Infrastructure as Code: CloudFormation Best Practices

AWS Summit Berlin 2016

Matthias Jung, Solutions Architect

April, 12th, 2016
Agenda

• Why CloudFormation?
• How to plan my stacks?
• How to get started?
• How to prevent errors?
• How to safely update stacks?
• How to extend CloudFormation?
Why CloudFormation?
Setting Up an Application

Series of Operational Tasks

Configure Network & Firewalls
Setup Load Balancer
Configure Servers
Setup Database
Configure Access Rights
...

Setting Up an Application

Series of API Calls to AWS

- Configure VPC
- Launch ELB
- Launch EC2 Instances
- Launch RDS Instance
- Define IAM Users
- ...

AWS
Setting Up an Application

Configure VPC
Launch ELB
Launch EC2 Instances
Launch RDS Instance
Define IAM Users
...

Series of API Calls to AWS

AWS CLI & SDKs
Setting Up an Application

Template of Resources

VPC
ELB
EC2 Instances
RDS Instance
IAM Users
...

AWS
Anatomy of a CloudFormation Template

- Parameters
- Mappings
- Conditions
- Resources
- Output

JSON Template
Key Benefits

- Automation
- Reuse & Sharing
- Infrastructure as Code
- Atomicity
- Modular
- Configurable
- Integrated
- Start Quickly
How to plan my stacks?
Organize by Layers

Frontend Services
- Consumer Website, Seller Website, Mobile Backend

Backend Services
- Search, Payments, Reviews, Recommendations

Shared Services
- CRM DBs, Common Monitoring/Alarms, Subnets, Security Groups

Base Network
- VPCs, Internet Gateways, VPNs, NATs

Identity
- IAM Users, Groups, Roles
Organize by Environments

Layers of stacks

Environments

Dev1, Dev2, Integ, Prod
Think Services & Decouple

```
"Parameters" : {
    "CustDBEndPoint"
}

"Outputs" : {
    "CustDBEndPoint"
}
```
Reuse

Website1

“Resources” : {
   “ELB”,
   “AutoScaling”,
   “RDS”
}

Website2

“Resources” : {
   “ELB”,
   “AutoScaling”,
   “DynamoDB”
}
Website 1

"Resources" : {
  "ELB",
  "AutoScaling",
  "RDS"
}

Website 2

"Resources" : {
  "ELB",
  "AutoScaling",
  "DynamoDB"
}
Nested stacks

Reuse

Frontend

"Resources": {
  "ELB",
  "AutoScaling"
}

Backend1

"Resources": {
  "NestedStack",
  "RDS"
}

Backend2

"Resources": {
  "NestedStack",
  "DynamoDB"
}
Nested stacks

Role Specialization

Frontend

```
"Resources": {
  "ELB",
  "AutoScaling"
}
```

Backend1

```
"Resources": {
  "NestedStack",
  "RDS"
}
```

Backend2

```
"Resources": {
  "NestedStack",
  "DynamoDB"
}
```
How to get started?
# Start with Existing Template

## Amazon Virtual Private Cloud

<table>
<thead>
<tr>
<th>Template Name</th>
<th>Description</th>
<th>View</th>
<th>View in Designer</th>
<th>Launch</th>
</tr>
</thead>
<tbody>
<tr>
<td>A single Amazon EC2 in an Amazon VPC</td>
<td>Creates a VPC and adds an Amazon EC2 instance with an Elastic IP address and a security group.</td>
<td>View</td>
<td>View in Designer</td>
<td>[Launch Stack]</td>
</tr>
<tr>
<td>Amazon VPC with static routing to an existing VPN</td>
<td>Creates a private subnet with a VPN connection that uses static routing to an existing VPN endpoint.</td>
<td>View</td>
<td>View in Designer</td>
<td>[Launch Stack]</td>
</tr>
<tr>
<td>Autoscaling and load-balancing website in an Amazon VPC</td>
<td>Creates a load balancing, auto scaling sample website in an existing VPC.</td>
<td>View</td>
<td>View in Designer</td>
<td>[Launch Stack]</td>
</tr>
<tr>
<td>Amazon VPC with DNS and public IP addresses</td>
<td>Creates a VPC with DNS support and public IP addresses enabled.</td>
<td>View</td>
<td>View in Designer</td>
<td>[Launch Stack]</td>
</tr>
<tr>
<td>Publicly accessible Amazon EC2 instances that are in an Auto Scaling group</td>
<td>Creates a load balancing, autoscaling group with instances that are directly accessible from the Internet.</td>
<td>View</td>
<td>View in Designer</td>
<td>[Launch Stack]</td>
</tr>
<tr>
<td>Amazon EC2 with</td>
<td>Creates an Amazon EC2</td>
<td>View</td>
<td>View in Designer</td>
<td>[Launch Stack]</td>
</tr>
</tbody>
</table>

On this page:

- Auto Scaling
- Amazon DynamoDB
- Amazon EC2
- Amazon ElastiCache
- AWS Elastic Beanstalk
- Elastic Load Balancing
- AWS Identity and Access Management
- AWS OpsWorks
- Amazon Relational Database Service
- Amazon Redshift
- Amazon Route 53
- Amazon Simple Storage Service
- Amazon Simple Queue Service

CloudFormer

AWS CloudFormation gives you an easier way to create a collection of related AWS resources (a stack) by describing your requirements in a template. To create a stack, fill in the name for your stack and select a template. You may choose one of the sample templates to get started quickly, or one of your own templates stored in S3 or on your local hard drive.

Stack Name:
MyCloudFormerStack

Template:
- Use a sample template
- Upload a Template File
- Provide a Template URL

Show Advanced Options
- Notifications (optional):
  - Amazon SNS (no notification)
  - Amazon CloudWatch
  - Creation Timeout (minutes): 10
  - Rollback on Failure: Yes/No

Welcome to the AWS CloudFormation template creation utility. This utility helps you to create a CloudFormation template from the existing resources currently running in your account using a few simple steps. While the created template is complete and can be launched as a CloudFormation stack, it is a starting point for further customizations. You should consider the following:

- Add Parameters to enable stacks to customize to the specific environment.
- Add Mapping to allow the template to be customized to the specific environment.
- Replace static values with "Ref" and "GetAtt" functions to refer property values between resources where the value of one property is dependent on the value of a property from a different resource.
- Use CloudFormation metadata and -on-failure helper scripts to deploy, fix, package, and run commands on your Amazon EC2 instances.
- Provide the following for your Amazon RDS database instance (name and master password).
- Customize or add more outputs to let important information needed by the stack user.

Select the AWS Region: US East (Virginia)
When you press "Create template" we will analyze all of your AWS resources in your account. This may take a little time.

What's New?
- Support for Amazon S3 resources:
  - Support Amazon CloudWatch Logs, Amazon SNS, Amazon CloudFront, Amazon S3, Amazon DynamoDB, Amazon S3, Amazon S3 Object Storage, and Amazon S3 Policy
- Support for Amazon S3 buckets
- Support for Amazon S3 bucket policies
- Support for Amazon S3 bucket policy

Known Issues:
- Amazon RDS database instances in a VPC are not associated with VPC security groups. You will need to manually add these to your template once it is created.

For more information on how to build a template see the AWS CloudFormation User Guide. You can also check out our sample templates demonstrating various template features.

By default, the account credentials will be used; this means that any CloudFormation stack created, however, they can be overridden by setting 

---

Amazon EC2 Elastic IP Addresses

Amazon EC2 Instances

Select/Unselect all Amazon EC2 Instances

---

AWS CloudFormation Template Details:
- **Stack Name**: MyCloudformer
- **Region**: US East (Virginia)
- **Template**: CloudFormer
- **Description**: AWS CloudFormation template for creating a stack with Amazon EC2 instances.
- **Options**: Select/Unselect all Amazon EC2 Instances
- **Template URL**: S3 Bucket
Pick an IDE

It's JSON!
=> Emacs, notepad, vi
Code Generators

**cf_factory 0.0.5**

Cf-factory is a Ruby library to generate CloudFormation templates.

`INSTALL > gem install cf_factory`

**GitHub**

This repository: Search

**cloudtools / troposphere**

troposphere - Python library to create AWS CloudFormation descriptions
CloudFormation Designer

AWS::EC2::Volume (Property: Volumes)

```
{
   "Resources": {
      "I4NOSY": {
         "Type": "AWS::EC2::Instance",
         "Properties": {
            "InstanceType": "t2.medium",
            "DisableAutoTermination": "true",
            "Monitoring": "true"
         },
         "Volumes": [
            {
               "VolumeId": {
                  "Ref": "VOLVQOQ"
               }
            },
            {
               "VolumeId": 
            }
         ]
      }
   }
}
```
How to prevent errors?
Add Comments

{
   "Description" : "This is a sample template.",

   "Resources" : {
      "Bucket98004" : {
         "Type" : "AWS::S3::Bucket",
         "Metadata" : {
            "Comment" : "Image bucket for ZIP code 98004",
            "Version" : "1.2.1_1"
         }
      }
   }
}
Validate your Templates

- JSON Syntax
- Circular Dependencies
- Template Structure

validate-template

Description

Validates a specified template.

Synopsis

```
validate-template
[--template-body <value>]
[--template-url <value>]
[--cli-input-json <value>]
[--generate-cli-skeleton]
```
Use Parameter Types

"Parameters" : {
    "aVpcId" : {
        "Type" : "AWS::EC2::VPC::Id"
    },
    "bSubnetIds" : {
        "Type" : "List<AWS::EC2::Subnet::Id>"
    },
    "cSecurityGroups" : {
        "Type" : "List<AWS::EC2::SecurityGroup::Id>"
    }
}
Use Parameter Types

Specify Parameters

Specify values or use the default values for the parameters that are associated with your AWS CloudFormation template.

Parameters

aVpc  
vpc-ea814e8f (10.0.0.0/16)  
VpcId of your existing Virtual Private Cloud (VPC)

bSubnets

- subnet-e85150ae
- subnet-f89b419d
- subnet-70b1ef36
- subnet-807791f7

The list of SubnetIds in your Virtual Private Cloud (VPC)

cSecurityGroups

- web4-WebServerSecurityGroup-1D7AQ98RDR3SK
- web1-WebServerSecurityGroup-JJXAZ2723AL9
- sp6-SharePointFoundationSecurityGroup-O788PH7WGC34
Use Parameter Constraints

"Parameters" : {
    "SourceCIDRForRDP" : {
        "Description" : "CIDR block to allow RDP from",
        "Type" : "String",
        "MinLength" : "9",
        "MaxLength" : "18",
    }
}
Check IAM Permissions

user -> template -> CloudFormation

VPC

cloud
Check IAM Permissions

1. Permissions to call CloudFormation
Check IAM Permissions

2. Permissions to create resources
Check AWS Limits

# of AWS CloudFormation stacks

# of EC2, RDS, EBS IOPS, etc.

user

template

CloudFormation
How to debug?
View Events

PROMPT> aws cloudformation describe-stack-events --stack-name myteststack

```json
{
  "StackEvents": [
    {
      "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/myteststack/4660",
      "EventId": "af67ef60-0b8f-11e3-8b8a-500150b352e0",
      "ResourceStatus": "CREATE_COMPLETE",
      "ResourceType": "AWS::CloudFormation::Stack",
      "Timestamp": "2013-08-23T01:02:30.070Z",
      "StackName": "myteststack",
      "LogicalResourceId": "myteststack"
    }
  ]
}
```
Debugging Tips

- Deactivate Rollback Flag during tests
- Put “breakpoints” via WaitConditions
- Test user data & scripts separately, e.g. Moustache
- Log stack events in DWH or logging service
- Use CloudTrail and AWS Config to track changes
- Redirect local Cfn log files to CloudWatch Logs
Use CloudWatch Logs for Debugging

![CloudWatch Logs interface](ow.ly/E0zO3)

<table>
<thead>
<tr>
<th>Creation Time</th>
<th>Event Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>2014-07-19 00:05:05 UTC-7</td>
<td>2014-07-19 07:05:02,342 [INFO] Running config cfn-hup-config</td>
</tr>
<tr>
<td>2014-07-19 00:05:05 UTC-7</td>
<td>2014-07-19 07:05:02,437 [INFO] enabled service cfn-hup</td>
</tr>
<tr>
<td>2014-07-19 00:05:05 UTC-7</td>
<td>2014-07-19 07:05:03,638 [INFO] Restarted cfn-hup successfully</td>
</tr>
<tr>
<td>2014-07-19 00:05:05 UTC-7</td>
<td>2014-07-19 07:05:03,704 [INFO] Running config application-config</td>
</tr>
<tr>
<td>2014-07-19 00:05:15 UTC-7</td>
<td>2014-07-19 07:05:14,603 [INFO] Started httpd successfully</td>
</tr>
<tr>
<td>2014-07-19 00:05:15 UTC-7</td>
<td>2014-07-19 07:05:14,974 [INFO] ConfigSets completed</td>
</tr>
</tbody>
</table>
How to protect running stacks?
Protect Stacks from Unintended Changes

Frontend Services
• Consumer Website, Seller Website, Mobile Backend

Backend Services
• Search, Payments, Reviews, Recommendations

Shared Services
• CRM DBs, Common Monitoring /Alarms, Subnets, Security Groups

Base Network
• VPCs, Internet Gateways, VPNs, NATs

Identity
• IAM Users, Groups, Roles
Protect Stacks from Unintended Changes

{
  "Effect": "Deny",
  "Action": [ "cloudformation:*" ],
  "Resource":
}

{
  "Effect": "Allow",
  "Action": [ "cloudformation:*" ],
  "Resource":
    "arn:aws:cloudformation:us-west-2:123456789:stack/FrontEnd*"
}
Protect Stacks from Drift

User → CloudFormation → Template → Resources → VPC

No drift detected.
Protect Resources with IAM and Tags

```json
{
    "Effect": "Deny",
    "Action": [
        "ec2:TerminateInstances"
    ],
    "Condition": {
        "Null": {
            "ec2:ResourceTag/*/cloudformation": "true"
        }
    },
    "Resource": "*"
}
```
Stack Policies Against Unwanted Updates

user → template → CloudFormation

Deny update of database
Stack Policies Against Unwanted Updates

- User creates a template in CloudFormation.
- Attempt to update CloudFormation.
- Deny update of database.
Stack Policies Against Unwanted Updates

```
{
  "Statement": [
    {
      "Effect": "Deny",
      "Action": "Update:*",
      "Principal": "*",
      "Resource": "LogicalResourceId/ProductionDatabase"
    },
    {
      "Effect": "Allow",
      "Action": "Update:*",
      "Principal": "*",
      "Resource": "*"
    }
  ]
}
```

Stack Policies Against Unwanted Updates

“Do not update the databases”

"Effect" : "Deny",
"Principal" : "*",
"Action" : "Update:*",
"Resource" : "*",
"Condition" : {
    "StringEquals" : {
        "ResourceType" : [
            "AWS::RDS::DBInstance",
            "AWS::Redshift::Cluster"
        ]
    }
}

“Okay to update, unless the update requires replacement”

"Effect" : "Deny",
"Principal" : "*",
"Action" : "Update:Replace",
"Resource" : "LogicalResourceId/MyInstance"
How to safely update stacks?
Choose an Update Style

In-place

Templates

Stacks

Blue-Green

Traffic
Choose an Update Style

In-place

Fast, Simple & Cost Efficient

Blue-Green

Robust
Review Updates

What is going to be updated?

- Run Textual Diffs
- Pay attention to impact on Related Resources
  - Ref and Get:Att
- Check for Update Mode
  - No Interruption
  - Some Interruption
  - Replacement
- Check for Drift
Review Updates

What is going to be updated?

• Preview Feature with Change Sets
• Pay attention to impact on Related Resources
  • Ref and Get:Att
• Check for Update Mode
  • No Interruption
  • Some Interruption
  • Replacement
• Check for Drift
Review Impact via Change Sets

CloudFormation > Stack: LAMP-Stack-3 > Change set detail: BigChanges

**BigChanges**

**Overview**
- **ID**: am:aws.cloudformation.us-east-1:34814629041:changeSet/BigChanges/be420e82-5ae4-bc53-ad76-60b8d34c290a
- **Description**: Big changes for my stack
- **Created time**: 2016-03-23 19:09:36 UTC-0700
- **Status**: CREATE_COMPLETE
- **Stack name**: LAMP-Stack-3

**Change set input**

**Changes**
The changes CloudFormation will make if you execute this change set

<table>
<thead>
<tr>
<th>Action</th>
<th>Logical ID</th>
<th>Physical ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add</td>
<td>WebServerAutoScalingGroup</td>
<td></td>
</tr>
<tr>
<td>Remove</td>
<td>WebServerInstance</td>
<td>i-d47ac24f</td>
</tr>
<tr>
<td>Add</td>
<td>WebServerLaunchConfig</td>
<td></td>
</tr>
<tr>
<td>Add</td>
<td>WebSiteLoadBalancer</td>
<td></td>
</tr>
</tbody>
</table>

**Execute change set**

Are you sure you want to execute the **BigChanges** change set to update the **LAMP-Stack-3** stack?
How to extend CloudFormation?
Extend with Stack Events

Stack

provision

Web Analytics Service
Extend with Stack Events

Stack Events

Stack

Web Analytics Service

“Create, Update, Rollback, or Delete”

Worker

Amazon SNS Topic

provision

14:34:34 UTC-0700  CREATE_COMPLETE  Type: AWS:AutoScaling:AutoScalingGroup  Logical ID: WebServerGroup
Extend with Lambda Custom Resources

What you need:

• Lambda function talking with CloudFormation

```javascript
//Sends response to the pre-signed S3 URL
function sendResponse(event, context, responseStatus, responseHeaders, responseBody, error, request) {
  var response = JSON.stringify({
    Status: responseStatus,
    Reason: "See the details in CloudWatch Log Streams",
    PhysicalResourceId: context.logStreamName,
    StackId: event.StackId,
   RequestId: event.RequestId,
    LogicalResourceId: event.LogicalResourceId,
    Data: responseData
  });
  return { status: responseStatus, headers: {'Content-Type': 'application/json'}, body: response };
}
```
Extend with Lambda Custom Resources

What you need:
• Custom resource in CloudFormation

```
"MyCustomResource" : {
  "Type" : "Custom::TestLambdaCrossStackRef",
  "Properties" : {
    "ServiceToken": { "Fn::Join": [ "", [ "arn:aws:lambda:", { "Ref": "AWS::StackName" }, "NetworkStackName"] ] }
  }
}
```
Extend with Lambda Backed Custom Resources
Extend with Lambda Backed Custom Resources

“Create, Update, Rollback, or Delete”
Summary

• Why CloudFormation?
• How to plan my stacks?
• How to get started?
• How to prevent errors?
• How to safely update stacks?
• How to extend CloudFormation?