Controlling the Clouds

An Introduction to AWS CloudFormation

Who are you listening to?

Murdock (Chris) - DevOps Manager at Conga

20 years in IT, 18 months of focus on AWS

murdock@getconga.com

@generic42

https://www.linkedin.com/in/murdock42

We are hiring! - https://getconga.com/about/careers/

Conga like the drum? Or the dance line?

Conga is a document generation solution that integrates with your Salesforce environment to enable the usage of templates for Word/Excel/PowerPoint/PDF document creation.

From Marketing:

Conga's suite of solutions creates more efficient organizations by simplifying and automating data, documents, contracts and reporting. As the provider of the #1 paid application on the Salesforce AppExchange, we have more than a decade of experience increasing the value of the Sales Cloud by removing systems and process pain points that impede the customer lifecycle. Our 8000+ customers are passionate about our platform and support giving us 5 stars on the Salesforce AppExchange.

Stats; like those tests in highschool but less interesting

	2015	2017
Documents/month (millions)	1	4.5
Production deployments per month	1	4 (limited)
Beta Environments	0	10
Deployment time (minutes)	90	9

From 90 to 9 - “Take your stinking paws off me you damn dirty admin!”

- What is AWS CloudFormation?
- Primer - Infrastructure as Code, Immutable Infrastructure, Orchestration
- CloudFormation Formats
- CloudFormation Template Sections
- CloudFormation Designer
- Basic Sample Template
- Complex Sample Template
- Live Demo (This is never a bad idea!)
- Gotchas
- Resource List

What is AWS CloudFormation?

From Amazon - AWS CloudFormation gives developers and systems administrators an easy way to create and manage a collection of related AWS resources, provisioning and updating them in an orderly and predictable fashion.

From Me - A clean (sometimes) way of defining AWS resources, connecting those and deploying them in a repeatable fashion.

Primer - LMGTFY!

Infrastructure as Code - is the process of managing and provisioning computer data centers through machine-readable definition files, rather than physical hardware configuration or interactive configuration tools.

Immutable infrastructure - is an approach to managing services and software deployments on IT resources wherein components are replaced rather than changed. An application or services is effectively redeployed each time any change occurs

Orchestration - in this sense is about aligning the business request with the applications, data, and infrastructure. It defines the policies and service levels through automated workflows, provisioning, and change management.

CloudFormation Formats - YAML, I never knew I loved you

JSON	YAML

{
  "Parameters" : {
    "KeyName" : {
      "Description" : "JSON is messy and finicky",
      "Type" : "AWS::EC2::KeyPair::KeyName"
    }
  }
}
	Parameters:
	KeyName:
	   Description: YAML looks cleaner and is easier to maintain
	   Type: AWS::EC2::KeyPair::KeyName

CloudFormation Template Sections - Nothing clever here

Parameters - Variables and AWS specific items that get passed into the Resources e.g. Stack owner, Environment, Key

Mappings - Arrays that can be queried in Conditions or Resources e.g. Security Groups, Subnets, AMI, Instance Size

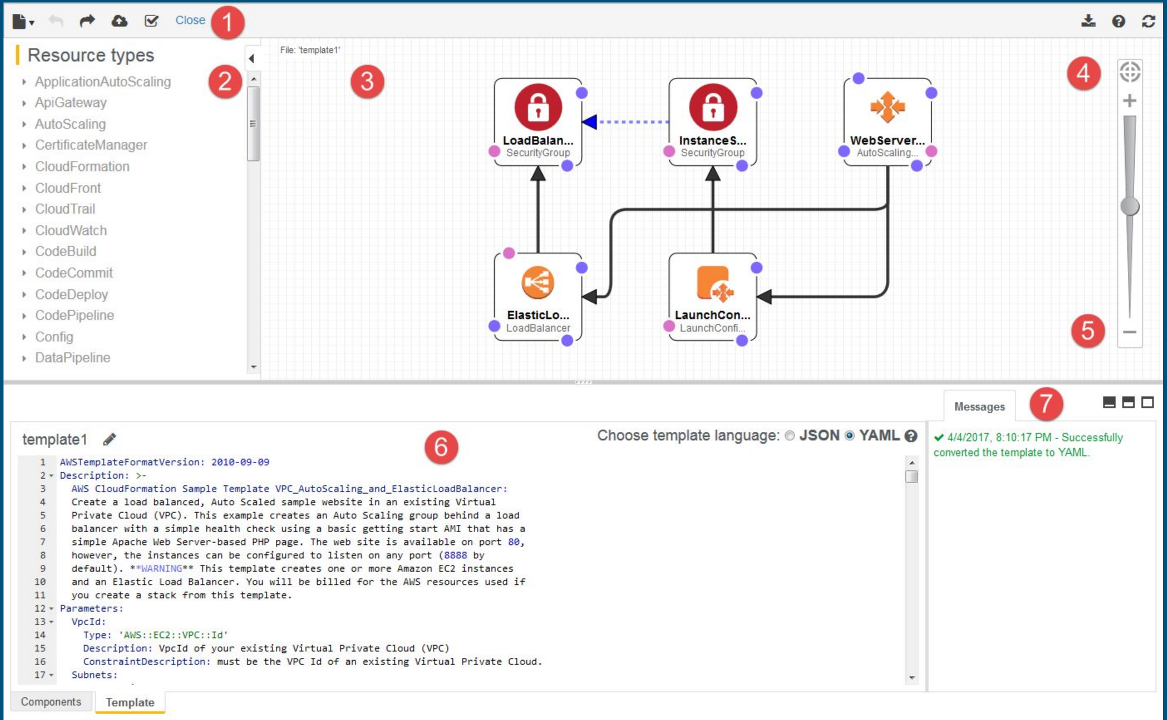
Conditions - Allows the creation of resources based upon Intrisic Functions within the stack e.g. Fn::And, Fn::If, Fn::And, Fn::Not, Fn::Equals, Fn::Or

Resources - Definitions of the AWS service component that should be instantiated.

Outputs - Information about the stack itself, queryable against resources defined above.

CloudFormation Designer

1. Toolbar
2. Resource Types
3. Canvas
4. Fit to window
5. Full vs Split
6. Inline JSON/YAML Editor
7. You screwed up



Choose template language: ○ JSON ● YAML

```
AWSTemplateFormatVersion: 2010-09-09
Description: >-
  AWS CloudFormation Sample Template VPC_AutoScaling_and_ElasticLoadBalancer:
  Create a load balanced, Auto Scaled sample website in an existing Virtual
  Private Cloud (VPC). This example creates an Auto Scaling group behind a load
  balancer with a simple health check using a basic getting start AMI that has a
  simple Apache Web Server-based PHP page. The web site is available on port 80,
  however, the instances can be configured to listen on any port (8888 by
  default). **WARNING** This template creates one or more Amazon EC2 instances
  and an Elastic Load Balancer. You will be billed for the AWS resources used if
  you create a stack from this template.
Parameters:
  VpcId:
    Type: 'AWS::EC2::VPC::Id'
    Description: VpcId of your existing Virtual Private Cloud (VPC)
    ConstraintDescription: must be the VPC Id of an existing Virtual Private Cloud.
  Subnets:
```

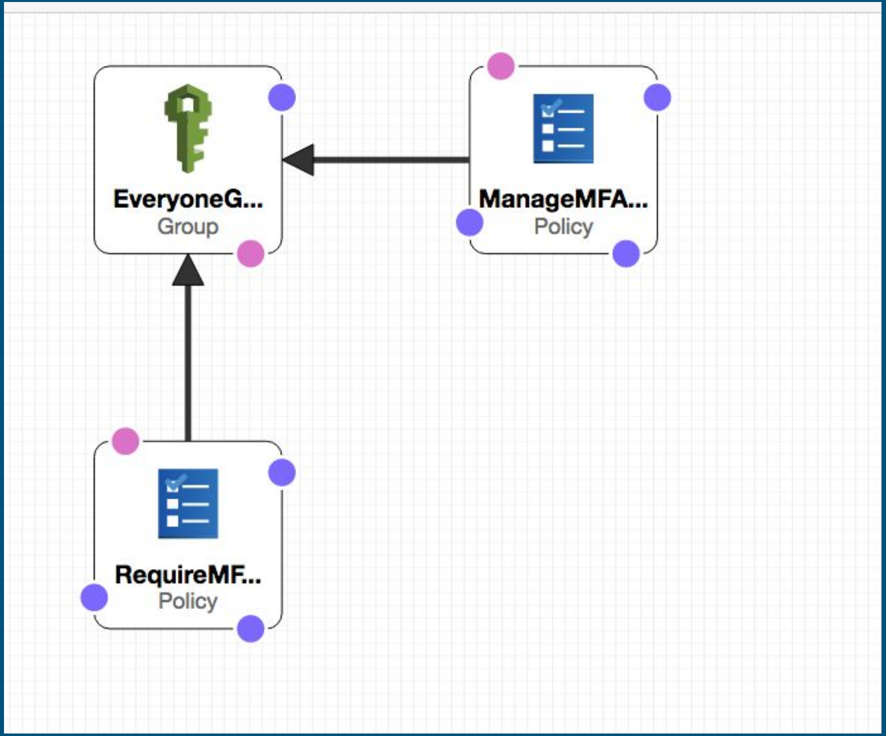
✓ 4/4/2017, 8:10:17 PM - Successfully converted the template to YAML.

Basic sample template - Oh yea, I can do this!

Goal: Require MFA for all IAM users and allow them to add their own MFA device.

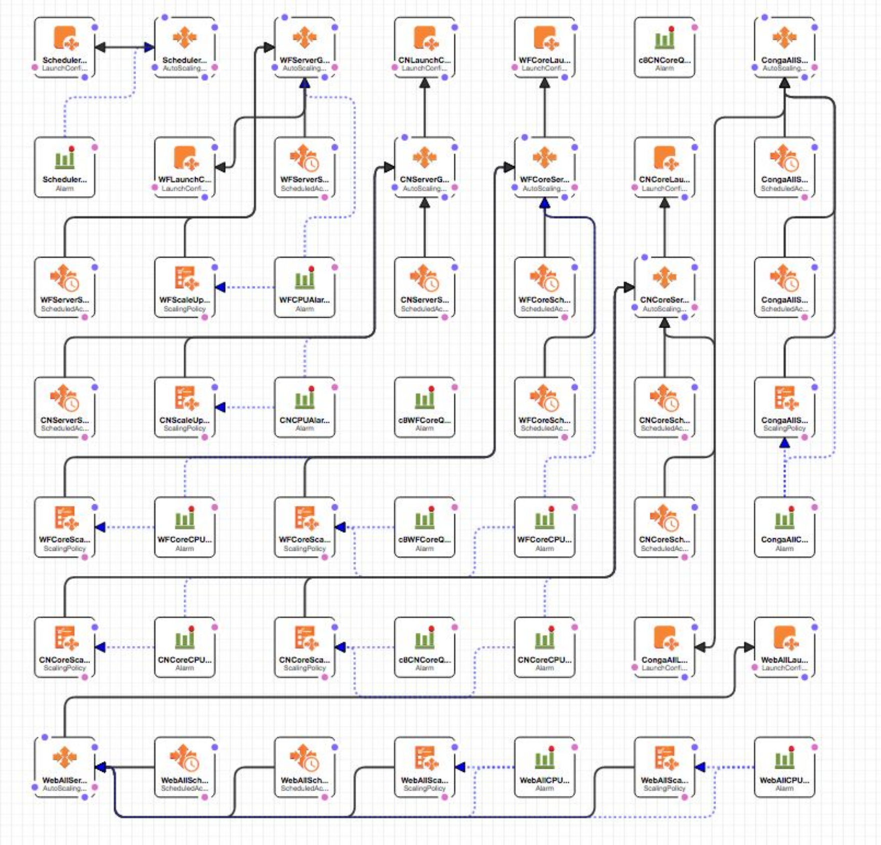
Solution: Create “Everyone” IAM Group and attach two IAM Policies “ManageMFADevice” and “RequireMFADevice”

*Thanks to Roger Sigs for these policy outlines



Complex sample template - What have I done?

- 1200 lines
- Builds Beta, Staging or Production
- Generates
  - AutoScaling Groups (x6)
    - Scheduled scaling actions
    - Response scaling action
  - Elastic Load Balancers (3 external 2 internal)
  - Route53 DNS Entries
  - CloudWatch Alarms



Scheduler... LaunchConfi...
Scheduler... AutoScaling...
WFServerG... AutoScaling...
CNLaunchC... LaunchConfi...
WFCoreLau... LaunchConfi...
c8CNCoreQ... Alarm
CongaAllS... AutoScaling...
Scheduler... Alarm
WFLaunchC... LaunchConfi...
WFServerS... ScheduledAc...
CNServerG... AutoScaling...
WFCoreSer... AutoScaling...
CNCoreLau... LaunchConfi...
CongaAllS... ScheduledAc...
WFServerS... ScheduledAc...
WFScaleUp... ScalingPolicy
WFCPUAlar... Alarm
CNServerS... ScheduledAc...
WFCoreSch... ScheduledAc...
CNCoreSer... AutoScaling...
CongaAllS... ScheduledAc...
CNServerS... ScheduledAc...
CNScaleUp... ScalingPolicy
CNCPUAlar... Alarm
c8WFCoreQ... Alarm
WFCoreSch... ScheduledAc...
CNCoreSch... ScheduledAc...
CongaAllS... ScalingPolicy
WFCoreSca... ScalingPolicy
WFCoreCPU... Alarm
WFCoreSca... ScalingPolicy
c8WFCoreQ... Alarm
WFCoreCPU... Alarm
CNCoreSch... ScheduledAc...
CongaAllC... Alarm
CNCoreSca... ScalingPolicy
CNCoreCPU... Alarm
CNCoreSca... ScalingPolicy
c8CNCoreQ... Alarm
CNCoreCPU... Alarm
CongaAllL... LaunchConfi...
WebAllLau... LaunchConfi...
WebAllSer... AutoScaling...
WebAllSch... ScheduledAc...
WebAllSch... ScheduledAc...
WebAllSca... ScalingPolicy
WebAllCPU... Alarm
WebAllSca... ScalingPolicy
WebAllCPU... Alarm

Live Demo - When you see me sweat

To the cloud!

Gotchas - Why won’t you f%$^ing work!

- CloudFormation Designer
  - Focuses on proper JSON/YAML formatting
  - Doesn’t validate stack names
  - Doesn’t check IAM Roles
  - Ignores parameters and conditions when validating
- CloudFormation
  - Fails with obscure error messages (or none at all)
  - Automatically rolls back a stack, clearing error messages - Advanced Settings - Rollback on failure, No
- AWS resources
  - Elasticache nodes/clusters take a long time to spin up and subsequently delete

Resources - I swear I didn’t make this stuff up

Getting multiple outputs of out nested stacks -
https://www.linkedin.com/pulse/getting-multiple-outputs-out-cloudformation-nested-stacks-murdock

Passing parameters to nested stacks -
http://www.linkedin.com/pulse/aws-cloudformation-passing-parameters-nested-stacks-murdock

Building with unknowns -

https://www.linkedin.com/pulse/cloudformation-building-unknowns-aws-christopher-murdock