Automating Your Architecture

REASONS TO AUTOMATE · INFRASTRUCTURE AS CODE · CLOUDFORMATION · QUICK STARTS · AMAZON Q DEVELOPER · STUDY & REVIEW WELL-ARCHITECTED

01 MUST KNOW

if you remember five things, remember these

01 **Automate to kill manual risk:** manual processes are **error prone, unreliable, and inadequate for an agile business**, and they fail on four fronts — no **repeatability at scale**, no **version control**, no **audit trail**, and **inconsistent configurations**. Automation reduces manual intervention, delivers reproducible environments, improves productivity, and automates testing and scaling.

02 **IaC = a template of desired state: infrastructure as code** writes a template that provisions and manages cloud resources — **human-readable and machine-consumable**. You declare the *desired state* without scripting every step to reach it, so you can **replicate, redeploy, and repurpose** infrastructure. The key benefits are **reusability, repeatability, and maintainability**.

03 **CloudFormation builds stacks: AWS CloudFormation** models, creates, and manages a collection of AWS resources called a **stack**. There is **no extra charge** — you pay only for the resources you create. Create, update, and delete stacks for orderly, predictable, version-controlled provisioning; one stack can span **multiple services in a single Region**.

04 **Only Resources is required:** a template is authored in **JSON or YAML** and has exactly one required section — `Resources`. Everything else is optional: `Parameters` (runtime values), `Mappings` (lookup tables), `Conditions` (whether resources are created — two AZs in prod vs. one in dev), `Transform` (AWS SAM), and `Outputs` (returned values).

05 **Manage the stack, not the resources:** the stack is a **handle** to everything it created. Update by editing the template and running **Update Stack**; preview edits with a **change set** before applying; run **drift detection** to find resources changed outside CloudFormation; **delete the stack** to clean up its resources (keep individual ones with a `DeletionPolicy`).

02 KEY TERMS

TERM	DEFINITION
Infrastructure as code (IaC)	Provisioning and managing cloud resources by defining them in a template file that is both human-readable and machine-consumable
AWS CloudFormation	IaC service that models, creates, and manages a collection of AWS resources; no extra charge beyond the resources created
Stack	A collection of AWS resources that you create, update, and delete — and manage — as a single unit
Template	The document (JSON or YAML) that describes the AWS resources CloudFormation should provision; treated as source code
Resources (section)	The only required template section; specifies the AWS resources to create and their properties
Parameters	Optional section holding values you pass to the template at runtime when creating or updating a stack
Conditions	Optional section that controls whether certain resources are created or properties assigned during create/update
Outputs	Optional section describing the values returned when you view a stack's properties (e.g., an instance ID or VPC ID)
Change set	A preview of the changes CloudFormation will make to a stack, reviewed and approved before you run it
Drift detection	An operation that reports whether a stack's actual resources still match the template (IN_SYNC vs. MODIFIED)

TERM	DEFINITION
Nested stack	A stack created as part of another stack; lets you factor common components into reusable, referenced templates
AWS Quick Starts	Automated reference architectures — a CloudFormation template plus a deployment guide — built on AWS best practices

03 MANUAL BUILDS VS. INFRASTRUCTURE AS CODE

the anti-pattern and the fix

MANUAL PROCESS (ANTI-PATTERN)

console-by-console, every step by hand

- Slow builds; skilled staff tied up on config
- No repeatability at scale across Regions
- No version control, so no rollback
- No audit trail of resource-level changes
- Inconsistent configurations across instances

INFRASTRUCTURE AS CODE (AUTOMATION)

one template, provisioned on demand

- Less manual intervention and production access
- Reproducible, standardized environments
- Roll back by redeploying an older version
- Change every stack by editing the template
- Delete the stack to remove its resources

04 ANATOMY OF A CLOUDFORMATION TEMPLATE

only Resources is required; the rest are optional

SECTION	REQUIRED?	PURPOSE
AWSTemplateFormatVersion	Optional	Template format version (not the same as the API or WSDL version)
Description	Optional	Text string describing the template; must follow the format version
Metadata	Optional	Objects that provide additional information about the template
Parameters	Optional	Values passed to the template at runtime (create or update a stack)
Rules	Optional	Validates a parameter or combination of parameters during create/update
Mappings	Optional	Key-to-value lookup table for conditional parameter values
Conditions	Optional	Controls whether resources are created or properties assigned
Transform	Optional	For serverless apps, specifies the AWS SAM version to use
Resources	REQUIRED	The AWS resources to create and their properties — the only required section
Outputs	Optional	Values returned when you view the stack's properties

05 HOW CLOUDFORMATION WORKS

template → stack → managed resources

01 **Define** resources in a template (or use a prebuilt one), in JSON or YAML → 02 **Upload** the template to CloudFormation, or point to one stored in Amazon S3 → 03 Run **create stack** → resources are provisioned across services as a running environment → 04 The **stack keeps control** — later update stack, detect drift, or delete stack

06 AWS IAC SERVICES BUILT ON CLOUDFORMATION

behind the scenes, each one deploys via CloudFormation

SERVICE	WHAT IT IS
AWS Elastic Beanstalk	Fully managed service that launches an AWS environment from your uploaded application code; handles provisioning, load balancing, scaling, and health monitoring
AWS Quick Starts	Automated reference architectures based on CloudFormation templates; open source and modular, so they are customizable
AWS SAM	Extension of CloudFormation with shorthand syntax for common serverless infrastructure; deploys straight to CloudFormation

SERVICE	WHAT IT IS
AWS Amplify	Framework for full-stack web and mobile apps with less coding and less backend knowledge; builds CloudFormation stacks
AWS CDK	Open source framework to model and provision resources through CloudFormation using familiar programming languages

07 EXAM TRAPS

where the knowledge check gets you

- × "CloudFormation costs extra." — There is **no additional charge** for CloudFormation; you pay only for the AWS resources the stack creates.
- × "Resources is optional like the other sections." — `Resources` is the **only required section** of a template; format version, parameters, conditions, and outputs are all optional.
- × "A change set applies the update." — A change set only **previews** the changes; you must review it and then **run** it for CloudFormation to update the stack.
- × "Fix drift by editing the resource in the console." — Editing a resource directly is what **causes** drift. The correct fix is to update the **template** and run **Update Stack**, keeping the model and the deployment in sync.
- × "Deleting a stack leaves the resources behind." — Deleting a stack **deletes the resources it created**; use a `DeletionPolicy` on a resource when you need to preserve or back it up.
- × "YAML and JSON produce different templates." — Both author the **same** template; the trade-off is readability and embedded comments (YAML) vs. cross-system compatibility and easy parsing (JSON).
- × "AWS Quick Starts are just AWS Marketplace AMIs." — Marketplace AMIs are **single-vendor** solutions that run on EC2; Quick Starts are **modular, customizable** reference architectures that might not use EC2 at all.
- × "On the sample exam, Outputs sizes the environment." — `Outputs` only returns values; the section that decides **whether resources are created** (e.g., two AZs vs. one) is `Conditions`.

08 SELF-QUIZ

answers are upside-down below

Q1 What is the only required section of a CloudFormation template?

Q2 Define a CloudFormation stack.

Q3 Does using CloudFormation itself cost extra?

Q4 In which two formats can a CloudFormation template be authored?

Q5 Which template section holds values you pass in at runtime when creating or updating a stack?

Q6 Which CloudFormation feature lets you preview changes before you implement them?

Q7 Which operation reports whether a stack's actual resources still match its template?

Q8 Which attribute preserves or backs up a resource when its stack is deleted or updated?

Q9 Name two AWS IaC services that use CloudFormation behind the scenes.

Q10 Sample exam: one template must build production across two AZs and development in one AZ. Which optional section provides that logic?

ANSWER KEY (rotate the page)

Q1 Resources **Q2** A collection of AWS resources that you create, update, delete, and manage as a single unit **Q3** No — there is no additional charge; you pay only for the resources it creates **Q4** JSON and YAML **Q5** Parameters **Q6** Change sets **Q7** Drift detection **Q8** The DeletionPolicy attribute **Q9** Any two of: Elastic Beanstalk, Quick Starts, AWS SAM, Amplify, AWS CDK **Q10** Conditions