CloudFormation Section 4 Updating Stacks

**Updating Stacks:**

* Do you need to change a stack’s settings or resources? Update the stack instead of deleting it and creating a new stack is a good option.
* You can submit new parameters or and updated template
* CloudFormation compares the changes and updates only the changed resources
* Update methods:
  + Direct Update – Used to quickly deploy updates
  + Change Sets – Use to preview changes

Change sets allow you to preview changes and insure that no unexpected changes occur.

**Update behaviors:**

* CloudFormation updates resources based on differences between what you submit and the stack’s current template
* Resources that are not updated run without disruption during the update process
* CloudFormation uses the following behaviors for resources being updated
  + **Update with no interruption:** Updates resource without disrupting operation of that resource and without changing the resource’s physical ID
  + **Update with some interruption:** Updates the resource with some interruption and retains the physical ID
  + **Replacement:** Recreates the resource during an update, which also generates a new physical ID

**Update behaviors continued:**

* The update method used depends on which property you update for a given resource type. It is good to know resources update behavior before performing an update.
* The update behavior of a resource can dictates when to modify resources to reduce the impact of these changes on your application.

**Modifying a stack template:**

* To modify resources and properties of a Stack, you can modify the Stack’s template
* Use the template for the existing stack, and make your updates to that template
* Get a copy of your stack template from your source control or from CloudFormation
* You can update a template from the console or the CLI

**Directly Updating Stacks:**

* Submit a template or input parameters that specify updates to the resources in the stack
* Updates deploy immediately
* To use an existing template, make a copy and store locally or in S3 bucket

**Monitor Progress:**

* Monitor the progress of a stack’s update by viewing the stack’s events
* Update process starts with an UPDATE\_IN\_PROGRESS event for the stack
* After CloudFormation has successfully updated the stack you will see UPDATE\_COMPLETE
* If and update of a resource fails, you will see UPDATE\_FAILED
* If the update fails, CloudFormation rolls back any resource that it had updated during the update to their configuration before the update

**Updating Stacks Part 2:**

**Cancelling Updates:**

* After a stack update has begun, you can cancel the stack update if the stack is still in UPDATE\_IN\_PROGRESS state.
* You can’t cancel an update after it has finished, but you can update back to the previous settings.
* Cancelling the stack update will force a rollback to the previous state of the stack.

**Stack Policies:**

* All update actions are allowed on all resource by default
* By default, anyone with stack update permissions can update all the resources in the stack
* Stack policy can be used to protect stack resources from unintentional/mistaken updates or deletes
* Stack policy is a JSON document that defines the update actions that can be performed on designated resources
* With stack policy – all resources in the stack are protected by default
* Set an explicit allow on a resource to allow an update to it
* You can only have one stack policy per stack
* A stack policy applies to all users who attempt to update the stack
* A stack policy applies only during stack updates, it is not a replacement for IAM

**Change Sets:**

**Using Change Sets:**

* Allow a preview of how proposed changes will impact your running resources
* Create and manage change sets using the console, CLI, or API
* They do not indicate whether AWS CloudFormation will successfully update a stack

**Steps:**

1. Create a Change set
2. View the Change set
3. Create additional Change sets
4. Execute the Change set

**Creating a Change Set:**

* Create a change set for a running stack
* Submit the changes that you want to make by providing a modified template, new input parameter values, or both
* CloudFormation generates a change set by comparing the stack with the changes you submitted.
* After executing a change set, CloudFormation deletes all change sets associated with the stack because they are no longer valid for the updated stack
* To update a protected resource with a change set, you must first update the stack policy

**Cross Stack References:**

**Exporting Stacks:**

* What are Cross-Stack references? A CloudFormation feature which allows you to import and export stacks values between stacks.
* Why use them? Well, this is code after all. What are some of the key paradigms of software? Modularization, code reuse.
* Template can become very long and unmanageable. Break them up.
* Split up work amongst team members. Network teams and DB teams can work on separate templates but share data. Database team needs to know the subnet that will house their DB servers.
* Export the data from the network stack

**Cross-Stack References – Export:**

* Cross-Stack references are imports and exports between stacks
* Very simple and easy to use concept but very powerful

Example:

Outputs:

SubnetID1:

Value: !Ref Subnet1

Export: Name: ExampleSubnet

The export is implemented with just one line in the output section.

**Cross-Stack References – Import:**

* Note: The Export name must be unique within the AWS account and region
* Now you can import the value into your other stacks.

Example:

EC2Instance:

Type: AWS::EC2::Instance

Properties:

SubnetID:

- !ImportValue ExampleSubnet

The order of the stack creation matters. Create the stack with the exports first!

**Nested Stacks:**

* What are Nested Stacks? Stacks created as part of other stacks
* AWS::CloudFormation::Stack – allows creation of a stack within another stack
* Code reuse, modularization
* Common configurations: Load Balancers
* Nested stacks can contain nested stacks
* Stack updates should be initiated from the root stack

**Cross-Stack References vs Nested Stacks:**

* Nested Stacks – Deploy and manage all stacks from one stack
* Cross-Stack References – Manage all stacks separately
* Need to isolate sharing information? Use Nested Stacks
* Share information with other stacks? Use Cross-Stack References