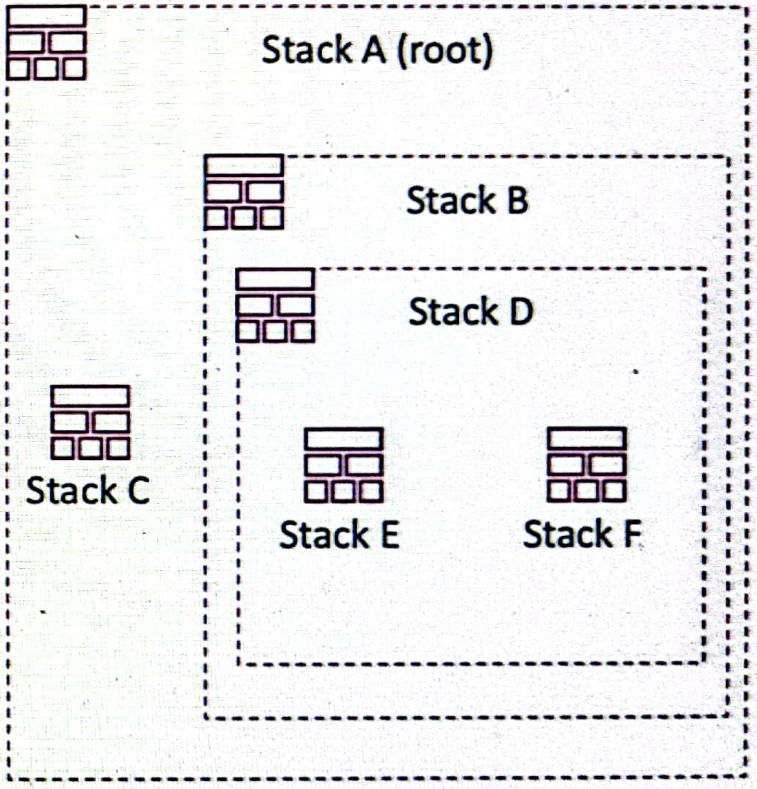
Nested Stacks Overview

- Nested stacks are stacks as part of other stacks
- They allow you to isolate repeated patterns / common components in separate stacks and call them from other stacks
- Example:
  - Load Balancer configuration that is re-used
  - Security Group that is re-used
- Nested stacks are considered best practice
- To update a nested stack, always update the parent (root stack)
- Nested stacks can have nested stacks themselves!



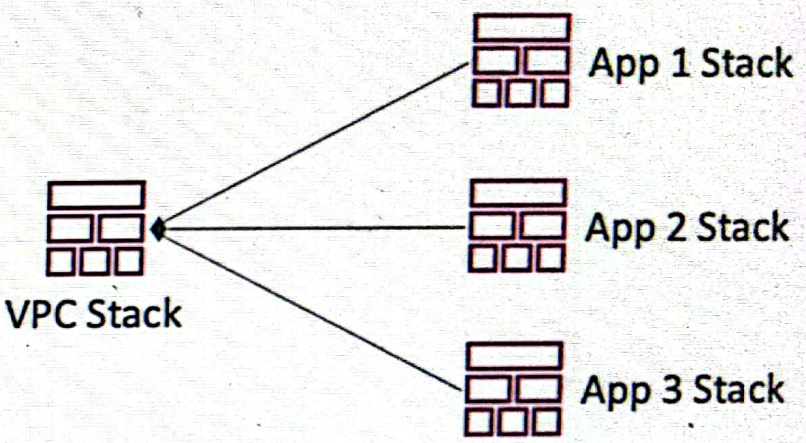
Nested Stacks Update

- To update a nested stack…

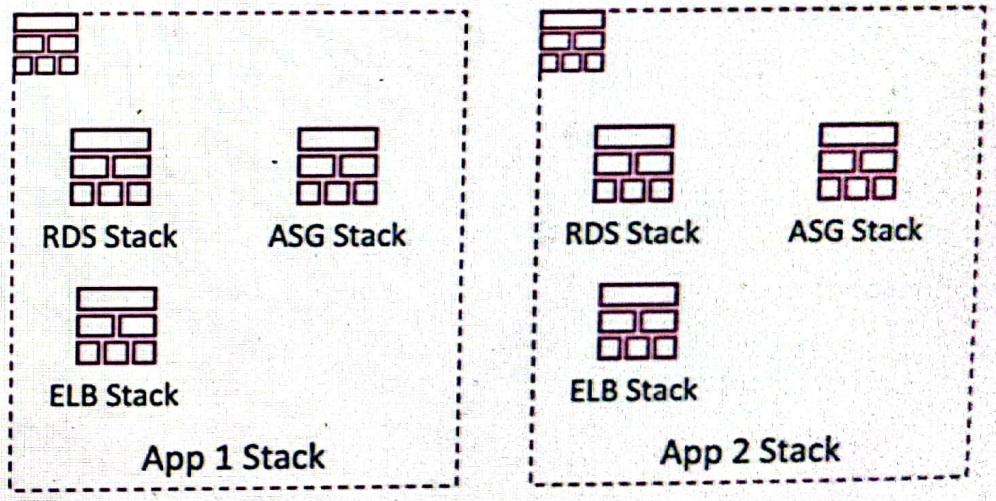
- Ensure the updated nested stacks are uploaded onto S3 first
- Then re-upload your root stack

Cross Stacks vs Nested Stacks

- Cross Stacks
  - Helpful when stacks have different lifecycles
  - Use Outputs Export and Fn::ImportValue
  - When you need to pass export values to many stacks (VPC Id, etc…)



- Nested Stacks
  - Helpful when components must be re-used
  - Ex: re-use how to properly configure an Application Load Balancer
  - The nested stack only is important to the higher-level stack (it's not shared)



Closing Comments on Nested Stacks

Closing Comments on Nested Stacks

Great nested stacks example at https://github.com/aws-samples/ecs-refarch-cloudformation/blob/master/master.yaml

Exported Stack Output Values vs. Using Nested Stacks:

- If you have a central resource that is shared between many different other stacks, use Exported Stack Output Values
- If you need other stacks to be updated right away if a central resource is updated, use Exported Stack Output Values
- If the resources can be dedicated to one stack only and must be re-usable pieces of code, use Nested Stacks
- Note that you will need to update each Root stack manually in case of Nested stack updated