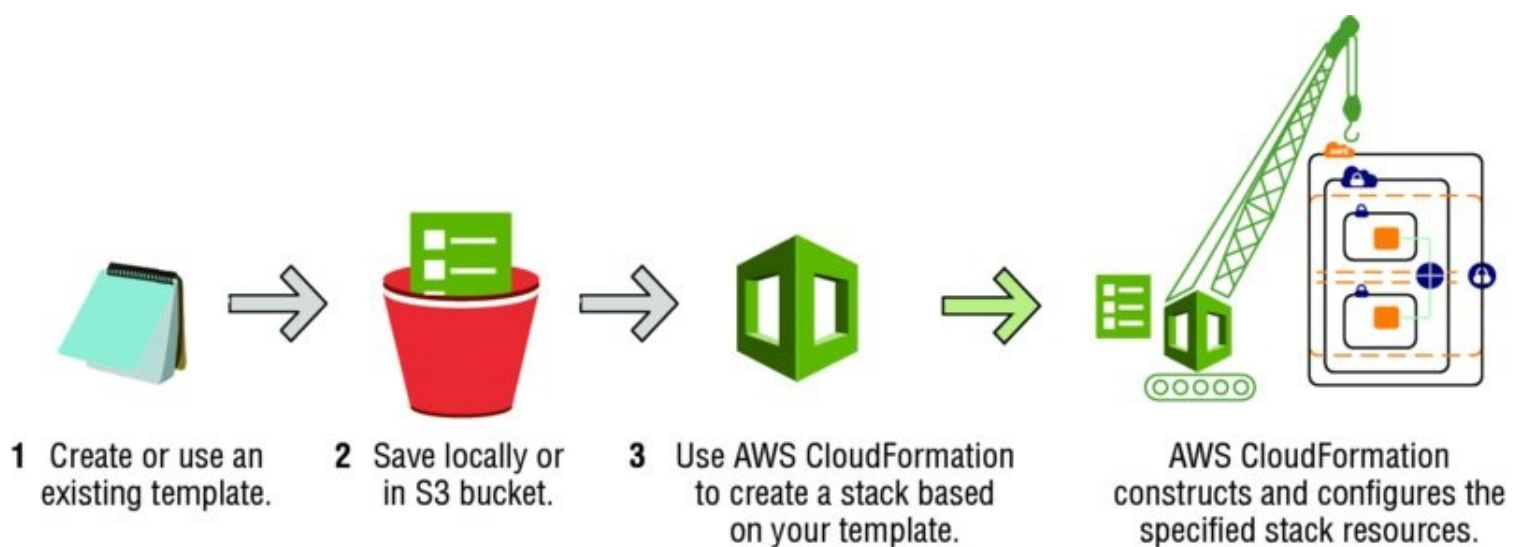
AWS provides a variety of tools that help organizations manage your AWS resources. This section provides an overview of the management tools that AWS provides to organizations.

## Amazon CloudWatch

*Amazon CloudWatch* is a monitoring service for AWS Cloud resources and the applications running on AWS. It allows organizations to collect and track metrics, collect and monitor log files, and set alarms. By leveraging Amazon CloudWatch, organizations can gain system-wide visibility into resource utilization, application performance, and operational health. By using these insights, organizations can react, as necessary, to keep applications running smoothly.

## AWS CloudFormation

*AWS CloudFormation* gives developers and systems administrators an effective way to create and manage a collection of related AWS resources, provisioning and updating them in an orderly and predictable fashion. AWS CloudFormation defines a JSON-based templating language that can be used to describe all the AWS resources that are necessary for a workload. Templates can be submitted to AWS CloudFormation and the service will take care of provisioning and configuring those resources in appropriate order (see [Figure 1.4](#)).



**FIGURE 1.4** AWS CloudFormation workflow summary

deploying applications, and running scripts. One of the key AWS OpsWorks features is a set of lifecycle events that automatically run a specified set of recipes at the appropriate time on each instance.

An instance represents a single computing resource, such as an Amazon EC2 instance. It defines the resource's basic configuration, such as operating system and size. Other configuration settings, such as Elastic IP addresses or Amazon EBS volumes, are defined by the instance's layers. The layer's recipes complete the configuration by performing tasks, such as installing and configuring packages and deploying applications.

You store applications and related files in a repository, such as an Amazon S3 bucket or Git repo. Each application is represented by an *app*, which specifies the application type and contains the information that is needed to deploy the application from the repository to your instances, such as the repository URL and password. When you deploy an app, AWS OpsWorks triggers a Deploy event, which runs the Deploy recipes on the stack's instances.



Using the concepts of stacks, layers, and apps, you can model and visualize your application and resources in an organized fashion.

Finally, AWS OpsWorks sends all of your resource metrics to Amazon CloudWatch, making it easy to view graphs and set alarms to help you troubleshoot and take automated action based on the state of your resources. AWS OpsWorks provides many custom metrics, such as CPU idle, memory total, average load for one minute, and more. Each instance in the stack has detailed monitoring to provide insights into your workload.

## Use Cases

AWS OpsWorks supports many DevOps efforts, including, but not limited to:

**Host Multi-Tier Web Applications** AWS OpsWorks lets you model and visualize your application with layers that define how to configure a set of resources that are managed together. Because AWS OpsWorks uses the Chef framework, you can bring your own recipes or leverage hundreds of community-built configurations.

**Support Continuous Integration** AWS OpsWorks supports DevOps principles, such as continuous integration. Everything in your environment can be automated.

## AWS CloudFormation

*AWS CloudFormation* is a service that helps you model and set up your AWS resources so that you can spend less time managing those resources and more time focusing on your applications that run in AWS. AWS CloudFormation allows organizations to deploy, modify, and update resources in a controlled and predictable way, in effect applying version control to AWS infrastructure the same way one would do with software.

## Overview

AWS CloudFormation gives developers and systems administrators an easy way to create and manage a collection of related AWS resources, provisioning and updating them in an orderly

and predictable fashion. When you use AWS CloudFormation, you work with *templates* and *stacks*.

You create AWS CloudFormation templates to define your AWS resources and their properties. A *template* is a text file whose format complies with the JSON standard. AWS CloudFormation uses these templates as blueprints for building your AWS resources.



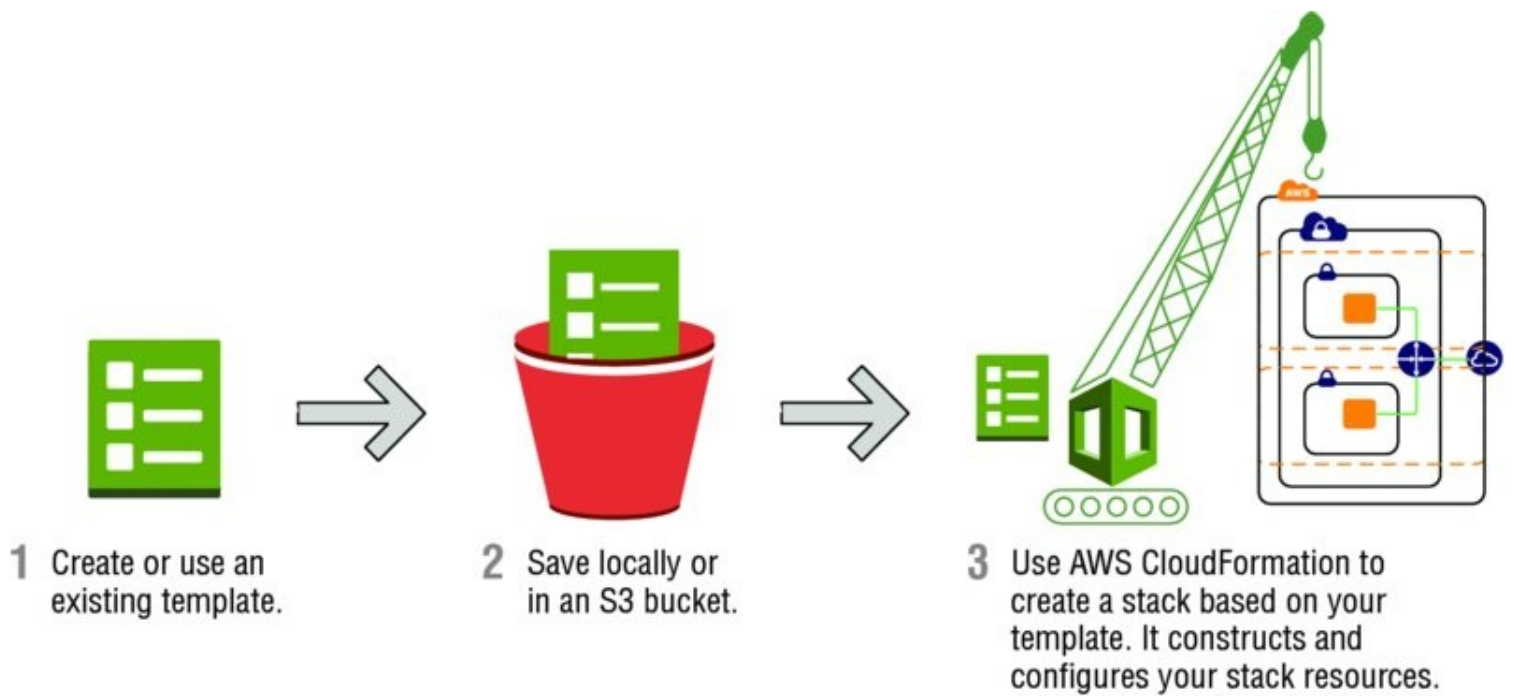
When you use AWS CloudFormation, you can reuse your template to set up your resources consistently and repeatedly. Just describe your resources once, and then provision the same resources over and over in multiple regions.

When you use AWS CloudFormation, you manage related resources as a single unit called a stack. You create, update, and delete a collection of resources by creating, updating, and deleting stacks. All of the resources in a stack are defined by the stack's AWS CloudFormation template. Suppose you created a template that includes an Auto Scaling group, Elastic Load Balancing load balancer, and an Amazon RDS database instance. To create those resources, you create a stack by submitting your template that defines those resources, and AWS CloudFormation handles all of the provisioning for you. After all of the resources have been created, AWS CloudFormation reports that your stack has been created. You can then start using the resources in your stack. If stack creation fails, AWS CloudFormation rolls back your changes by deleting the resources that it created.

Often you will need to launch stacks from the same template, but with minor variations, such as within a different Amazon VPC or using AMIs from a different region. These variations can be addressed using parameters. You can use parameters to customize aspects of your template at runtime, when the stack is built. For example, you can pass the Amazon RDS database size, Amazon EC2 instance types, database, and web server port numbers to AWS CloudFormation when you create a stack. By leveraging template parameters, you can use a single template for many infrastructure deployments with different configuration values. For example, your Amazon EC2 instance types, Amazon CloudWatch alarm thresholds, and Amazon RDS read-replica settings may differ among AWS regions if you receive more customer traffic in the United States than in Europe. You can use template parameters to tune the settings and thresholds in each region separately and still be sure that the application is deployed consistently across the regions.

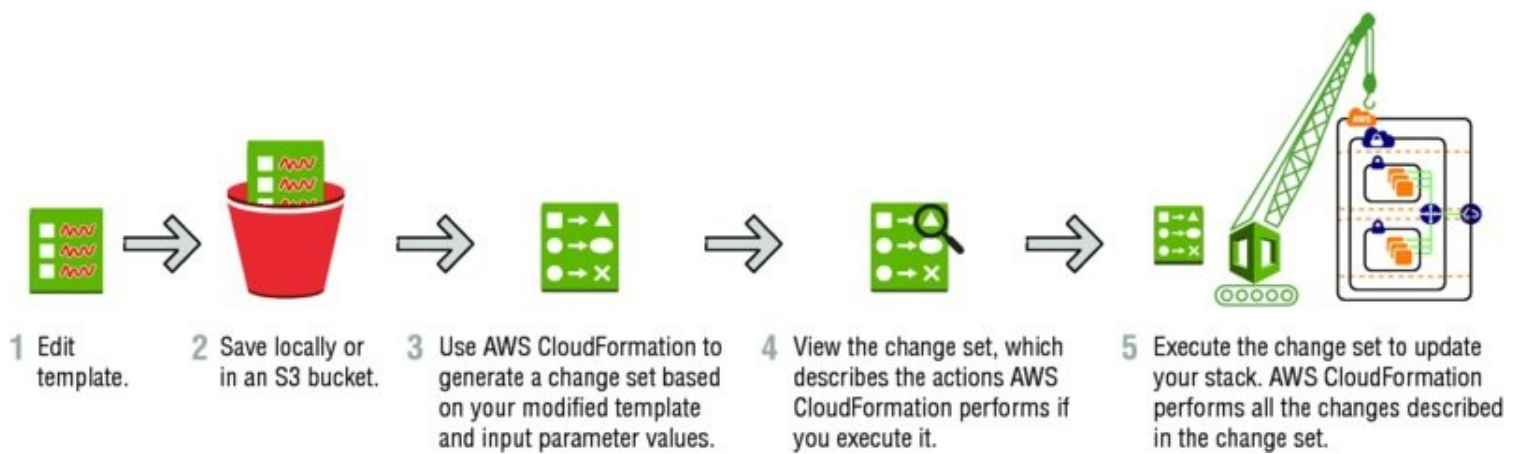
[Figure 11.8](#) depicts the AWS CloudFormation workflow for creating stacks.






**FIGURE 11.8** Creating a stack workflow

Because environments are dynamic in nature, you inevitably will need to update your stack’s resources from time to time. There is no need to create a new stack and delete the old one; you can simply modify the existing stack’s template. To update a stack, create a *change set* by submitting a modified version of the original stack template, different input parameter values, or both. AWS CloudFormation compares the modified template with the original template and generates a change set. The change set lists the proposed changes. After reviewing the changes, you can execute the change set to update your stack. [Figure 11.9](#) depicts the workflow for updating a stack.



**FIGURE 11.9** Updating a stack workflow

When the time comes and you need to delete a stack, AWS CloudFormation deletes the stack and all of the resources in that stack.



**NOTE** If you want to delete a stack but still retain some resources in that stack, you can use a deletion policy to retain those resources. If a resource has no deletion policy, AWS CloudFormation deletes the resource by default.

After all of the resources have been deleted, AWS CloudFormation signals that your stack has been successfully deleted. If AWS CloudFormation cannot delete a resource, the stack will not be deleted. Any resources that haven't been deleted will remain until you can successfully delete the stack.

## Use Case

By allowing you to replicate your entire infrastructure stack easily and quickly, AWS CloudFormation enables a variety of use cases, including, but not limited to:

**Quickly Launch New Test Environments** AWS CloudFormation lets testing teams quickly create a clean environment to run tests without disturbing ongoing efforts in other environments.

**Reliably Replicate Configuration Between Environments** Because AWS CloudFormation scripts the entire environment, human error is eliminated when creating new stacks.

**Launch Applications in New AWS Regions** A single script can be used across multiple regions to launch stacks reliably in different markets.

## AWS Elastic Beanstalk

*AWS Elastic Beanstalk* is the fastest and simplest way to get an application up and running on AWS. Developers can simply upload their application code, and the service automatically handles all of the details, such as resource provisioning, load balancing, Auto Scaling, and monitoring.

## Overview

AWS comprises dozens of building block services, each of which exposes an area of functionality. While the variety of services offers flexibility for how organizations want to manage their AWS infrastructure, it can be challenging to figure out which services to use and how to provision them. With AWS Elastic Beanstalk, you can quickly deploy and manage applications on the AWS cloud without worrying about the infrastructure that runs those applications. AWS Elastic Beanstalk reduces management complexity without restricting choice or control.

There are key components that comprise AWS Elastic Beanstalk and work together to provide the necessary services to deploy and manage applications easily in the cloud. An *AWS Elastic Beanstalk application* is the logical collection of these AWS Elastic Beanstalk components, which includes environments, versions, and environment configurations. In AWS Elastic Beanstalk, an application is conceptually similar to a folder.

An *application version* refers to a specific, labeled iteration of deployable code for a web application. An application version points to an Amazon S3 object that contains the deployable code. Applications can have many versions and each application version is unique. In a running environment, organizations can deploy any application version they already uploaded to the application, or they can upload and immediately deploy a new application version. Organizations might upload multiple application versions to test differences between one version of their web application and another.