Creating Automated and Repeatable Deployments

CONFIG MANAGEMENT · AMIs & LAUNCH TEMPLATES · INFRASTRUCTURE AS CODE · JSON & YAML · CLOUDFORMATION · TROUBLESHOOTING · CI/CD

STUDY & REVIEW

01 MUST KNOW

if you remember five things, remember these

01 **Infrastructure as code (IaC):** encode the definition of infrastructure resources into versioned configuration files, exactly like application software. This brings **consistency and repeatability** to the *entire* resource lifecycle — provisioning, configuration management, monitoring, governance, and optimization — not just to the initial build.

02 **AWS CloudFormation:** models and provisions AWS resources from a single **JSON or YAML template**; the running instantiation is a **stack**; it creates, updates, and deletes a set of resources as **one unit**, supports most AWS services, and **detects drift**. AWS recommends managing *all* stack resources through CloudFormation.

03 **Template anatomy:** only the **Resources** section is required — **Parameters**, **Mappings**, and **Outputs** are optional. Intrinsic functions wire elements together: `Ref` (parameters/resources), `Fn::FindInMap` (mappings), `Fn::GetAtt` (other attributes), `Fn::Select`, `Fn::Join`. `DependsOn` forces creation order.

04 **Failures roll back by default:** if any resource cannot be created, CloudFormation **rolls back and deletes** everything it made — override with `--on-failure DO_NOTHING`. A **WaitCondition** plus **cfn-signal** tells CloudFormation an instance's bootstrap *actually* finished, because passing status checks is not the same as being ready.

05 **CI/CD: continuous integration** merges code into a central repo and auto-builds and tests it; **continuous delivery** auto-builds, tests, and stages a release but a human approves production; **continuous deployment** deploys to production automatically. **CodePipeline** orchestrates the flow; **CodeBuild** compiles and tests.

02 KEY TERMS

TERM	DEFINITION
Configuration management	Automating, monitoring, and tracking configuration changes so resources launch quickly and consistently, and changes deploy to running instances
Idempotent	A configuration server creates or configures a resource once and only once; manual changes are detected and rolled back to preserve integrity
Amazon Machine Image (AMI)	A reusable base image for launching EC2 instances; anchored to the Region it is created in and must be explicitly copied to other Regions
User data	Scripts that run on instance launch — the simplest approach to configuration, but harder to version and manage at enterprise scale
Launch template	Stores EC2 launch parameters (AMI ID, instance type, subnet, key pair, security group); supports versioning and reduces deployment errors
Infrastructure as code (IaC)	Defining and version-controlling infrastructure in configuration files, just like application code, for repeatable deployments
AWS CloudFormation	Service that models and provisions AWS infrastructure predictably and repeatedly from a template
Template	The specification of the AWS resources to create — written in JSON or YAML; can be instantiated many times
Stack	A collection of AWS resources created from a template; created, updated, and deleted as a single unit
Change set	A preview of how proposed changes would affect running resources before you execute the update
Drift detection	Identifies stack resources whose configuration was changed outside of CloudFormation so you can re-sync them

TERM	DEFINITION
Intrinsic function	Built-in template functions (`Ref`, `Fn::FindInMap`, `Fn::GetAtt`, `Fn::Select`, `Fn::Join`) that reference or compute values inside a template

03 CLOUDFORMATION TEMPLATE ANATOMY

only Resources is required

SECTION	REQUIRED?	PURPOSE
Parameters	Optional	Name-value inputs supplied at create/update time; referenced later with `Ref`
Mappings	Optional	Two-level static lookup tables — most often the latest AMI ID per Region; read with `Fn::FindInMap`
Resources	Required	The AWS assets to create; each has a `Type` and optional `Properties`; `DependsOn` sets order
Init	Optional	`AWS::CloudFormation::Init` metadata the `cfn-init` script uses to install packages, write files, run commands
WaitCondition	Optional	Blocks stack completion until the instance signals success with `cfn-signal` (or the timeout is reached)
Outputs	Optional	Return values from created resources — URLs, names — visible in the console and via CLI/API

04 MANUAL / SCRIPTED CONFIGURATION VS. INFRASTRUCTURE AS CODE

why IaC wins at scale

MANUAL OR AD-HOC SCRIPTS

fast to start, painful to scale

- Manual configuration is slow, error-prone, and drifts toward inconsistency
- Plain scripts are hard to version and manage at the enterprise level
- Changes are undocumented and difficult to reproduce
- Rollback and multi-Region replication are manual work

INFRASTRUCTURE AS CODE

consistent, repeatable, auditable

- Infrastructure defined in versioned JSON or YAML files
- Same template redeployed across dev, test, and prod
- Failed deployments roll back automatically
- Covers the whole lifecycle — provision, configure, monitor, govern, optimize

05 DEPLOYING A STACK WITH CLOUDFORMATION

`01` Author the **template** in JSON or YAML → `02` **Validate** the template (catch syntax and some semantic errors) → `03` Create a **change set** to preview the impact → `04` **Create the stack** via console, CLI, or API → `05` CloudFormation **provisions the resources** as one unit — rolls back on failure → `06` Read the **Outputs** (URLs, names) once the stack completes

06 INFRASTRUCTURE LIFECYCLE → AWS SERVICE

IaC spans every stage

LIFECYCLE STAGE	THEME	AWS SERVICE
Resource provisioning	Repeatable process	AWS CloudFormation
Configuration management	Incremental changes	EC2 Systems Manager · OpsWorks for Chef
Monitoring and performance	Metrics-driven monitoring	Amazon CloudWatch
Compliance and governance	Tracking changes over time	AWS Config
Resource optimization	Data-driven decisions	AWS Trusted Advisor

07 EXAM TRAPS

where the knowledge check gets you

- × “An AMI you create is available in every Region.” — An AMI is **anchored to the Region** where it was created; to launch from it elsewhere you must **explicitly copy** it first, and the copy can fail if no matching AKI exists in the target Region.
- × “The template sections are all optional.” — **Resources is the only required section.** Parameters, Mappings, and Outputs are optional; a template with no Resources creates nothing.
- × “Passing EC2 status checks means CloudFormation knows the bootstrap finished.” — Status checks only prove the instance is **running**. Your init code must call **`cfn-signal`** against a **WaitCondition** to report that installs and configuration truly completed.
- × “A failed stack leaves the resources it created in place.” — The **default is to roll back and delete** everything it built. Use `--on-failure DO_NOTHING` (or `disable-rollback`) to keep resources for debugging — but you cannot set both.
- × “Deleting a stack always deletes every resource.” — Set `DeletionPolicy: Retain` (or `DeleteOnTermination: false` for an EBS volume) to preserve a resource, and some resources — such as a **non-empty S3 bucket** — may not be deleted at all.
- × “`Ref` always returns the same kind of value.” — Its return **depends on the resource**: an S3 bucket returns its **name**, an SQS queue its **URL**, an EC2 instance its **instance ID**. Use `Fn::GetAtt` for any other attribute.
- × “YAML indentation can use tabs.” — **Never use tabs in YAML.** Use spaces (two is standard); tabs are misread as whitespace and break the document. Comments start with #.
- × “Continuous delivery deploys straight to production.” — Continuous delivery auto-builds, tests, and **stages** the release, but a **human approves** the production deploy. Automatic push to production is **continuous deployment**.

08 SELF-QUIZ

answers are upside-down below

Q1 What is the only required section of an AWS CloudFormation template?

Q2 What term names the collection of AWS resources created from a template and managed as one unit?

Q3 Which CloudFormation capability identifies stack resources that were changed outside of CloudFormation?

Q4 Which intrinsic function retrieves a value from the Mappings section of a template?

Q5 Name two launch parameters that an EC2 launch template can store.

Q6 After an instance's bootstrap script finishes, which command signals a WaitCondition that it succeeded?

Q7 Which two log files do you analyze first when a WaitCondition times out or returns an error?

Q8 Which CloudFormation feature lets you preview how proposed changes affect running resources before you apply them?

Q9 In which two data-serialization formats can CloudFormation templates be written?

Q10 Which AWS service builds, tests, and deploys your code every time there is a code change, orchestrating the release?

ANSWER KEY (rotate the page)

Q1 Resources **Q2** a stack **Q3** drift detection **Q4** Fn::FindInMap **Q5** any two of: AMI ID, instance type, subnet, key pair, security group **Q6** cfn-signal **Q7** cfn-init.log and cfn-wire.log **Q8** a change set **Q9** JSON and YAML **Q10** AWS CodePipeline