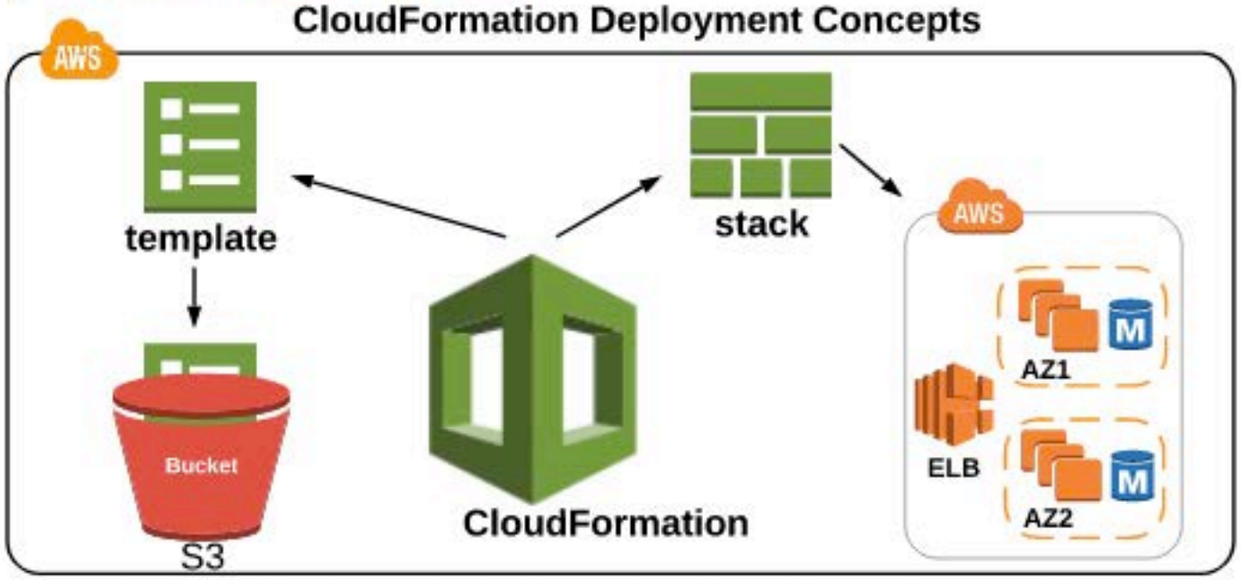
Choose a Topic

Application Deployments With CloudFormation

Scenario: Using CloudFormation to create infrastructure, known as infrastructure as code, is a common technique that a DevOps Professional will need to master. The code is a CloudFormation template which describes the infrastructure that will be deployed. The infrastructure is a CloudFormation Stack, created from the template. In this scenario we will create a stack containing an Auto Scaling Group of EC2 instances backed by an Amazon RDS database. The section will cover several ways to work with the stack using Creation Policies, Helper Scripts, Stack Policies, Wait Conditions, and Update Policies.

CloudFormation Deployment Concepts

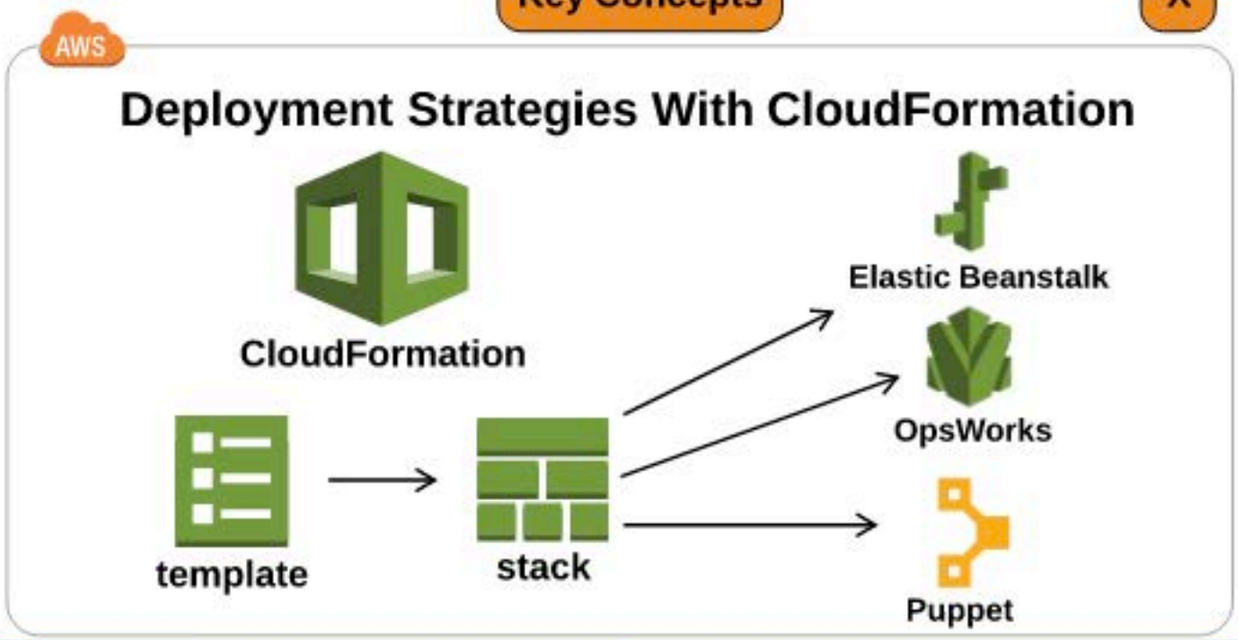


Choose a Topic

Application Deployments With CloudFormation

Key Concepts

X



CloudFormation Deployment Strategies

- When considering CloudFormation Deployments, we need to look at: Deployment Methods, Environments, Requirements (time, cost, application), and the lifecycle of the application.
- Bootstrapping applications with CloudFormation (Helper Scripts) - cfn-init, cfn-signal, cfn-get-metadata, cfn-hup.
- Bootstrapping considerations
  - Can speed up by pre-baking AMIs and deploy with auto scaling.
  - Use NoEcho property set to true to hide sensitive information.
  - Perform rolling update on Auto Scaling groups to avoid downtime.
- Deploying with CloudFormation and Puppet
  - Use Puppet to provision, configure, and patch applications.
  - Puppet deployments have a master and client nodes.
  - Puppet provides config mgmt and ensure the state of our instances.
- Deploying with CloudFormation and OpsWorks
  - OpsWorks supports Chef recipes.
  - OpsWorks supports dynamic configurations (ideal for apps with longer lifecycles).
  - Use CloudFormation to manage OpsWorks resources while Opsworks configures software, deploys applications, scales the envrionment, and monitors resources.
  - OpsWorks provides a higher level of abstraction, making it more convenient to deploy certain environments.

Choose a Topic

Application Deployments With CloudFormation

X

AWS

CloudFormation Key Concepts

- CloudFormation is "infrastructure as code" which can be version controlled, encourages collaboration, and can be automated.
- Can use a single template to create an environment or use multiple templates.
- Can interact with other tools such as Puppet (example: CloudFormation creates instances, Puppet puts the instances in a specific state).
- Updating a stack examples:
  - Changing the AMI of our instances
  - Updating a CloudWatch alarm
  - Updating Auto Scaling Groups
- Updating Stack steps:
  - Update the template
  - Update the stack with the new template
- Stack Policies - Can be used to prevent resource updates
  - json documents
  - Similar to IAM and Bucket Policies
- Considerations for updating a stack:
  - How will the update affect the resource
  - Is the change mutable or immutable
- Rollbacks and Deletions of Stacks
  - Can change the default behavior to not roll back. Investigate the resource that caused any issues by checking logs. Then manually delete the stack.
  - If ANYTHING fails during stack creation, the stack will be rolled back.
- Rollback Failure reasons:
  - Nested Stacks - dependencies between resources
  - A resource was modified outside of the template

Choose a Topic

Application Deployments With CloudFormation

X

AWS

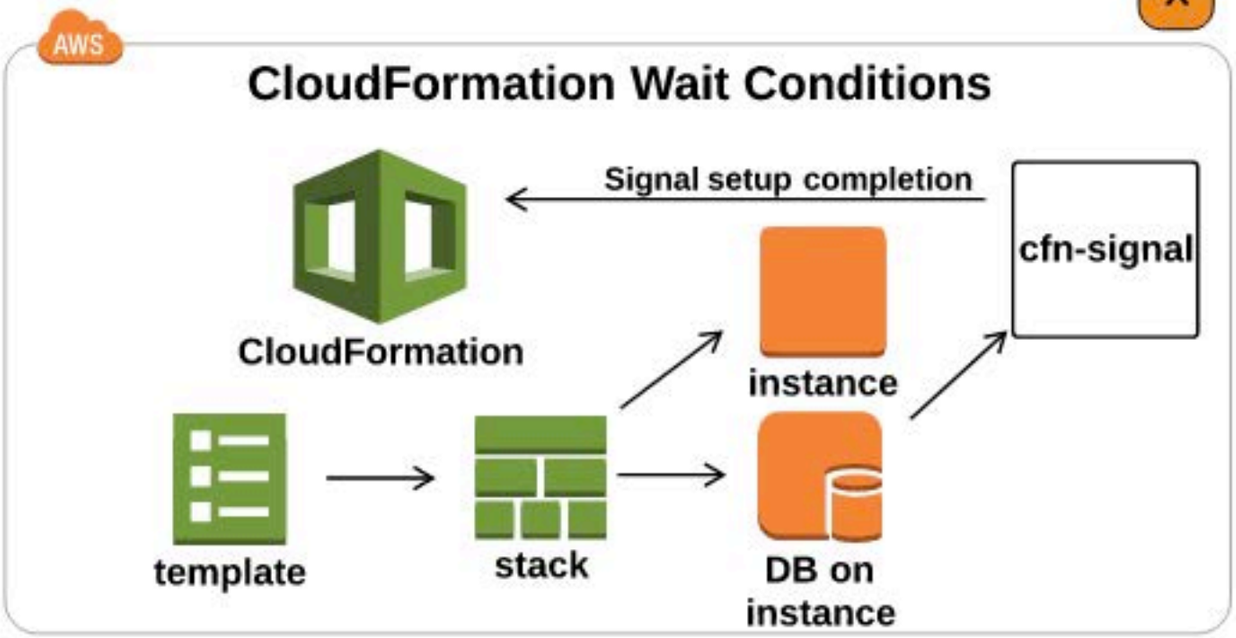
CloudFormation Deployments Key Concepts

- Deployment with CloudFormation and Elastic Beanstalk
  - Elastic Beanstalk can handle deployments:
    - All at once deployment
    - Rolling Deployment
    - Rolling with additional batch
    - Immutable Deployment
  - This deployment can help decouple our environment.
  - Can integrate Elastic Beanstalk with services like RDS, S3, DynamoDB without having to throw away these important resources.
- Using Elastic Beanstalk with CloudFormation vs using OpsWorks with CloudFormation:
  - Doesn't allow as much flexibility for some configurations and deployments.
  - Is more suitable for shorter application lifecycles where an environment can be thrown away with each deployment.
- Deploying with CloudFormation and code services:
  - Can use CloudFormation to provision the infrastructure, and then depoy and manage applications through CodeDeploy, CodeCommit, and CodePipeline.
- In-Place vs Disposable Methods
  - In-Place upgrades - perform updates on existing resources, faster (don't have to wait on new resources), can be used with applications that don't keep sessions, OpsWorks and Puppet can be used.
  - Disposable Upgrades - Roll out a new set of resources and remove older ones, work well with immutable infrastructure and Blue/Green deployments, Elastic Beanstalk and CloudFormation are better suited for this (although OpsWorks and Puppet can be used).

Choose a Topic

Application Deployments With CloudFormation

X



CloudFormation Wait Conditions

- Can be used to pause the creation of a stack and wait for a signal to continue.
- Useful when creating and configuring resources outside the template.
- Declared in the template.
- Remain in CREATE_IN_PROGRESS state until they receive the required number of success signals or time out.
- Using Ref with the handle gives access to a pre-signed url where we can send success or failure signals.
- To send a success or failure signal, we can send an HTTP request with the pre-signed url:

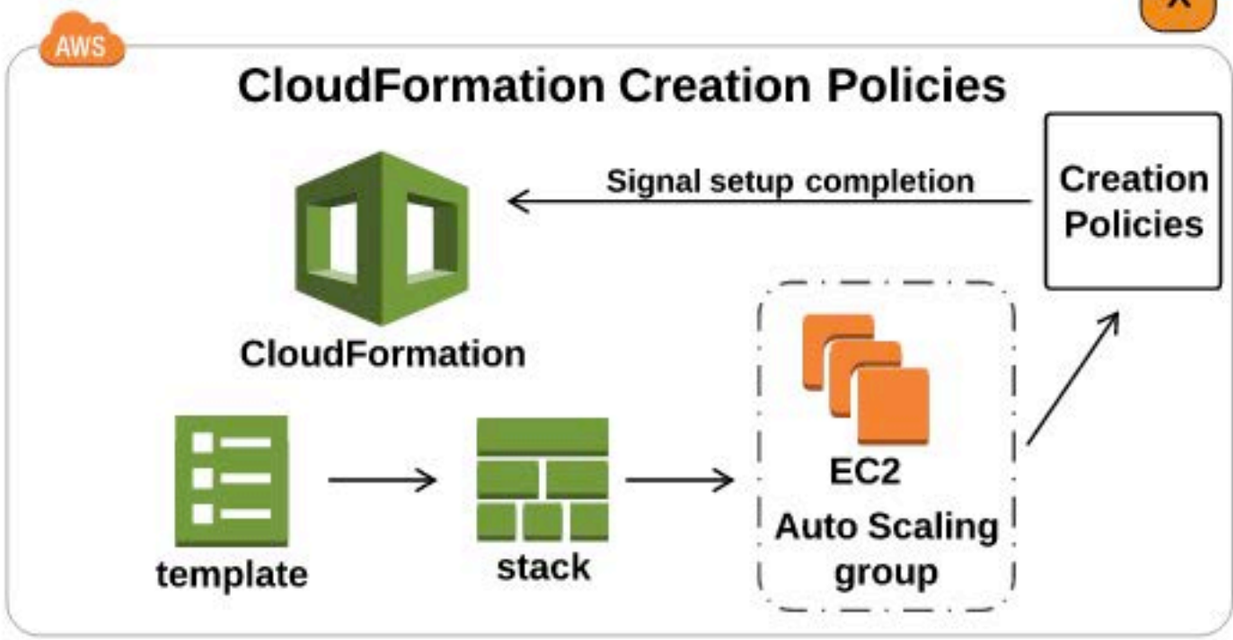
```
{
    "Status" : "SUCCESS (or FAILURE)",
    "UniqueId" : "ID1567",
    "Data" : "The application is ready",
    "Reason" : "configuration complete"
}
```

- When should Wait Conditions be used? Use Cases:
  - Synchronizing resource creation between different resources in the template.
  - Waiting for external resources (like an on-prem resource) to be created.
- Can also combine the use of DependsOn and WaitConditions to make the latter wait for a resource to be created before the timeout clock starts ticking.

Choose a Topic

Application Deployments With CloudFormation

X



CloudFormation Creation Policies

- Creation Policies pause the creation of a resource until a certain number of success signals have been received within a time period.
- Use Cases: Useful when configuring EC2 instances with bootstrapping or EC2 instances in Auto Scaling Groups.
- Signals can be sent back to the stack using helper scripts or through the SignalResource API or CLI call (signals can be seen in events of the stacks).
- CloudFormation invokes the CreationPolicy when its associated resources is created.
- Creation Policies and Wait Conditions
  - Can use Creation Policies with Wait Conditions in order to:
    - Nest the Creation Policy inside the Wait Condition
    - Track the progress of bootstrapping an instance
    - Coordinate the creation of different resources
      - Example: Once an action on an instance has completed, trigger the creation of another resource that depended on that action.

Choose a Topic

Back

AWS

CloudFormation Cross-Stack References

Network Stack

Application Stack

```
Network Stack
Outputs:
  VPC
    Description: Reference VPC
    Value: !Ref VPC
    Export:
      Name: ProdVPC
```

```
Application Stack
Resources:
  myTargetGroup
  Type: AWS::ELBv2::TargetGroup
  Properties:
    VPCId:
      Fn::ImportValue: ProdVPC
```

CloudFormation Cross-Stack References

- A CloudFormation Best Practice is to separate resources into multiple templates in a layered, or service-oriented architecture.
- This approach can be used to split templates out by department such as networking, application, and security. Each group can have their own template and avoid issues incurred when one template is shared amongst several groups.
- But how can these separate groups share resource information between their separate templates? Cross-Stack References.
- Cross-Stack References can be used to export resources from one AWS CloudFormation stack to another.
- If your Application team needs a VPC ID from the Network Stack, the VPC ID can be outputted from the Network Stack and imported to the Application Stack.

Choose a Topic

Back

AWS

CloudFormation Drift Detection

CloudFormation

Has someone modified our Stack resources outside of the Stack? That is bad practice! CloudFormation Drift Detection can detect these changes and help us restore the integrity of our Stacks.

Template

Stack

AWS

CloudFormation Drift Detection

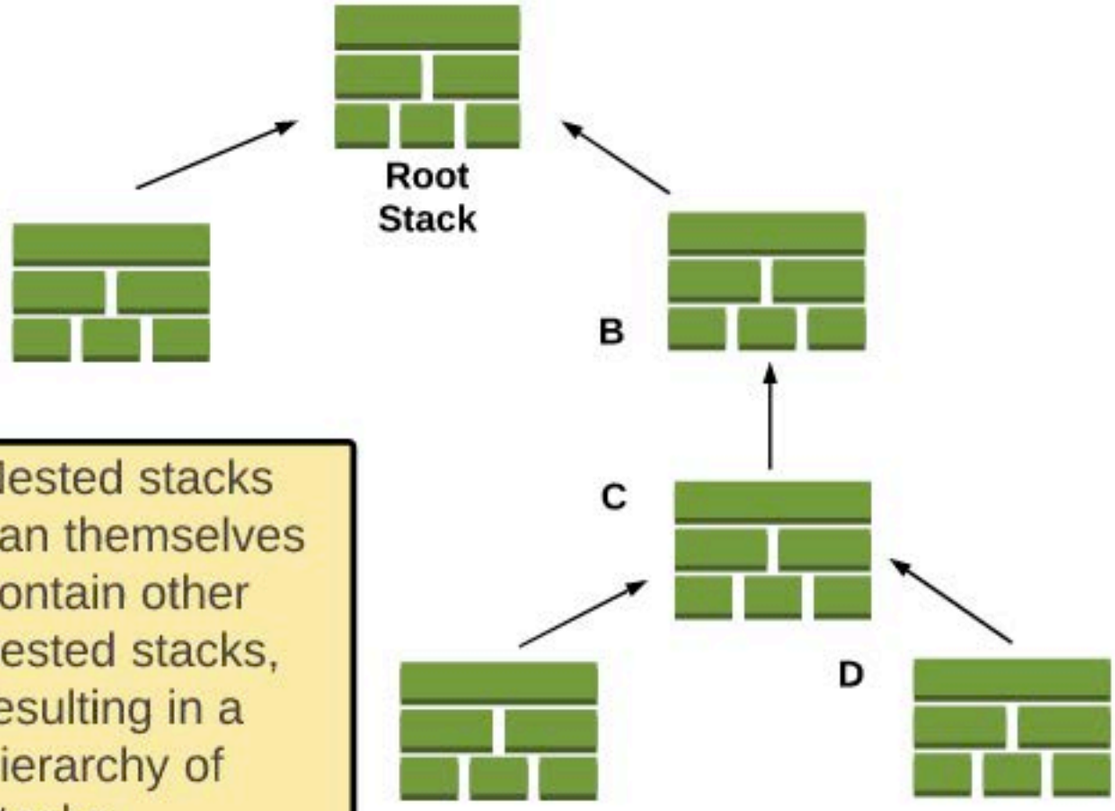
- **GOAL** :Your template completely and precisely specifies your infrastructure and you can rest assured that you can use it to create a fresh set of resources at any time. Disaster Recovery is a good example of the need to be able to trust your templates.
- But what if your stack resources are changed external to CloudFormation? Example: Someone changes Security Group ingress or an EC2 instance type.
- CloudFormation Drift Detection can be used to detect and alert you to changes made to your stack resources. Basically, if your template no longer matches the charateristics of the resources in your stack, Drift Dtection will point this out. From there, you can take steps to get your stack to match your template.

Choose a Topic

Back

AWS

Template Architecture - Nested Stacks



Nested stacks can themselves contain other nested stacks, resulting in a hierarchy of stacks.

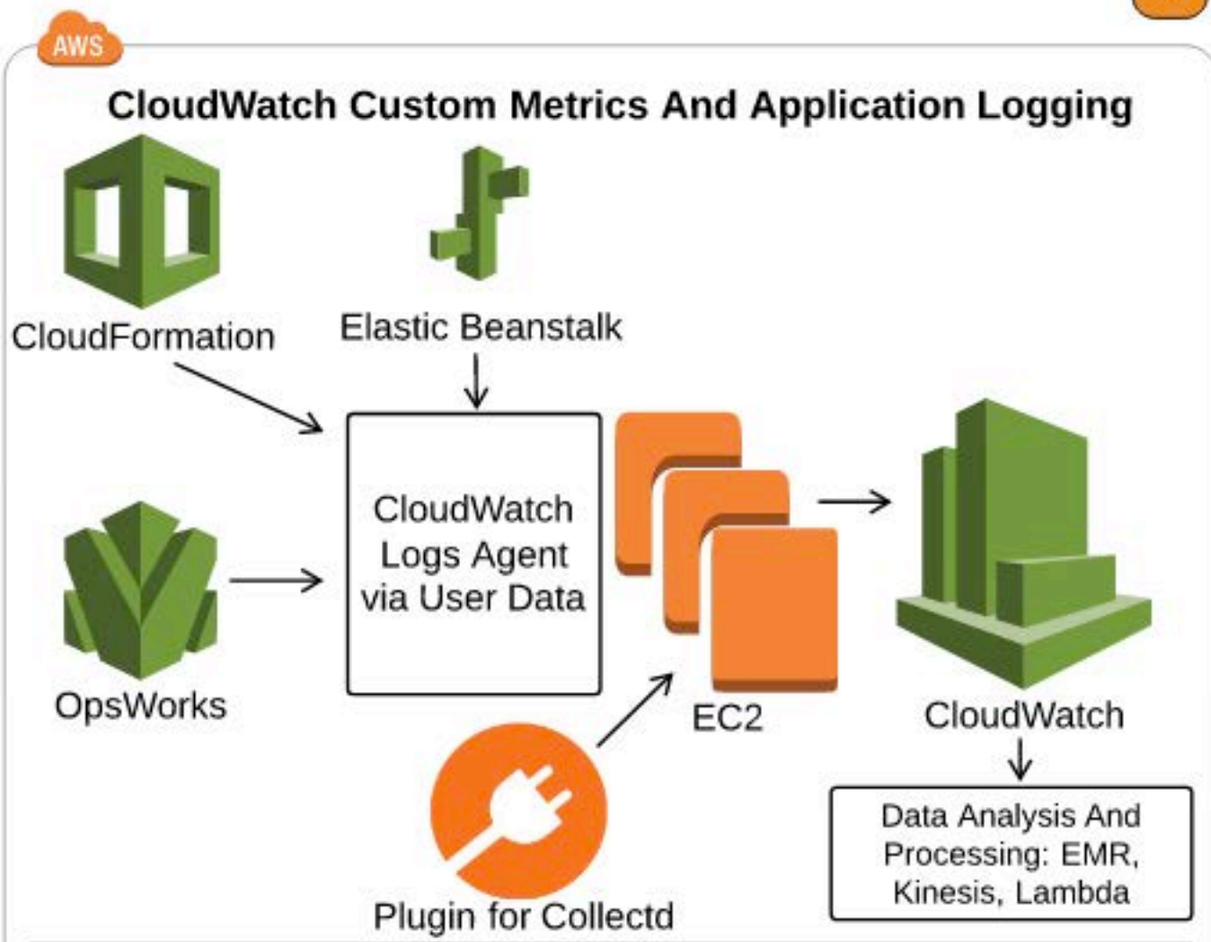
Nested Stacks to Create Reusable Templates

- Organize Stacks by Lifecycle and Ownership
- Use Nested Stacks to Reuse Common Template Patterns
- Create dedicated templates for the common patterns and reuse them by nesting your stacks.
- To create nested stacks, use the AWS::CloudFormation::Stack resource in your template to reference other templates.
- The Root Stack is the top level stack to which all nested stacks ultimately belong.

Choose a Topic

Application Deployments With CloudFormation

X



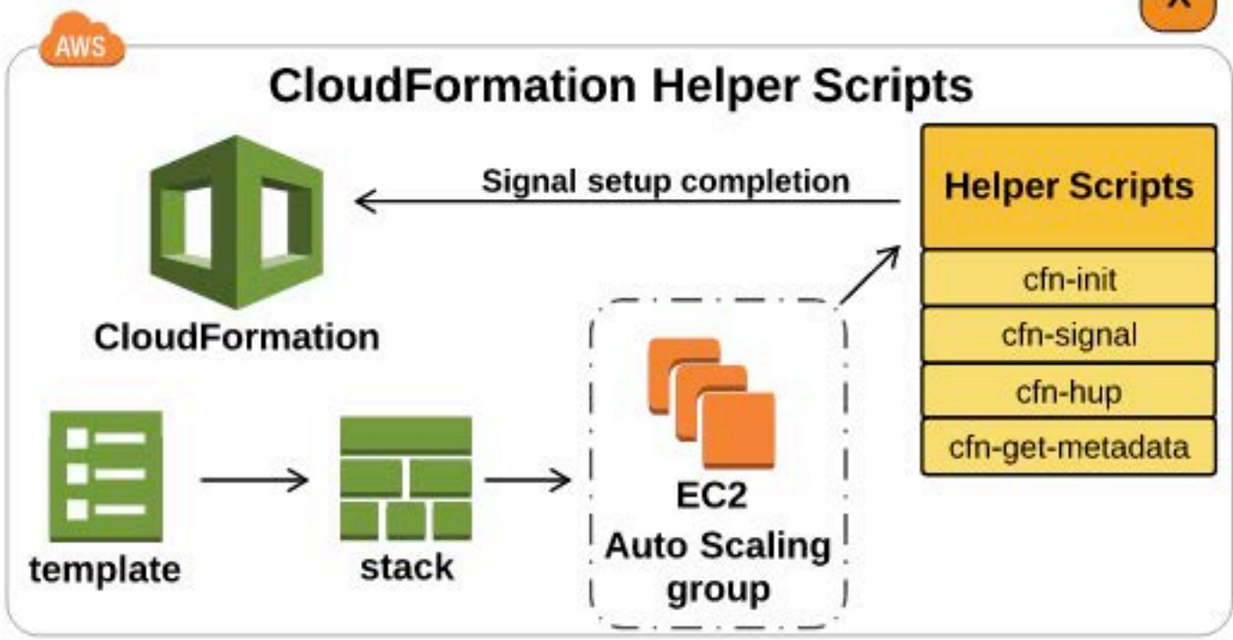
Custom Metrics And Logging

- Provide the flexibility to publish custom data to CloudWatch.
- Beneficial for troubleshooting and creating alarms.
- Options to install and configure custom metrics:
  - Install the CloudWatch Logs Agent on existing instances. Install the agent using OpsWorks, CloudFormation, or Elastic Beanstalk.
  - Use the API, CLI, or SDKs to install the Logs Agent or Collectd.
- OpsWorks as an example, steps:
  - 1) Install the agent 2) Configure the agent (specify which log file to monitor on each EC2 instance, specify where to send logs) 3) Make sure the agent is running.
- Search and Filter Metric Data with Metric Filters which have 4 key elements:
  - Filter Pattern, Metric Name, Metric Namespace, Metric Value (example: to count 404s, we could use a value of 1 for each 404 found).

Choose a Topic

Application Deployments With CloudFormation

X



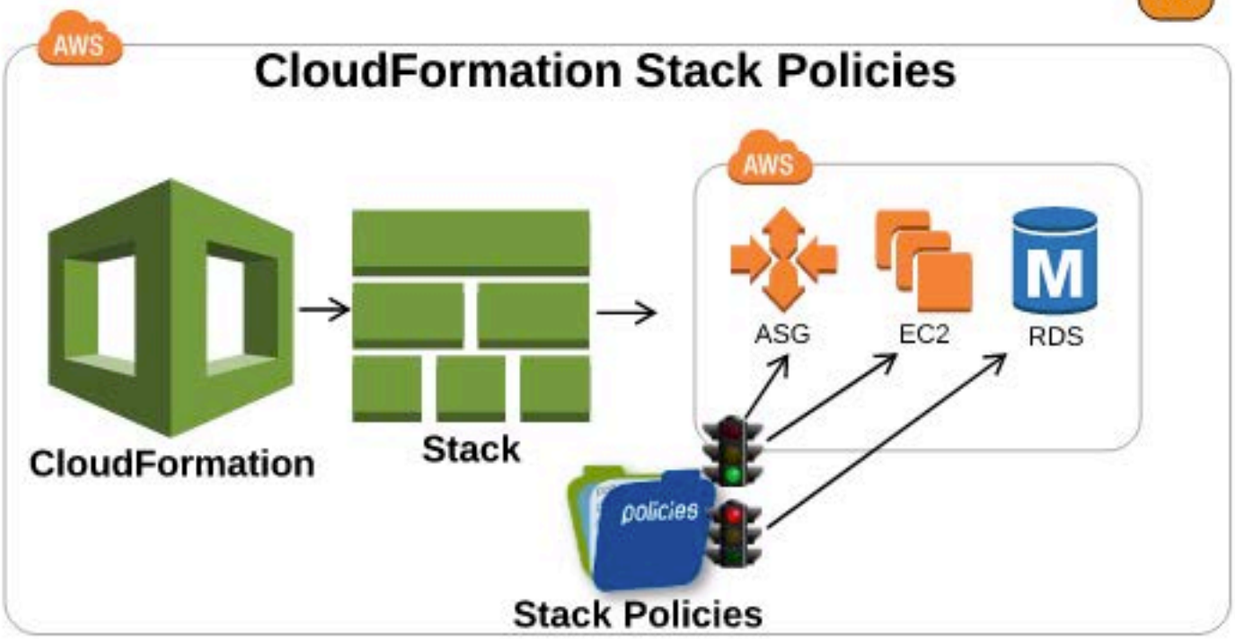
CloudFormation Helper Scripts

- Helper Scripts can be used to:
  - Send signals back to a stack
  - Configure and bootstrap instances
  - Update instances
- How to access Helper Scripts:
  - They are pre-installed on the latest version of the Amazon Linux AMI (in /opt/aws/bin)
  - Also available from the Amazon Linux yum repository
  - For Windows 2008 or later, you can install them with python
- Helper Scripts:
  - **cfn-init** - Installs packages, create and write files to disk, start/stop services
    - required options - stack and resource
  - **cfn-signal** - used to signal back to a stack success or failure
  - **cfn-hup** - A daemon that detects changes in resource metadata and runs actions when a change is detected.
  - **cfn-get-metadata** - Used to get a metadata block from CloudFormation and print it out to standard output:
    - cfn-get-metadata --access-key access.key\ --secret-key secret.key\ --credential-file | f credential.file\ --key | k key\ --stack stack.name.or.id\ --resource | -r logical.resource.id

Choose a Topic

Application Deployments With CloudFormation

X



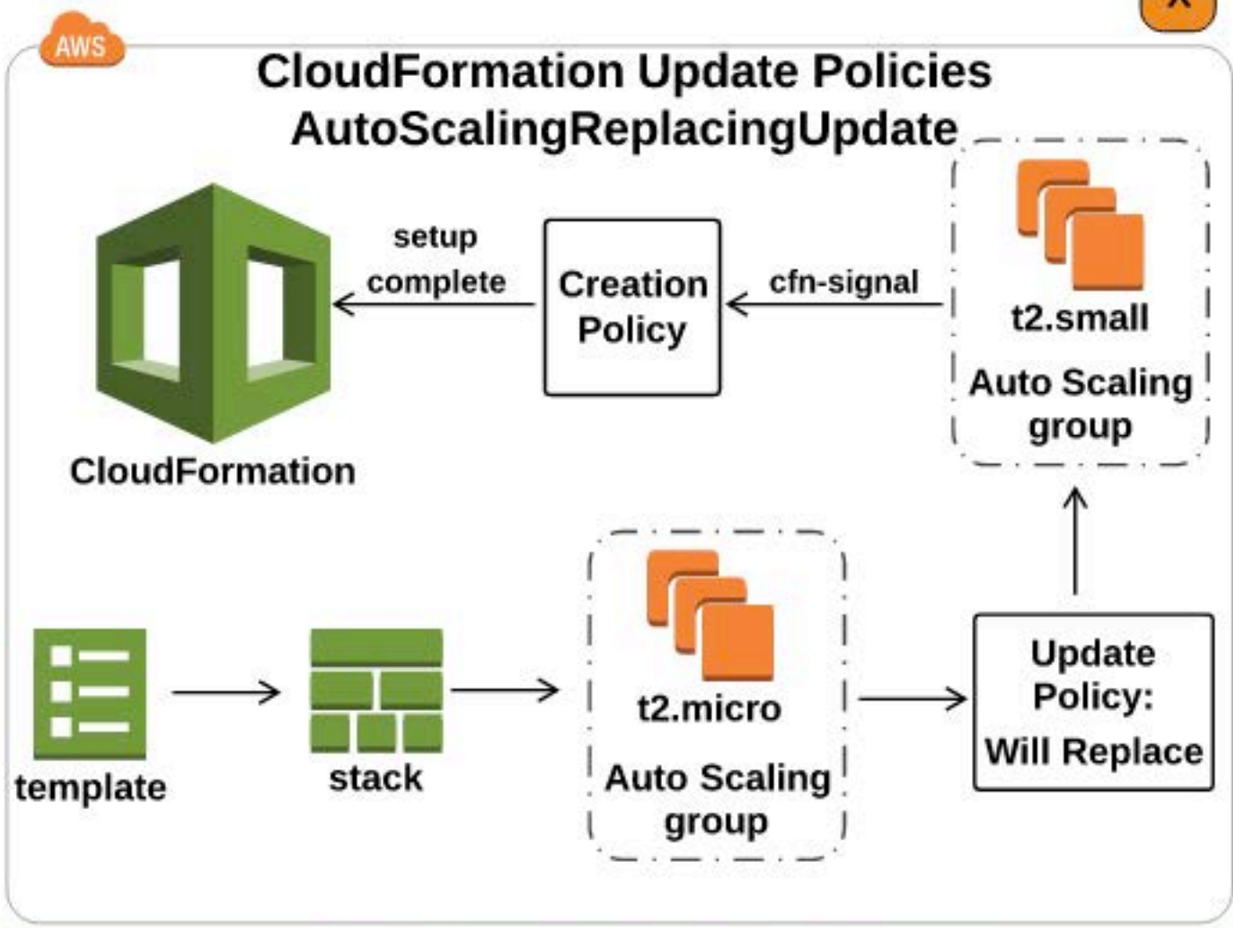
CloudFormation Stack Policies

- Stack Policies are used to control which resources can be updated, and by what actions.
- By default, all actions are allowed on all resources.
- When setting a stack policy, all resources become protected by default. We must explicitly allow an action on a resource.
- Policies can be created at stack creation time, or applied to an existing stack.
- We can only have 1 stack policy per stack, but one policy can have multiple statements.
- Policies have:
  - Effect, Action, Principal, Resource, Condition
- Action options:
  - Update:Modify, Update:Replace, Update:Delete, Update:*
- How can we update protected resources?
  - We can use an overriding stack policy
    - This policy is temporary - only for that specific update
    - We specify it at update time (via the console or CLI)
    - The overriding policy needs to have an allow statement for the resources we want to update.

Choose a Topic

Application Deployments With CloudFormation

X

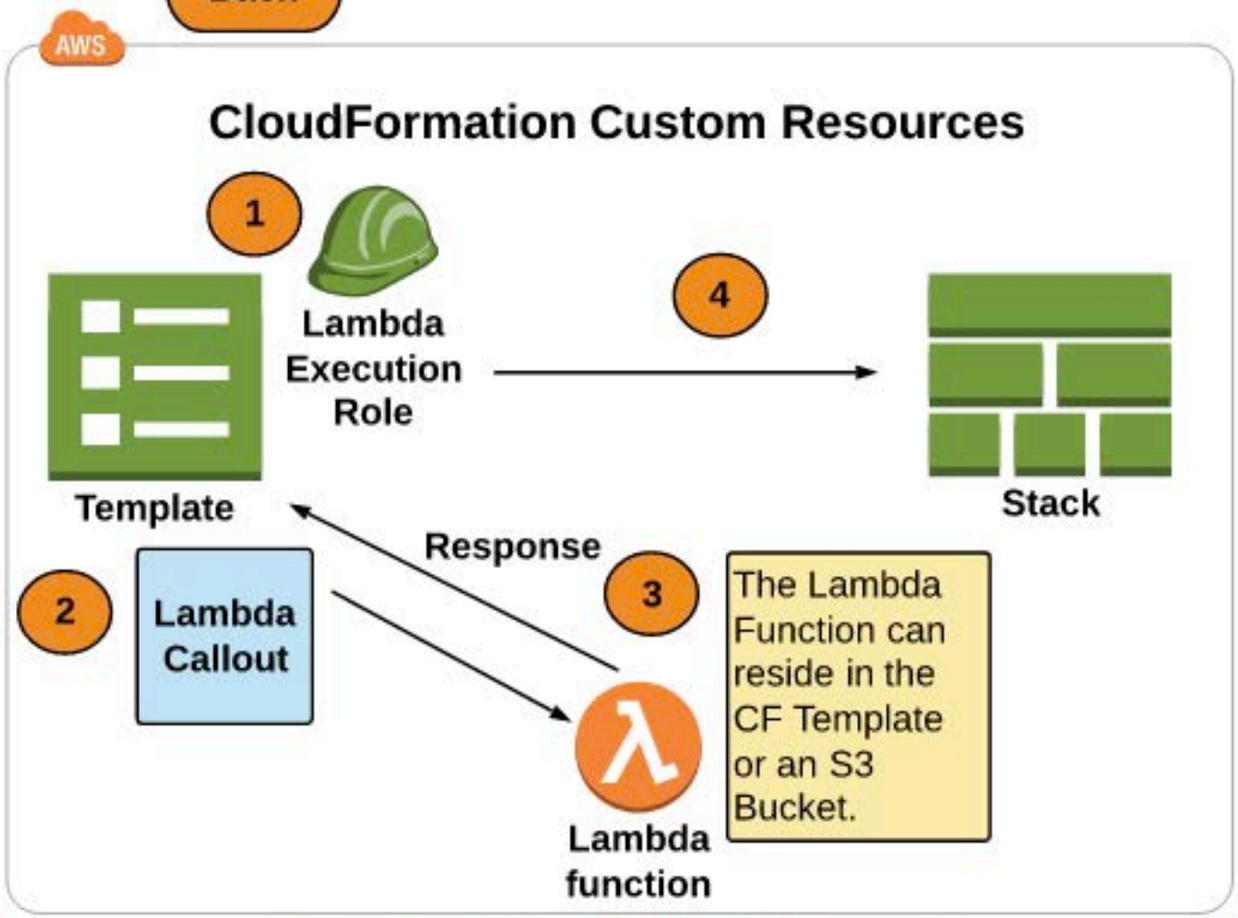


CloudFormation Update Policies

- How can we update instances through Auto Scaling Groups with minimal downtime?
- How can we update ASG Launch Configurations and have those changes update all instances under the ASG?
- CloudFormation **UpdatePolicy** - can describe how instances and ASGs are updated depending on the policy that we configure.
  - AutoScalingReplacingUpdate
  - AutoScalingRollingUpdate
  - AutoScalingScheduledAction
- **AutoScalingReplacingUpdate** and **AutoScalingRollingUpdate** apply when we make changes to the Launch Configuration, the ASGs subnets, or when an update includes instances that don't match the current Launch Configuration.
- **AutoScalingScheduledAction** applies when we update a stack that includes an Auto Scaling group with an associated scheduled action.

Choose a Topic

Back



CloudFormation Custom Resources

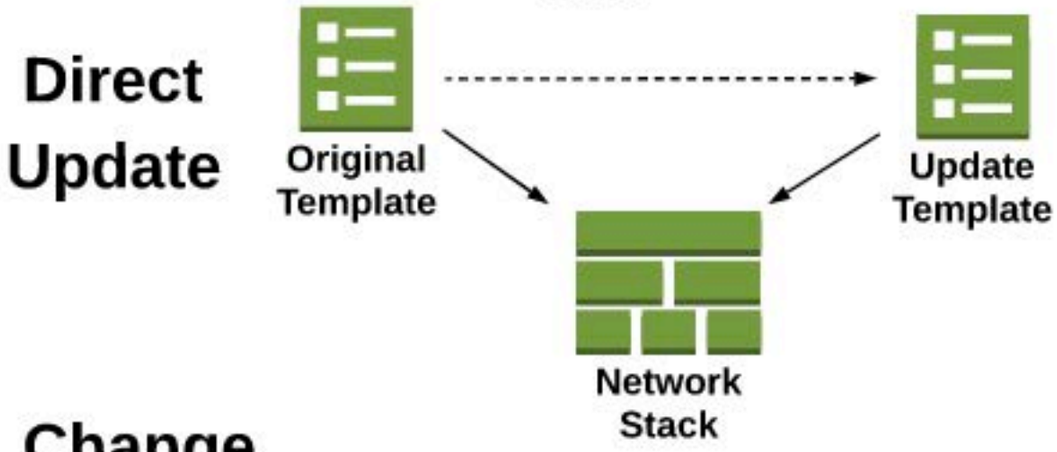
- 3 Actors: Template Developer, Custom Resource Provider, CloudFormation
- Steps for Configuring a Custom Resource in a CloudFormation Template:
  - Create Lambda Execution Role
  - Add the Lambda Function inline to the CloudFormation Template or reference it's location in an S3 Bucket.
  - Create the Lambda Callout which calls the Lambda function. Variables are passed to the Lambda function.
  - Create the response: This is what is returned from the Custom Resource (Lambda Function). The status of the request is returned as well as the actual data.

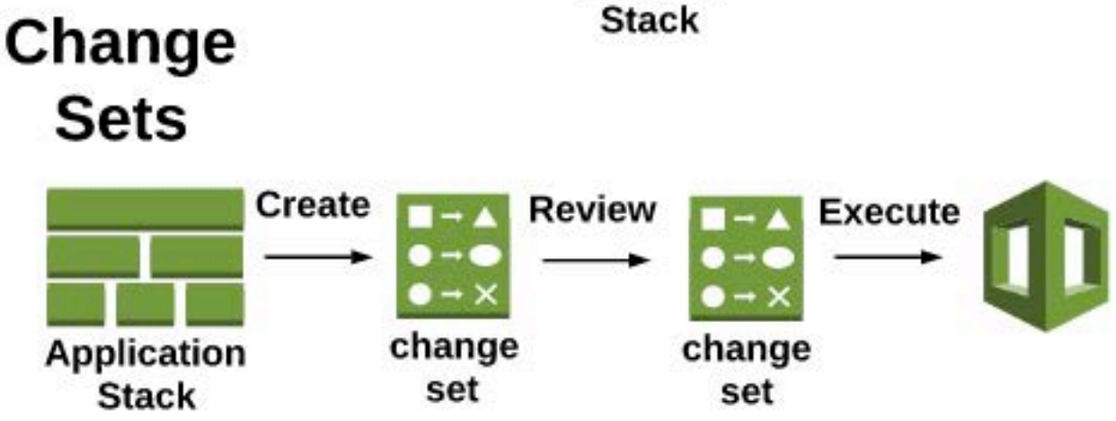
Choose a Topic

Back

AWS

CloudFormation Stack Updates And Change Sets





CloudFormation Stack Updates

- Two ways: Direct Updates or Change Sets
- When updating a stack, CloudFormation might interrupt resources or replace updated resources, depending on which properties you update.
  - Update Behaviors: Update with No Interruption, Update with Some Interruption, Replacement.
- With change sets, you can preview the changes AWS CloudFormation will make to your stack, and then decide whether to apply those changes.