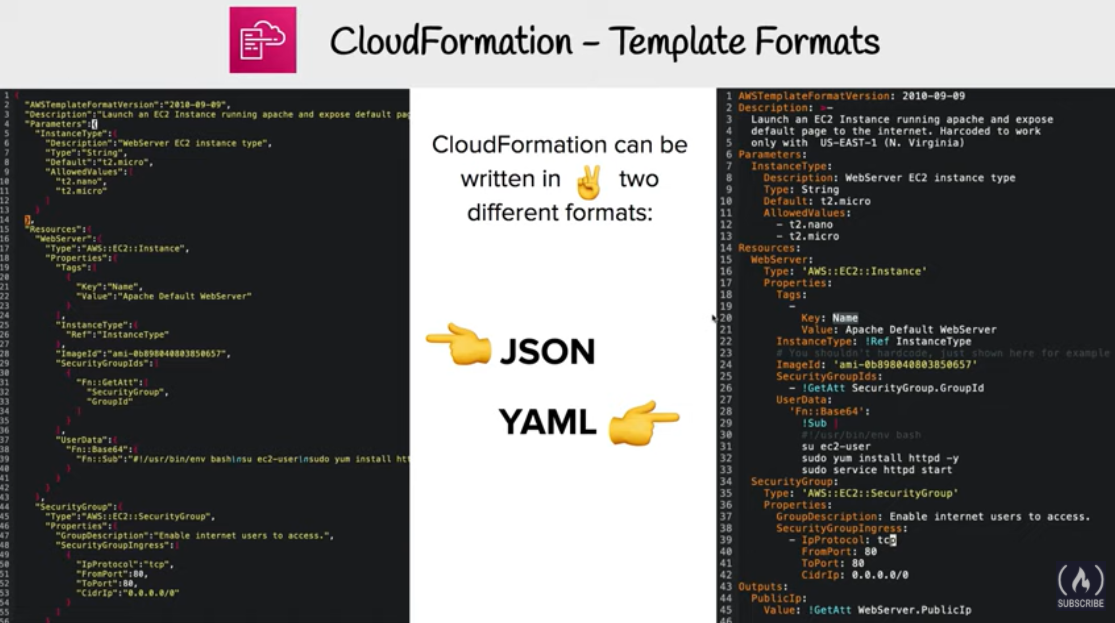
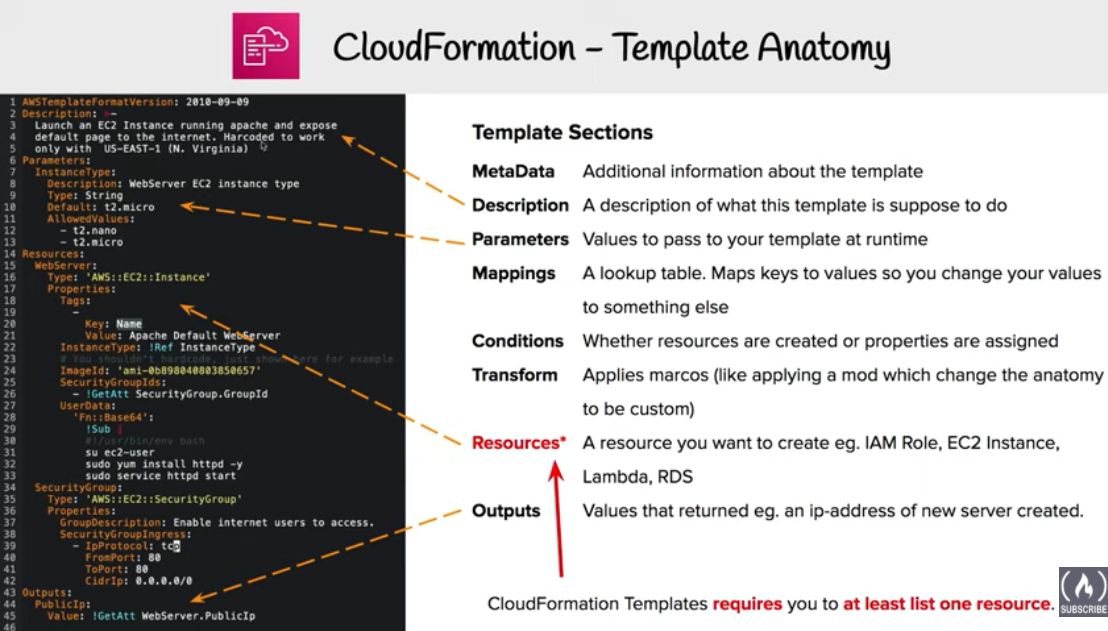
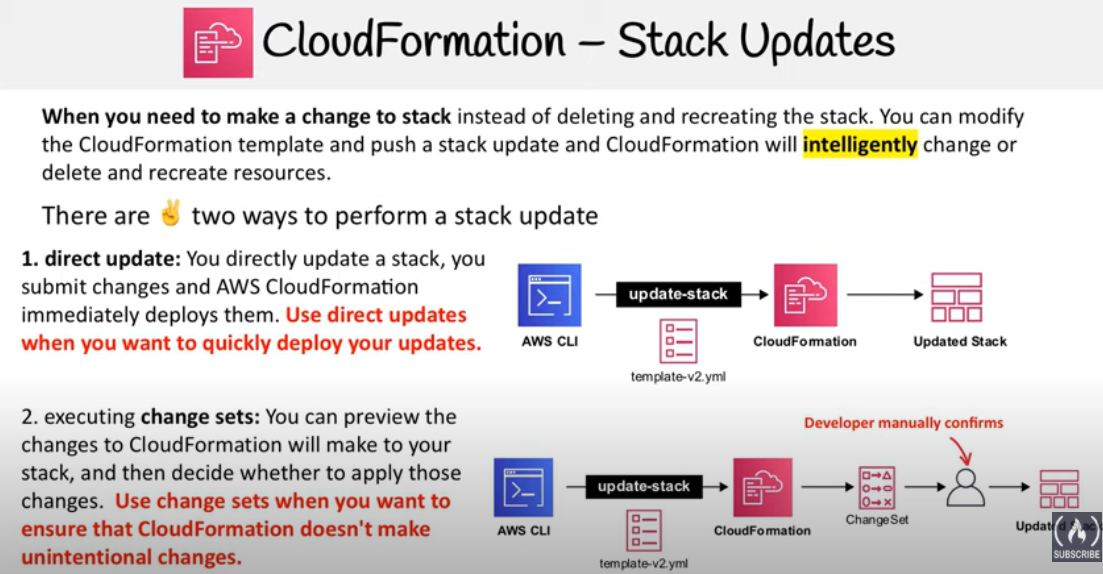
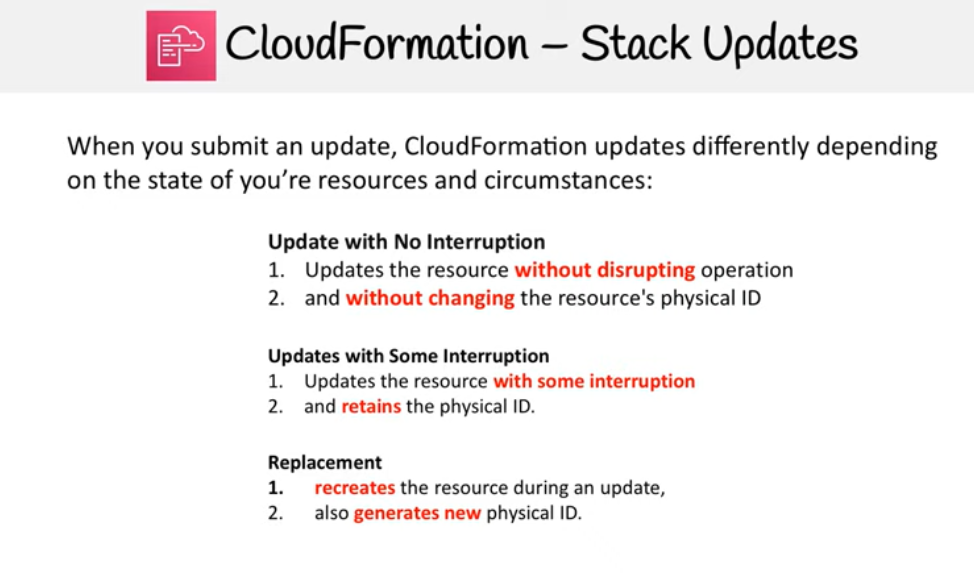
AWS CloudFormation

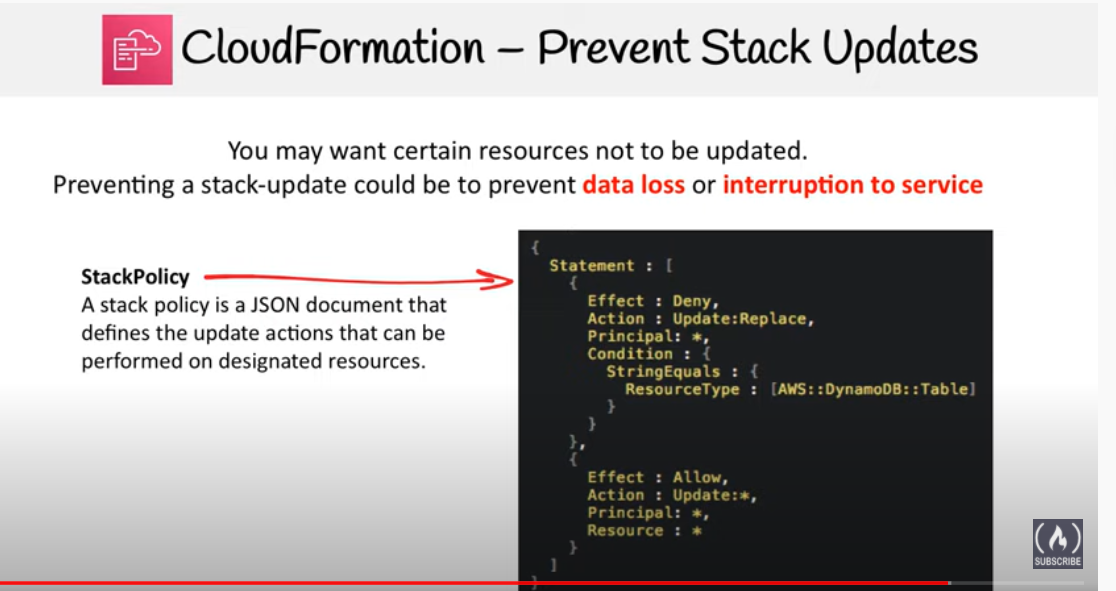
# 

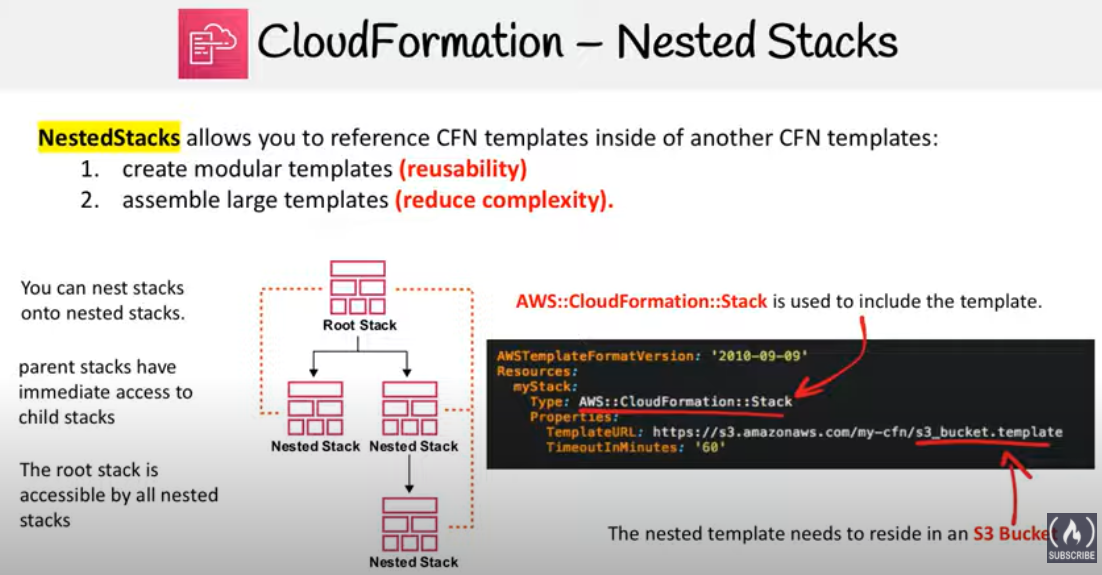


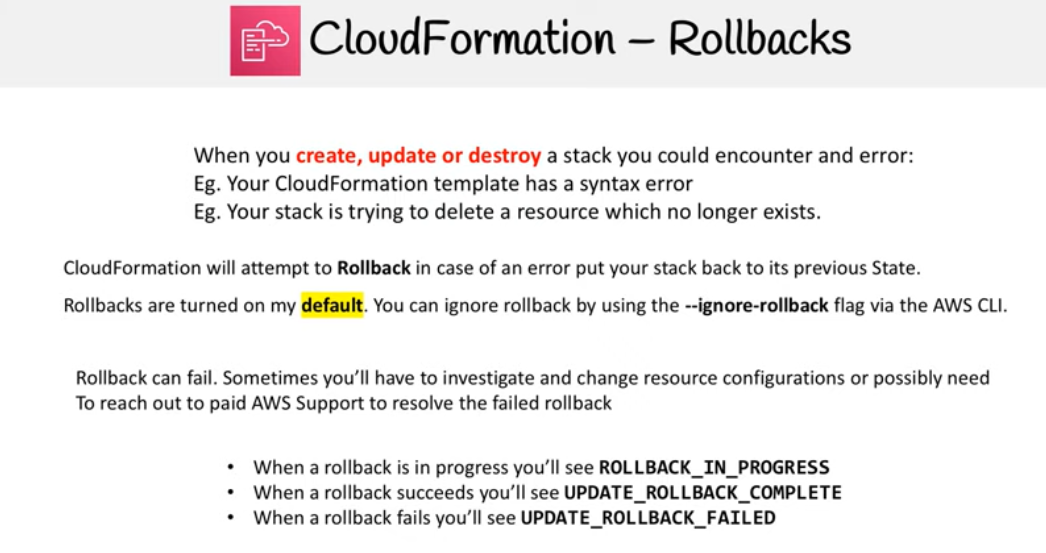


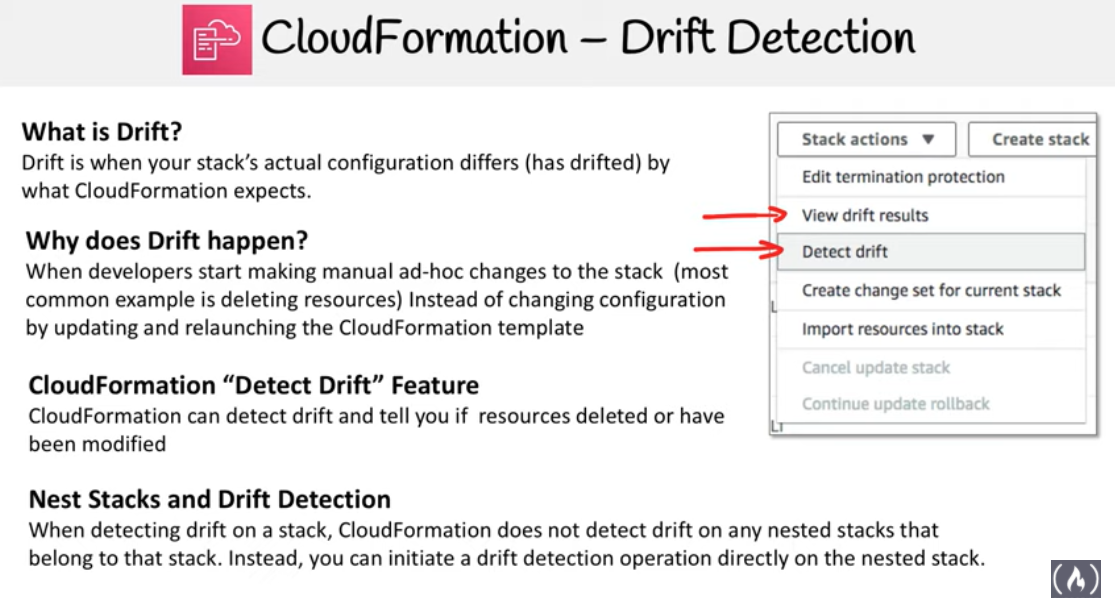


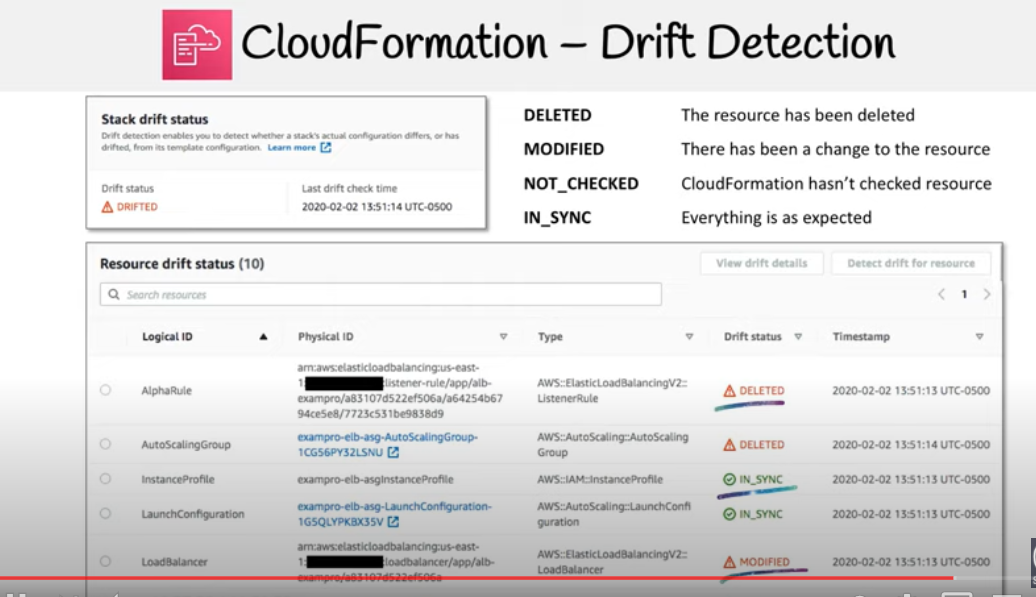


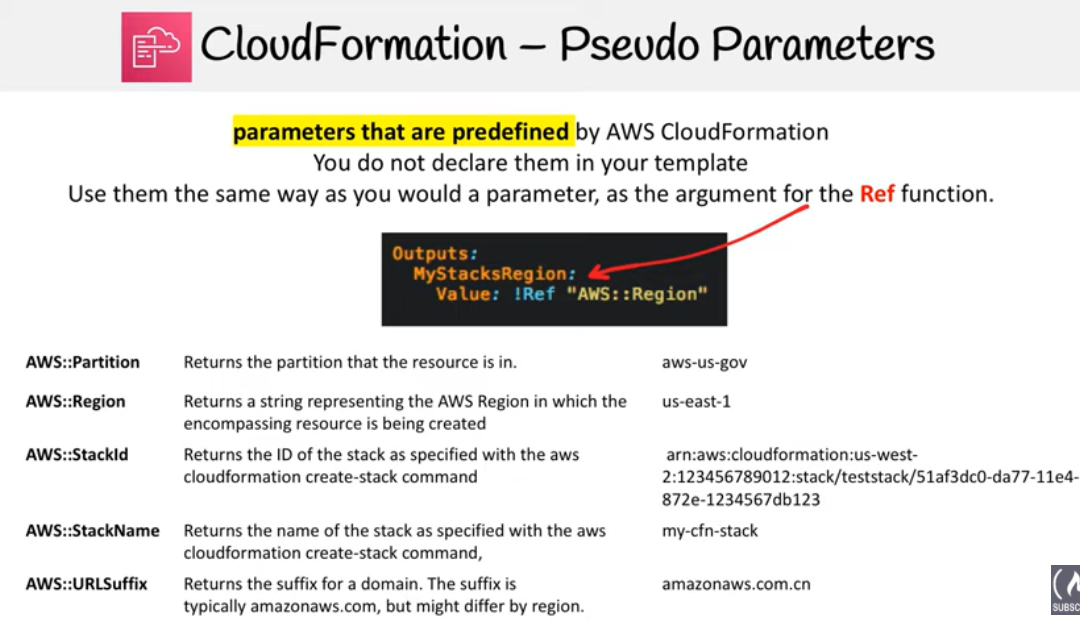


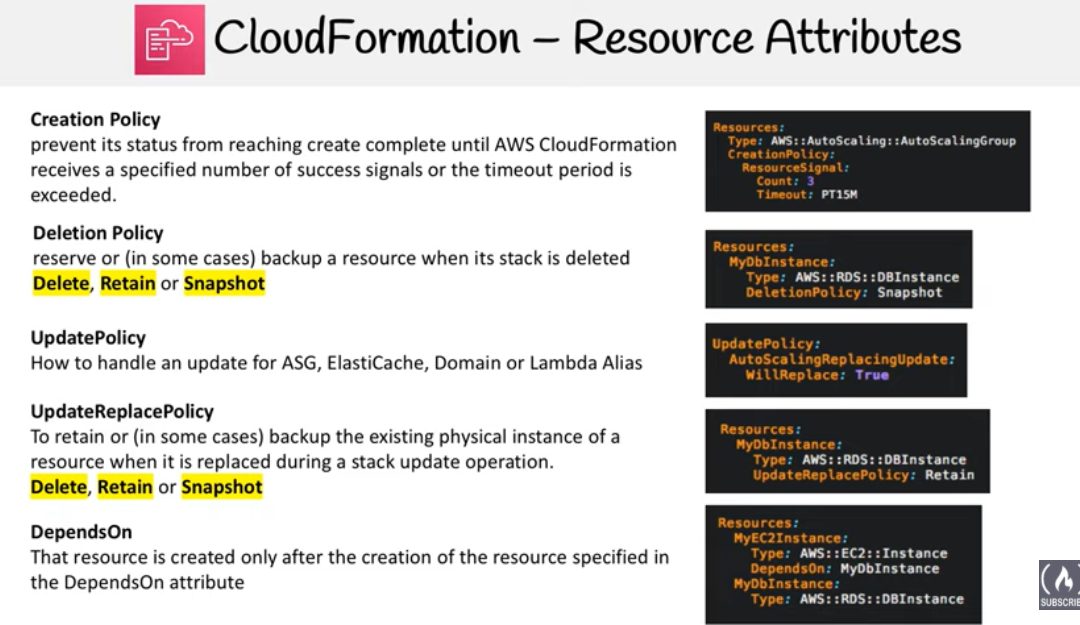


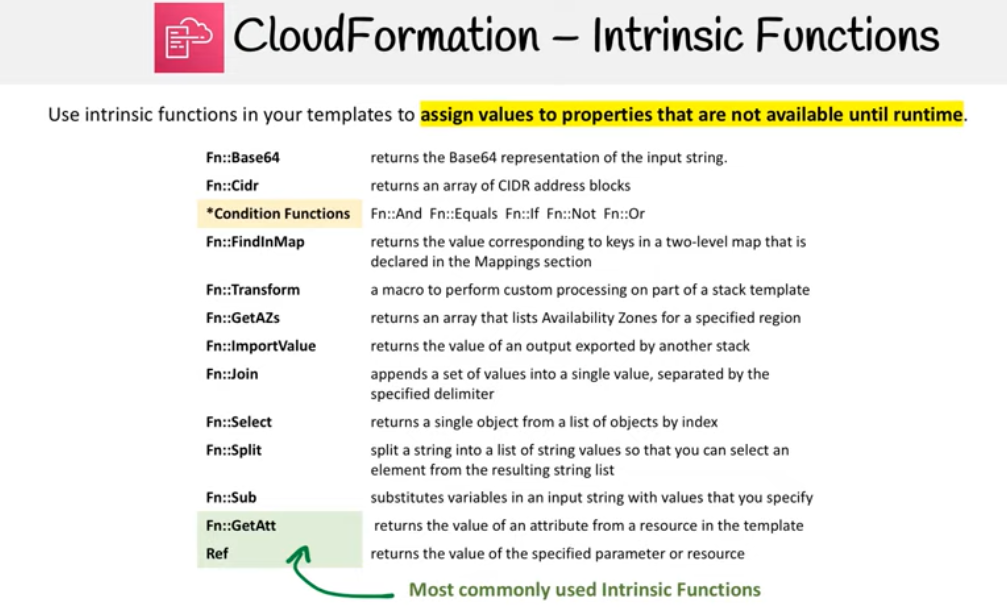


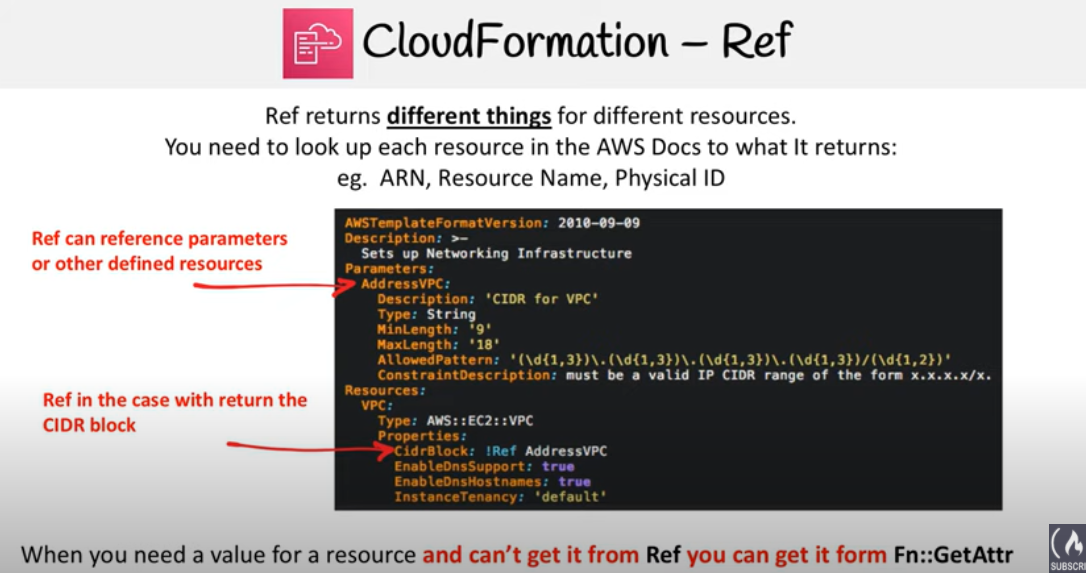


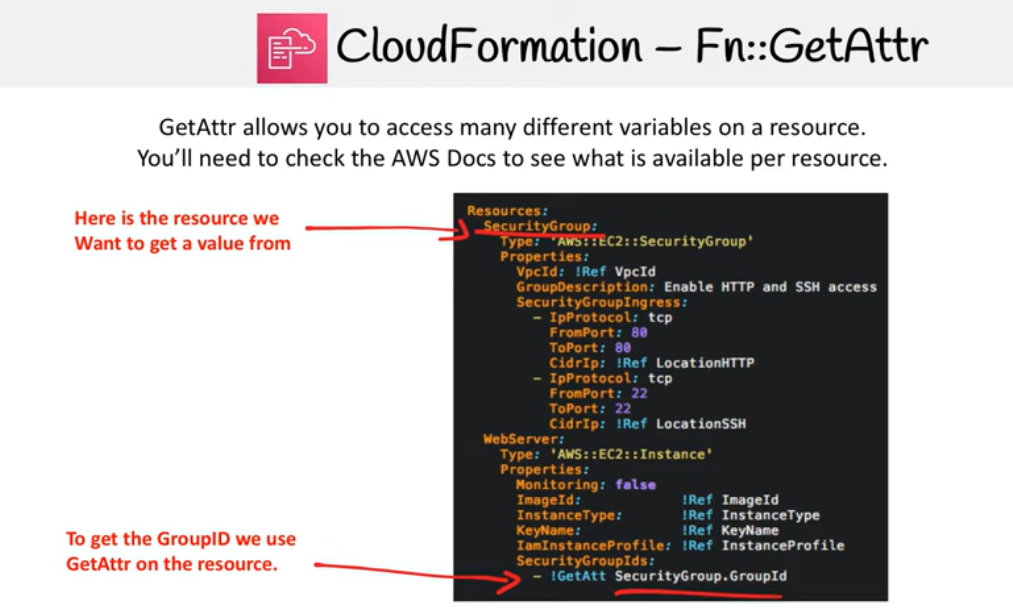


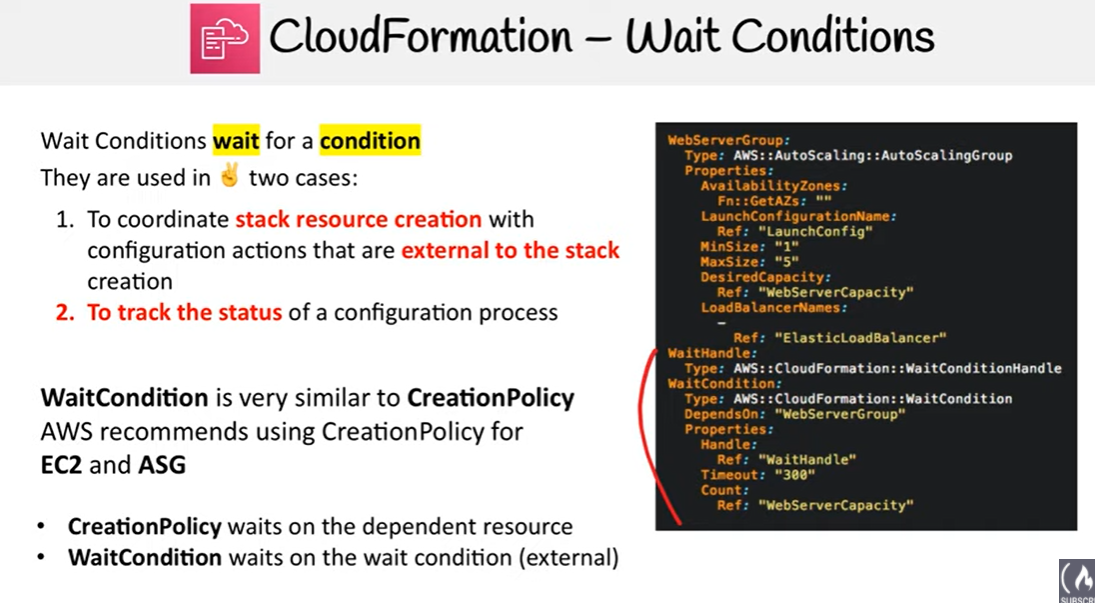




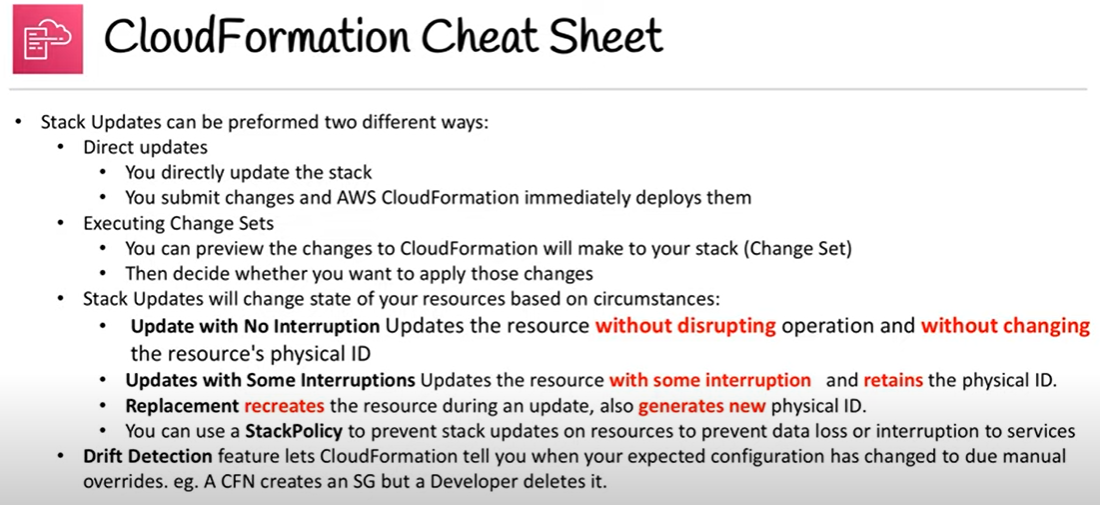


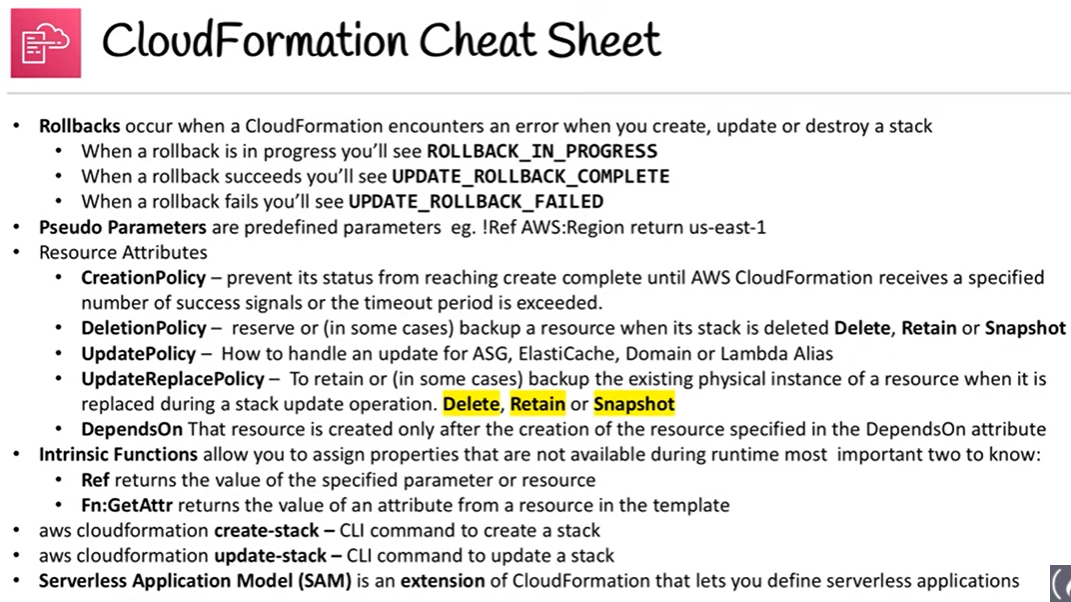






# 





# AWS CloudFormation

* AWS CloudFormation gives developers and systems administrators an easy way to create and manage a collection of related AWS resources, provision and update them in an orderly and predictable fashion
* CloudFormation consists of
  + Template
    - is an architectural diagram
    - a JSON or YAML-format, text-based file that describes all the AWS resources you need to deploy to run your application
  + Stack
    - is the end result of that diagram, which is actually provisioned
    - is the set of AWS resources that are created and managed as a single unit when CloudFormation instantiates a template.
* CloudFormation template can be used to set up the resources consistently and repeatedly over and over across multiple regions
* Resources can be updated, deleted and modified in a controlled and predictable way, in effect applying version control to the infrastructure as done for software code
* AWS CloudFormation Template consists of elements :-
  + List of AWS resources and their configuration values
  + An optional template file format version number
  + An optional list of template parameters (input values supplied at stack creation time)
  + An optional list of output values like public IP address using the Fn::GetAtt function
  + An optional list of data tables used to lookup static configuration values for e.g., AMI names per AZ
* CloudFormation supports Chef & Puppet Integration, meaning that you can deploy and configure right down the application layer
* CloudFormation provides a set of application bootstrapping scripts that enable you to install packages, files, and services on the EC2 instances by simply describing them in the CloudFormation template
* By default, **automatic rollback on error** feature is enabled, which will cause all the AWS resources that CloudFormation created successfully for a stack up to the point where an error occurred to be deleted.
* In case of automatic rollback, charges would still be applied for the resources the time they were up and running
* CloudFormation provides a ***WaitCondition*** resource that acts as a barrier, blocking the creation of other resources until a completion signal is received from an external source e.g. application, or management system
* CloudFormation allows deletion policies to be defined for resources in the template for e.g. resources to be retained or snapshots can be created before deletion useful for preserving S3 buckets when the stack is deleted

#### **Required Mainly for Developer, SysOps Associate & DevOps Professional Exam**

## AWS CloudFormation Concepts

* AWS CloudFormation, you work with templates and stacks
  + **Templates**
    - act as blueprints for building AWS resources.
    - is a JSON or YAML formatted text file, saved with any extension, such as .json, .yaml, .template, or .txt.
    - have additional capabilities to build complex sets of resources and reuse those templates in multiple contexts for e.g. use input parameters to create generic and reusable templates
    - Name used for a resource within the template is a logical name but when CloudFormation creates the resource, it generates a physical name that is based on the combination of the logical name, the stack name, and a unique ID
  + **Stacks**
    - Stacks manage related resources as a single unit,
    - Collection of resources can be created, updated, and deleted by creating, updating, and deleting stacks.
    - All the resources in a stack are defined by the stack’s AWS CloudFormation template
    - CloudFormation makes underlying service calls to AWS to provision and configure the resources in the stack and can perform only actions that the users have permission to do.
  + **Change Sets**
    - Change Sets presents a summary of the proposed changes CloudFormation will make when a stack is updated
    - Change Sets help check how the changes might impact running resources, especially critical resources, before implementing them
    - CloudFormation makes the changes to the stack only when the change set is executed, allowing you to decide whether to proceed with the proposed changes or explore other changes by creating another change set
    - Change sets don’t indicate whether AWS CloudFormation will successfully update a stack for e.g. if account limits are hit or the user does not have permission

## 

## CloudFormation Access Control

* **IAM**
  + IAM can be applied with CloudFormation to access control for users whether they can view stack templates, create stacks, or delete stacks
  + **IAM permissions need to be provided to the user to the AWS services and resources provisioned, when the stack is created**
  + Before a stack is created, AWS CloudFormation validates the template to check for IAM resources that it might create
* **Service Role**
  + A service role is an AWS IAM role that allows AWS CloudFormation to make calls to resources in a stack on the user’s behalf
  + By default, AWS CloudFormation uses a temporary session that it generates from the user credentials for stack operations.
  + For a service role, AWS CloudFormation uses the role’s credentials.
  + When a service role is specified, AWS CloudFormation always uses that role for all operations that are performed on that stack.

### Template Resource Attributes

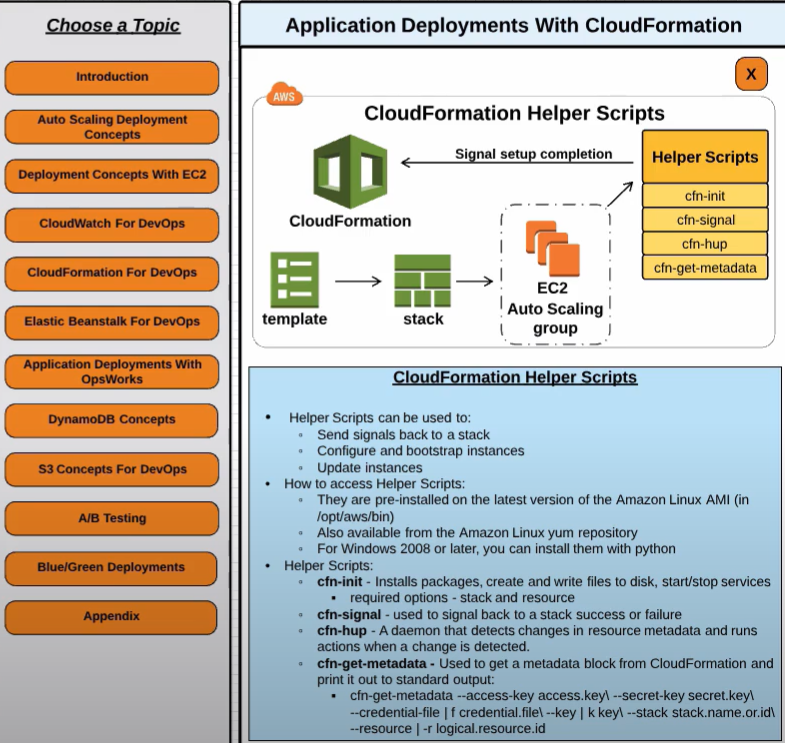
* **CreationPolicy** Attribute
  + is invoked during the associated resource creation
  + can be associated with a resource to prevent its status from reaching create complete until AWS CloudFormation receives a specified number of success signals or the timeout period is exceeded
  + helps to wait on resource configuration actions before stack creation proceeds for e.g. software installation on an EC2 instance
* **DeletionPolicy** Attribute
  + preserve or (in some cases) backup a resource when its stack is deleted
  + By default, if a resource has no DeletionPolicy attribute, AWS CloudFormation deletes the resource
  + To keep a resource when its stack is deleted,
    - specify **Retain** for that resource, to prevent deletion
    - specify **Snapshot** to create a snapshot before deleting the resource, if the snapshot capability is supported for e.g RDS, EC2 volume etc.



* **DependsOn** Attribute
  + helps specify that the creation of a specific resource follows another
  + resource is created only after the creation of the resource specified in the DependsOn attribute
* **Metadata** Attribute
  + enables association of structured data with a resource
* **UpdatePolicy** Attribute
  + defines AWS CloudFormation handles updates to the AWS::AutoScaling::AutoScalingGroup resource



Cloudformation Helper Scripts



## AWS CloudFormation Best Practices

Refer to Blog Post [CloudFormation Best Practices](https://jayendrapatil.com/aws-cloudformation-best-practices-certification/)

## Labs

* Qwiklabs Free labs
  + [Launching and Managing a Web Application with AWS CloudFormation](https://qwiklabs.com/focuses/349?locale=en&parent=catalog)

## AWS Certification Exam Practice Questions

1. What does Amazon CloudFormation provide?
   1. The ability to setup Autoscaling for Amazon EC2 instances.
   2. **A templated resource creation for Amazon Web Services.**
   3. A template to map network resources for Amazon Web Services
   4. None of these
2. A user is planning to use AWS CloudFormation for his automatic deployment requirements. Which of the below mentioned components are required as a part of the template?
   1. Parameters
   2. Outputs
   3. Template version
   4. **Resources**
3. In regard to AWS CloudFormation, what is a stack?
   1. Set of AWS templates that are created and managed as a template
   2. Set of AWS resources that are created and managed as a template
   3. **Set of AWS resources that are created and managed as a single unit**
   4. Set of AWS templates that are created and managed as a single unit
4. A large enterprise wants to adopt CloudFormation to automate administrative tasks and implement the security principles of least privilege and separation of duties. They have identified the following roles with the corresponding tasks in the company: (i) network administrators: create, modify and delete VPCs, subnets, NACLs, routing tables, and security groups (ii) application operators: deploy complete application stacks (ELB, Auto -Scaling groups, RDS) whereas all resources must be deployed in the VPCs managed by the network administrators (iii) Both groups must maintain their own CloudFormation templates and should be able to create, update and delete only their own CloudFormation stacks. The company has followed your advice to create two IAM groups, one for applications and one for networks. Both IAM groups are attached to IAM policies that grant rights to perform the necessary task of each group as well as the creation, update and deletion of CloudFormation stacks. Given setup and requirements, which statements represent valid design considerations? Choose 2 answers **[PROFESSIONAL]**
   1. **Network stack updates will fail upon attempts to delete a subnet with EC2 instances** (Subnets cannot be deleted with instances in them)
   2. Unless resource level permissions are used on the CloudFormation: DeleteStack action, network administrators could tear down application stacks (Network administrators themselves need permission to delete resources within the application stack & CloudFormation makes calls to create, modify, and delete those resources on their behalf)
   3. The application stack cannot be deleted before all network stacks are deleted (Application stack can be deleted before network stack)
   4. **Restricting the launch of EC2 instances into VPCs requires resource level permissions in the IAM policy of the application group**(IAM permissions need to be given explicitly to launch instances )
   5. Nesting network stacks within application stacks simplifies management and debugging, but requires resource level permissions in the IAM policy of the network group (Although stacks can be nested, Network group will need to have all the application group permissions)
5. Your team is excited about the use of AWS because now they have access to programmable infrastructure. You have been asked to manage your AWS infrastructure in a manner similar to the way you might manage application code. You want to be able to deploy exact copies of different versions of your infrastructure, stage changes into different environments, revert back to previous versions, and identify what versions are running at any particular time (development, test, QA, production). Which approach addresses this requirement?
   1. Use cost allocation reports and AWS Opsworks to deploy and manage your infrastructure.
   2. Use AWS CloudWatch metrics and alerts along with resource tagging to deploy and manage your infrastructure.
   3. Use AWS Beanstalk and a version control system like GIT to deploy and manage your infrastructure.
   4. **Use AWS CloudFormation and a version control system like GIT to deploy and manage your infrastructure.**
6. A user is usingCloudFormation to launch an EC2 instance and then configure an application after the instance is launched. The user wants the stack creation of ELB and AutoScaling to wait until the EC2 instance is launched and configured properly. How can the user configure this?
   1. It is not possible that the stack creation will wait until one service is created and launched
   2. The user can use the HoldCondition resource to wait for the creation of the other dependent resources
   3. The user can use the DependentCondition resource to hold the creation of the other dependent resources
   4. **The user can use the WaitCondition resource to hold the creation of the other dependent resources**
7. A user has created a CloudFormation stack. The stack creates AWS services, such as EC2 instances, ELB, AutoScaling, and RDS. While creating the stack it created EC2, ELB and AutoScaling but failed to create RDS. What will CloudFormation do in this scenario?
   1. CloudFormation can never throw an error after launching a few services since it verifies all the steps before launching
   2. It will warn the user about the error and ask the user to manually create RDS
   3. **Rollback all the changes and terminate all the created services**
   4. It will wait for the user’s input about the error and correct the mistake after the input
8. A user is planning to use AWS CloudFormation. Which of the below mentioned functionalities does not help him to correctly understand CloudFormation?
   1. **CloudFormation follows the DevOps model for the creation of Dev & Test**
   2. AWS CloudFormation does not charge the user for its service but only charges for the AWS resources created with it
   3. CloudFormation works with a wide variety of AWS services, such as EC2, EBS, VPC, IAM, S3, RDS, ELB, etc
   4. CloudFormation provides a set of application bootstrapping scripts which enables the user to install Software
9. A customer is using AWS for Dev and Test. The customer wants to setup the Dev environment with CloudFormation. Which of the below mentioned steps are not required while using CloudFormation?
   1. Create a stack
   2. **Configure a service**
   3. Create and upload the template
   4. Provide the parameters configured as part of the template
10. A marketing research company has developed a tracking system that collects user behavior during web marketing campaigns on behalf of their customers all over the world. The tracking system consists of an auto-scaled group of Amazon Elastic Compute Cloud (EC2) instances behind an elastic load balancer (ELB), and the collected data is stored in Amazon DynamoDB. After the campaign is terminated, the tracking system is torn down and the data is moved to Amazon Redshift, where it is aggregated, analyzed and used to generate detailed reports. The company wants to be able to instantiate new tracking systems in any region without any manual intervention and therefore adopted AWS CloudFormation. What needs to be done to make sure that the AWS CloudFormation template works in every AWS region? Choose 2 answers **[PROFESSIONAL]**
    1. IAM users with the right to start AWS CloudFormation stacks must be defined for every target region. (IAM users are global)
    2. The names of the Amazon DynamoDB tables must be different in every target region. (DynamoDB names should be unique only within a region)
    3. **Use the built-in function of AWS CloudFormation to set the AvailabilityZone attribute of the ELB resource.**
    4. Avoid using DeletionPolicies for EBS snapshots. (Don’t want the data to be retained)
    5. **Use the built-in Mappings and FindInMap functions of AWS CloudFormation to refer to the AMI ID set in the ImageId attribute of the Auto Scaling::LaunchConfiguration resource.**
11. A gaming company adopted AWS CloudFormation to automate load -testing of their games. They have created an AWS CloudFormation template for each gaming environment and one for the load -testing stack. The load – testing stack creates an Amazon Relational Database Service (RDS) Postgres database and two web servers running on Amazon Elastic Compute Cloud (EC2) that send HTTP requests, measure response times, and write the results into the database. A test run usually takes between 15 and 30 minutes. Once the tests are done, the AWS CloudFormation stacks are torn down immediately. The test results written to the Amazon RDS database must remain accessible for visualization and analysis. Select possible solutions that allow access to the test results after the AWS CloudFormation load -testing stack is deleted. Choose 2 answers. **[PROFESSIONAL]**
    1. **Define a deletion policy of type Retain for the Amazon QDS resource to assure that the RDS database is not deleted with the AWS CloudFormation stack.**
    2. **Define a deletion policy of type Snapshot for the Amazon RDS resource to assure that the RDS database can be restored after the AWS CloudFormation stack is deleted.**
    3. Define automated backups with a backup retention period of 30 days for the Amazon RDS database and perform point -in -time recovery of the database after the AWS CloudFormation stack is deleted. (as the environment is required for limited time the automated backup will not serve the purpose)
    4. Define an Amazon RDS Read-Replica in the load-testing AWS CloudFormation stack and define a dependency relation between master and replica via the DependsOn attribute. (read replica not needed and will be deleted when the stack is deleted)
    5. Define an update policy to prevent deletion of the Amazon RDS database after the AWS CloudFormation stack is deleted. (UpdatePolicy does not apply to RDS)
12. When working with AWS CloudFormation Templates what is the maximum number of stacks that you can create?
    1. 500
    2. 50
    3. **200**(Refer [link](https://aws.amazon.com/cloudformation/faqs/) – The limit keeps on changing to check for the latest)
    4. 10
13. What happens, by default, when one of the resources in a CloudFormation stack cannot be created?
    1. Previously created resources are kept but the stack creation terminates
    2. **Previously created resources are deleted and the stack creation terminates**
    3. Stack creation continues, and the final results indicate which steps failed
    4. CloudFormation templates are parsed in advance so stack creation is guaranteed to succeed.
14. You need to deploy an AWS stack in a repeatable manner across multiple environments. You have selected CloudFormation as the right tool to accomplish this, but have found that there is a resource type you need to create and model, but is unsupported by CloudFormation. How should you overcome this challenge? **[PROFESSIONAL]**
    1. Use a CloudFormation Custom Resource Template by selecting an API call to proxy for create, update, and delete actions. CloudFormation will use the AWS SDK, CLI, or API method of your choosing as the state transition function for the resource type you are modeling.
    2. Submit a ticket to the AWS Forums. AWS extends CloudFormation Resource Types by releasing tooling to the AWS Labs organization on GitHub. Their response time is usually 1 day, and they complete requests within a week or two.
    3. Instead of depending on CloudFormation, use Chef, Puppet, or Ansible to author Heat templates, which are declarative stack resource definitions that operate over the OpenStack hypervisor and cloud environment.
    4. **Create a CloudFormation Custom Resource Type by implementing create, update, and delete functionality, either by subscribing a Custom Resource Provider to an SNS topic, or by implementing the logic in AWS Lambda.**(Refer [link](http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/template-custom-resources.html))
15. What is a circular dependency in AWS CloudFormation?
    1. When a Template references an earlier version of itself.
    2. When Nested Stacks depend on each other.
    3. **When Resources form a DependOn loop.**(Refer [link](http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/troubleshooting.html#troubleshooting-errors-dependence-error), to resolve a dependency error, add a DependsOn attribute to resources that depend on other resources in the template. Some cases for e.g. EIP and VPC with IGW where EIP depends on IGW need explicitly declaration for the resources to be created in correct order)
    4. When a Template references a region, which references the original Template.
16. You need to run a very large batch data processing job one time per day. The source data exists entirely in S3, and the output of the processing job should also be written to S3 when finished. If you need to version control this processing job and all setup and teardown logic for the system, what approach should you use?
    1. Model an AWS EMR job in AWS Elastic Beanstalk. (cannot directly model EMR Clusters)
    2. **Model an AWS EMR job in AWS CloudFormation.**(EMR cluster can be modeled using CloudFormation. Refer [link](http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-emr-cluster.html))
    3. Model an AWS EMR job in AWS OpsWorks. (cannot directly model EMR Clusters)
    4. Model an AWS EMR job in AWS CLI Composer. (does not exist)
17. Your company needs to automate 3 layers of a large cloud deployment. You want to be able to track this deployment’s evolution as it changes over time, and carefully control any alterations. What is a good way to automate a stack to meet these requirements? **[PROFESSIONAL]**
    1. Use OpsWorks Stacks with three layers to model the layering in your stack.
    2. **Use CloudFormation Nested Stack Templates, with three child stacks to represent the three logical layers of your cloud.**(CloudFormation allows source controlled, declarative templates as the basis for stack automation and Nested Stacks help achieve clean separation of layers while simultaneously providing a method to control all layers at once when needed)
    3. Use AWS Config to declare a configuration set that AWS should roll out to your cloud.
    4. Use Elastic Beanstalk Linked Applications, passing the important DNS entries between layers using the metadata interface.
18. You have been asked to de-risk deployments at your company. Specifically, the CEO is concerned about outages that occur because of accidental inconsistencies between Staging and Production, which sometimes cause unexpected behaviors in Production even when Staging tests pass. You already use Docker to get high consistency between Staging and Production for the application environment on your EC2 instances. How do you further de-risk the rest of the execution environment, since in AWS, there are many service components you may use beyond EC2 virtual machines? **[PROFESSIONAL]**
    1. **Develop models of your entire cloud system in CloudFormation. Use this model in Staging and Production to achieve greater parity.**(Only CloudFormation’s JSON Templates allow declarative version control of repeatedly deployable models of entire AWS clouds. Refer [link](https://blogs.aws.amazon.com/application-management/blog/category/Best+practices))
    2. Use AWS Config to force the Staging and Production stacks to have configuration parity. Any differences will be detected for you so you are aware of risks.
    3. Use AMIs to ensure the whole machine, including the kernel of the virual machines, is consistent, since Docker uses Linux Container (LXC) technology, and we need to make sure the container environment is consistent.
    4. Use AWS ECS and Docker clustering. This will make sure that the AMIs and machine sizes are the same across both environments.
19. Which code snippet below returns the URL of a load balanced web site created in CloudFormation with an AWS::ElasticLoadBalancing::LoadBalancer resource name “ElasticLoad Balancer”? **[Developer]**
    1. **“Fn::Join” : [“”, [ “http://”, {“Fn::GetAtt” : [ “ElasticLoadBalancer”,”DNSName”]}]]**(Refer [link](http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/intrinsic-function-reference-getatt.html))
    2. “Fn::Join” : [“”,[ “http://”, {“Fn::GetAtt” : [ “ElasticLoadBalancer”,”Url”]}]]
    3. “Fn::Join” : [“”, [ “http://”, {“Ref” : “ElasticLoadBalancerUrl”}]]
    4. “Fn::Join” : [“”, [ “http://”, {“Ref” : “ElasticLoadBalancerDNSName”}]]
20. For AWS CloudFormation, which stack state refuses UpdateStack calls? **[Developer]**
    1. **<code>UPDATE\_ROLLBACK\_FAILED</code>**(Refer [link](http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/using-cfn-updating-stacks-continueupdaterollback.html))
    2. <code>UPDATE\_ROLLBACK\_COMPLETE</code>
    3. <code>UPDATE\_COMPLETE</code>
    4. <code>CREATE\_COMPLETE</code>
21. Which of these is not a Pseudo Parameter in AWS CloudFormation? **[Developer]**
    1. AWS::StackName
    2. AWS::AccountId
    3. **AWS::StackArn**(Refer [link](http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/pseudo-parameter-reference.html))
    4. AWS::NotificationARNs
22. Which of these is not an intrinsic function in AWS CloudFormation? **[Developer]**
    1. **Fn::SplitValue**(Refer [link](http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/intrinsic-function-reference.html))
    2. Fn::FindInMap
    3. Fn::Select
    4. Fn::GetAZs
23. Which of these is not a CloudFormation Helper Script? **[Developer]**
    1. cfn-signal
    2. cfn-hup
    3. **cfn-request**(Refer [link](http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/cfn-helper-scripts-reference.html))
    4. cfn-get-metadata
24. What method should I use to author automation if I want to wait for a CloudFormation stack to finish completing in a script? **[Developer]**
    1. Event subscription using SQS.
    2. Event subscription using SNS.
    3. **Poll using <code>ListStacks</code> / <code>list-stacks</code>.**(Only polling will make a script wait to complete. ListStacks / list-stacks is a real method. Refer [link](http://docs.aws.amazon.com/cli/latest/reference/cloudformation/list-stacks.html))
    4. Poll using <code>GetStackStatus</code> / <code>get-stack-status</code>. (GetStackStatus / get-stack-status does not exist)
25. Which status represents a failure state in AWS CloudFormation? **[Developer]**
    1. <code>UPDATE\_COMPLETE\_CLEANUP\_IN\_PROGRESS</code> (UPDATE\_COMPLETE\_CLEANUP\_IN\_PROGRESS means an update was successful, and CloudFormation is deleting any replaced, no longer used resources)
    2. <code>DELETE\_COMPLETE\_WITH\_ARTIFACTS</code> (DELETE\_COMPLETE\_WITH\_ARTIFACTS does not exist)
    3. **<code>ROLLBACK\_IN\_PROGRESS</code>**(ROLLBACK\_IN\_PROGRESS means an UpdateStack operation failed and the stack is in the process of trying to return to the valid, pre-update state Refer [link](http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/using-cfn-updating-stacks.html))
    4. <code>ROLLBACK\_FAILED</code> (ROLLBACK\_FAILED is not a CloudFormation state but UPDATE\_ROLLBACK\_FAILED is)
26. Which of these is not an intrinsic function in AWS CloudFormation? **[Developer]**
    1. Fn::Equals
    2. Fn::If
    3. Fn::Not
    4. **Fn::Parse**(Complete list of Intrinsic Functions: Fn::Base64, Fn::And, Fn::Equals, Fn::If, Fn::Not, Fn::Or, Fn::FindInMap, Fn::GetAtt, Fn::GetAZs, Fn::Join, Fn::Select, Refer [link](http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/intrinsic-function-reference.html))
27. You need to create a Route53 record automatically in CloudFormation when not running in production during all launches of a Template. How should you implement this? **[Developer]**
    1. **Use a <code>Parameter</code> for <code>environment</code>, and add a <code>Condition</code> on the Route53 <code>Resource</code> in the template to create the record only when <code>environment</code> is not <code>production</code>.**(Best way to do this is with one template, and a Condition on the resource. Route53 does not allow null strings for Refer [link](http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/conditions-section-structure.html))
    2. Create two templates, one with the Route53 record value and one with a null value for the record. Use the one without it when deploying to production.
    3. Use a <code>Parameter</code> for <code>environment</code>, and add a <code>Condition</code> on the Route53 <code>Resource</code> in the template to create the record with a null string when <code>environment</code> is <code>production</code>.
    4. Create two templates, one with the Route53 record and one without it. Use the one without it when deploying to production.