Infrastructure as Code (CloudFormation)

CloudFormation Physical & Logical Resources

- CloudFormation defines logical resources within TEMPLATES (using YAML or JSON). -- The logical resource defines the WHAT, and leaves the HOW up to the CFN product.
- A CFN stack creates a physical resource for every logical resource - updating or deleting them as a template changes.
- TEMPLATES create STACKS. A stack creates/modifies/deletes physical resources based on the logical resources in the template

CloudFormation - Template and Pseudo Parameters

Template and Pseudo Parameters are two methods to provide INPUT to a template, which can influence what resources are provisioned, and the configuration of those resources.

- Pseudo parameters. These params are injected and made available by AWS. Template params are user made with YAML/JSON

CloudFormation Intrinsic Functions

AWS CloudFormation provides several built-in functions that help you manage your stacks. Use intrinsic functions in your templates to assign values to properties that are not available until runtime.

- !Ref & Fn::GetAtt. Ref is primary value, attributes are secondaries. !GetAtt
- Fn::Join & Fn::Split
- Fn::GetAZs & Fn::Select
- Conditions Fn::IF,And,Equals,Not,Or
- Fn::Base64 & Fn::Sub. Base64 outputs base64 encoded text
- Fn::Cidr. Creating a cider range based on parent VPC CIDR range

CloudFormation Mappings

The optional Mappings section matches a key to a corresponding set of named values. For example, if you want to set values based on a region, you can create a mapping that uses the region name as a key and contains the values you want to specify for each specific region

- Use the Fn::FindInMap intrinsic function to retrieve values in a map.
- Mapping -> key:value
- Mappings Improves Template Portability

CloudFormation Outputs

The optional Outputs section declares output values that you can import into other stacks (to create cross-stack references), return in response (to describe stack calls), or view on the AWS CloudFormation console.

- For example, you can output the S3 bucket name for a stack to make the bucket easier to find.
- Output section is optional

- Visible as outputs from CLI, console UI, parent stack, can be exported to other stacks (cross-stack references)
- Components: Description and Value

DEMO - Template v2 - Portable - Skipping for now

CloudFormation Conditions

The optional Conditions section contains statements that define the circumstances under which entities are created or configured. You might use conditions when you want to reuse a template that can create resources in different contexts, such as a test environment versus a production environment.

- Conditions evaluated to True or False and are processed BEFORE resources are created
- In your template, you can add an EnvironmentType input parameter, which accepts either prod or test as inputs.
- Conditions are evaluated based on predefined pseudo parameters or input parameter values that you specify when you create or update a stack.
- Within each condition, you can reference another condition, a parameter value, or a mapping.
- Components: Parameter, Conditions, Resources

CloudFormation DependsOn

With the DependsOn attribute you can specify that the creation of a specific resource follows another. When you add a DependsOn attribute to a resource, that resource is created only after the creation of the resource specified in theDependsOn attribute.

CloudFormation Wait Conditions & cfn-signal

CreationPolicy, WaitConditions and cfn-signal can all be used together to prevent the status if a resource from reaching create complete until AWS CloudFormation receives a specified number of success signals or the timeout period is exceeded.

- The cfn-signal helper script signals AWS CloudFormation to indicate whether Amazon EC2 instances have been successfully created or updated.
- After you define all your conditions, you can associate them with resources and resource properties in the Resources and Outputs sections of a template
- tl;dr resource create paused until expected signal(s) received

CloudFormation Nested Stacks

Nested stacks allow for a hierarchy of related templates to be combined to form a single product

- A ROOT STACK can contain and create nested stacks .. each of which can be passed parameters and provide back outputs.
- Nested stacks should be used when the resources being provisioned **share a lifecycle** and are related.
- Single stacks have resource limits of 500, use nested to overcome this
- Stacks are isolated; can't easily reference each other. Nested stacked can reference outputs from parents, however

- Nested stacks for modular-type template re-use. Cross-reference stacks are for resource re-use

CloudFormation CloudFormation Cross-Stack References

Nested stacks for modular-type template re-use. CloudFormation Cross-Stack References are for resource re-use

- Outputs in one stack reference logical resources or attributes in that stack
- Outputs can be exported, and then using the !ImportValue intrinsic function, referenced from another stack.
    - Exports must be UNIQUELY NAMED in region and account
    - Cross-account exports don't work

CloudFormation Stack Sets

StackSets are a feature of CloudFormation allowing infrastructure to be deployed and managed across MULTIPLE REGIONS and MULTIPLE ACCOUNTS from a single location.

- Additionally it adds a dynamic architecture - allowing automatic operations based on accounts being added or removed from the scope of a StackSet.
- StackSets are containers in an admin account which contain stack instances which REFERENCE stacks
    - Stack instances and stacks are in TARGET ACCOUNTS
- Each stack is 1 region 1 account

CFN StackSet Terms

- Concurrent Accounts. How many concurrent stacks are created together. Eg. If you have 10 stacks to make, you set concurrnet acct to 2... you'll create stacks in 5 pairs.
- Failure Tolerance. Amount of individual deployments that can fail before StackSet is considered failed
- Retain Stacks. Delete stacks FROM YOUR stack set, but save them so they continue to RUN INDEPENDENTLY of your stack set by choosing the Retain Stacks option. You can then manage retained stacks outside of your stack set in AWS CloudFormation. "cut it loose"

CloudFormation Deletion Policy

With the DeletionPolicy attribute you can preserve or (in some cases) backup a resource when its stack is deleted. You specify a DeletionPolicy attribute for each resource that you want to control. If a resource has no DeletionPolicy attribute, AWS CloudFormation deletes the resource by default.

- Deletion Policies: Delete, Retain, or Snapshot (if supported)
    - Snapshots continue past stack lifetime
    - Retain keeps resource, but doesn't manage it
    - Delete is default
- DeletionPolicy is only applied when the stack is deleted, not when the stack is updated. If you update a stack, the DeletionPolicy attribute is ignored but the resources are deleted and re-created. This might create data loss.

CloudFormation Stack Roles

Stack roles allow an IAM role to be passed into the stack via PassRole

- A stack uses this role, rather than the identity interacting with the stack to create, update and delete AWS resources.
- It allows ROLE SEPARATION and is a powerful security feature

CloudFormation Init (CFN-INIT)

CloudFormationInit and cfn-init are tools which allow a desired state configuration management system to be implemented within CloudFormation

- Another way to provide config info into EC2 instance
- Use the AWS::CloudFormation::Init type to include metadata on an Amazon EC2 instance for the cfn-init helper script. If your template calls the cfn-init script, the script looks for resource metadata rooted in the AWS::CloudFormation::Init metadata key. cfn-init supports all metadata types for Linux systems & It supports some metadata types for Windows
- AWS::CloudFormation::Init which is procedural (you tell it how to bootstrap itself)
- ... or CFN-INIT which is where you tell the system WHAT you want and it builds it out (this one is idempotent). It is run ONCE as a part of bootstrapping; if CloudFormation::Init is updated, cfn-init isn't re-run (which is where cfn-hup comes in, see below)

CloudFormation cfn-hup

The cfn-hup helper is a daemon that detects changes in resource metadata and runs user-specified actions when a change is detected (Ie. re-running cfn-init). This allows you to make configuration updates on your running Amazon EC2 instances through the UpdateStack API action.

CloudFormation ChangeSets - CFN stack update dry runs (like Find/Replace dry runs in WordPress)

When you need to update a stack, understanding how your changes will affect running resources before you implement them can help you update stacks with confidence. Change sets allow you to preview how proposed changes to a stack might impact your running resources, for example, whether your changes will delete or replace any critical resources, AWS CloudFormation makes the changes to your stack only when you decide to execute the change set, allowing you to decide whether to proceed with your proposed changes or explore other changes by creating another change set.

CloudFormation Custom Resources

Custom resources enable you to write custom provisioning logic in templates that AWS CloudFormation runs anytime you create, update (if you changed the custom resource), or delete stacks

- AKA Custom Resources let CFN integrate with anything it doesn't yet, or doesn't natively support