amazon web services

SECUREKLOUD

19 BEST PRACTICES
FOR CREATING AMAZON CLOUD FORMATION TEMPLATES

INTRODUCTION TO AMAZON VPC

Amazon Virtual Private Cloud (Amazon VPC) enables we to launch Amazon Web Services (AWS) resources into a virtual network that we've defined. Thwas virtual network closely resembles a traditional network that we'd operate in your own datacenter, with the benefits of using the scalable infrastructure of AWS.

19 BEST PRACTICES
FOR CREATING AMAZON CLOUD FORMATION TEMPLATES

Amazon CloudFormation templates are widely used in the AWS cloud for environment creation by the IT and application teams. We have been helping enterprises with Amazon CFT based automation for years and following are some of the best practices to follow while creating Amazon CFT Templates. This article compiles the CFT practices /pointers used by us for provisioning complex large scale environments on AWS, not all of them will be applicable for all use cases.

PRACTICE #1:
VERSION YOUR CLOUDFORMATION TEMPLATES

CloudFormation template should be commenced with a template format version and description such that it could be artifact-able with version control tools like Git, SVN, CVS, etc.,

Example:

```
"AWSTemplateFormatVersion": "2010-09-09",
   "Description": "AWS CloudFormation Template for CustomerXYZ Version 5.0",
```

PRACTICE #2:
USE INPUT PARAMETERS

Input parameters section should be developed in scope with getting values from the end users of the CFT for environments, SSH Key Pairs, EC2 instance types, RDS instance types, ElastiCache node types, etc., and this makes the entire CloudFormation template configurable and maintainable.

Example:

```
"WebTierKeyName": {
        "Description": "Name of an existing EC2 KeyPair to enable SSH access to the WebTier",
        "Type": "String",
        "MinLength": "1",
        "MaxLength": "64",
        "Default": "testkey",
        "AllowedPattern": "[-_ a-zA-Z0-9]*",
        "ConstraintDescription": "can contain only alphanumeric characters, spaces, dashes and underscores."
```

PRACTICE #3:
AMI ID SHOULD BE A PARAMETER IN INPUTS SECTION

AMI ID should be asked as an input parameter from the end users for launching EC2 instances. It is highly recommended not to hard code it inside your template and make it configurable dynamically.

Example:

```
"NatAMIID": {
        "Type": "String",
        "Default": "ami-XXXXXXXX",
        "Description": "The AMIID of NAT Instance"
    },
```

PRACTICE #4:
MENTION THE AMI MAPPINGS

AMI Mappings should be included in the CloudFormation template to validate the respective AMI(s) in the region(s) specified else it will take default AMI for the region (which is not advisable).

Example:

```
"Mappings": {
      "RegionMap": {
        "us-east-1": {
          "AMI": "ami-XXXXXXXX"
        },
"us-west-2": {
          "AMI": "ami-XXXXXXXX"
      },
```

PRACTICE #5:
USE WAITCONDITIONHANDLE, WAITCONDITION, DEPENDSON WHEREVER APPLICABLE

When creating a large multi-tiered infrastructure using CFT on AWS, there are times when the order of the resource launch is important. For Example WaitCondition that waits for the desired number of instances in a web server group.

Using the AWS::CloudFormation::WaitCondition and AWS::CloudFormation::WaitConditionHandle resources, a wait condition can be placed with in a template to make AWS CloudFormation pause the creation of the stack and wait for a signal before it continues to create the stack.

We can specify that the creation of a specific resource to follow another using the “DependsOn” attribute. On adding a DependsOn attribute to a resource, it is created only after the creation of the resource specified in the DependsOn attribute.

Example:

```
"WebServerGroup" : {
  "Type" : "AWS::AutoScaling::AutoScalingGroup",
  "Properties" : {
    "AvailabilityZones" : { "Fn::GetAZs" : "" },
    "LaunchConfigurationName" : { "Ref" : "LaunchConfig" },
    "MinSize" : "1",
    "MaxSize" : "5",
    "DesiredCapacity" : { "Ref" : "WebServerCapacity" },
    "LoadBalancerNames" : [ { "Ref" : "ElasticLoadBalancer" } ]
  }
},
"WaitHandle" : {
  "Type" : "AWS::CloudFormation::WaitConditionHandle"
},
"WaitCondition" : {
  "Type" : "AWS::CloudFormation::WaitCondition",
  "DependsOn" : "WebServerGroup",
  "Properties" : {
    "Handle"  : { "Ref" : "WaitHandle" },
    "Timeout" : "300",
    "Count"   : { "Ref" : "WebServerCapacity" }
  }
}
```

For example, an Amazon EC2 instance with a public IP address is dependent on the VPC-gateway attachment if the VPC and Internet Gateway resources are also declared in the same template. The following snippet shows a sample gateway attachment and an Amazon EC2 instance that depends on that gateway attachment:

```
"GatewayToInternet" : {
  "Type" : "AWS::EC2::VPCGatewayAttachment",
  "Properties" : {
    "VpcId" : { "Ref" : "VPC" },
    "InternetGatewayId" : { "Ref" : "InternetGateway" }
  }
},
"EC2Host" : {
  "Type" : "AWS::EC2::Instance",
  "DependsOn" : "GatewayToInternet",
    "Properties" : {
      "InstanceType" : { "Ref" : "EC2InstanceType" },
      "KeyName"  : { "Ref" : "KeyName" },
      "ImageId"  : { "Fn::FindInMap" : [ "AWSRegionArch2AMI", { "Ref" :
"AWS::Region" },
        { "Fn::FindInMap" : [ "AWSInstanceType2Arch", { "Ref" : "EC2Instance-
Type" }, "Arch" ] } ] },
      "NetworkInterfaces" : [{
        "GroupSet"                : [{ "Ref" : "EC2SecurityGroup" }],
        "AssociatePublicIpAddress" : "true",
        "DeviceIndex"             : "0",
        "DeleteOnTermination"     : "true",
        "SubnetId"                : { "Ref" : "PublicSubnet" }
      }]
    }
}
```

PRACTICE #6:
TAG THE RESOURCES PROPERLY

Tags should be used for the creation of AWS resources. Tags can be environment, purpose, application specification etc., In addition to the stack name tags that AWS CloudFormation provides, custom tags can be added to the resources that support tagging. This helps in easy grouping of the assets associated with the environment.

Example:

```
"Tags": [
            {
              "Key": "Environment",
              "Value": {
                "Ref": "Environment"
              }
            },
            {
              "Key": "Purpose",
              "Value": "CustomerXYZ VPC"
```

PRACTICE #7:
UNDERSTAND YOUR RESOURCES

When a CloudFormation template is deleted all AWS resources assigned with the template will be automatically deleted by AWS. For example a VPC template will have all the related resources like Route tables, Subnets, Network ACLs, etc., will be deleted.

In real scenario, S3 buckets logs, EBS snapshots etc are associated with an infrastructure but they will not terminated when the CFT is deleted, hence have to be automated for deletion using scripts.

PRACTICE #8:
CREATE YOUR SECURITY GROUPS USING CFT

Security Groups can be governed and controlled using CloudFormation template for various tiers like Web, App, DB, etc.,

```
"GatewayTierSg": {
        "Type": "AWS::EC2::SecurityGroup",
        "Properties": {
          "GroupDescription": "Gateway Security Group",
          "VpcId": {
            "Ref": "Vpc"
          },
          "SecurityGroupIngress": [
            {
              "IpProtocol": "6",
              "FromPort": "22",
              "ToPort": "22",
              "CidrIp": "0.0.0.0/0"
            }
          ],
          "SecurityGroupEgress": [
            {
              "IpProtocol": "-1",
              "CidrIp": "10.0.0.0/16"
            }
          ]
        }
      },
```

PRACTICE #9:
CREATE YOUR SECURITY GROUPS USING CFT

Output section should be declared with AWS specific end points to trace back. The template Outputs section enables in returning one or more values to the user in response to the AWS CloudFormation describe-stacks command. Example your endpoints are shown on the Output section.

Example:

```
"RDSDatabaseEndpointDetail": {
        "Description": "RDSDatabaseEndpointDetails",
        "Value": {
          "Fn::GetAtt": [
            "RdsTierService",
            "Endpoint.Address"
          ]
        }
      },
      "ElastiCacheEndpointDetail": {
        "Description": "ElastiCacheEndpointDetails",
        "Value": {
          "Fn::GetAtt": [
            "ElatiCacheTierService",
            "ConfigurationEndpoint.Address"
          ]
        }
      },
      "WebTierElbExternalEndpointDetail": {
        "Description": "WebTierElb External Endpoint Access Details",
        "Value": {
          "Fn::GetAtt": [
            "WebTierElbExternal",
            "DNSName"
          ]
        }
      },
```

PRACTICE #10:
AUTOMATE APPLICATION INSTALLATION USING CLOUD-INIT

With the help of cloud-init the installation of various packages can be done during the boot itself just by passing the appropriate commands. In the cloud-init the scripts also can be passed through the automation of various operations can be done.

```
"Resources" : {
   "Ec2Instance" : {
     "Type" : "AWS::EC2::Instance",
     "Properties" : {
       "KeyName" : { "Ref" : "KeyName" },
       "SecurityGroups" : [ { "Ref" : "InstanceSecurityGroup" } ],
       "ImageId" : { "Fn::FindInMap" : [ "RegionMap", { "Ref" : "AWS::Region" },
"AMI" ]},
       "UserData" : { "Fn::Base64" : { "Fn::Join" : ["",[
          "#!/bin/bash -ex","\n",
          "yum -y install gcc-c++ make","\n",
          "yum -y install mysql-devel sqlite-devel","\n",
          "yum -y install ruby-rdoc rubygems ruby-mysql ruby-devel","\n",
          "gem install --no-ri --no-rdoc rails","\n",
          "gem install --no-ri --no-rdoc mysql","\n",
          "gem install --no-ri --no-rdoc sqlite3","\n",
          "rails new myapp","\n",
          "cd myapp","\n",
          "rails server -d","\n",
          "curl -X PUT -H 'Content-Type:' --data-binary '{\"Status\" : \"SUC-
CESS\",",
                                    "\"Reason\" : \"The application myapp is
ready\",",
                                    "\"UniqueId\" : \"myapp\",",
                                    "\"Data\" : \"Done\"}' ",
            "\"", {"Ref" : "WaitForInstanceWaitHandle"},"\"\n" ]]}}
     }
```

PRACTICE #11:
USE HELPER SCRIPTS TO START/STOP SERVICES ETC

AWS CloudFormation provides a set of Python helper scripts that you can use to install software and start services on an Amazon EC2 instance that you create as part of your stack. You can call the helper scripts directly from your template. The scripts work in conjunction with resource metadata that you define in the same template. The helper scripts run on the Amazon EC2 instance as part of the stack creation process.

```
"Resources" : {

"WebServer": {
"Type": "AWS::EC2::Instance",
"Metadata" : {
"Comment1" : "Configure the bootstrap helpers to install the Apache Web
Server and PHP",
"Comment2" : "The website content is downloaded from the CloudForma-
tionPHPSample.zip file",

"AWS::CloudFormation::Init" : {
"config" : {
"packages" : {
"yum" : {
"mysql" : [],
"mysql-server" : [],
"mysql-libs" : [],
"httpd" : [],
"php" : [],
"php-mysql" : []
}
},

"sources" : {
"/var/www/html" : "https://s3.amazonaws.com/cloudformation-exam-
ples/CloudFormationPHPSample.zip"
},

"files" : {
"/tmp/setup.mysql" : {
"content" : { "Fn::Join" : ["", [
"CREATE DATABASE ", { "Ref" : "DBName" }, ";\n",
"GRANT ALL ON ", { "Ref" : "DBName" }, ".* TO '", { "Ref" : "DBUsername" },
"'@localhost IDENTIFIED BY '", { "Ref" : "DBPassword" }, "';\n"
]]},
"mode" : "000644",
"owner" : "root",
"group" : "root"
}
},
```

PRACTICE #12:
MAKE USE OF PARAMETER SECTION CONSTRAINTS (WHEREVER APPLICABLE)

The parameter sections provide lot of constraints like MinValue, Max Value, Allowed values, Max length, etc. By effective usage of the of the parameter section you can validate the input parameters from the user. Example: We can restrict TCP port value to be between 1 to 65535, we can restrict the webserver port to 80,8888 etc. This feature is particularly useful when you are developing complex cloudformation template for automation.

```
"Parameters" : {
   "UserAccount": {
     "Default": "guest",
     "NoEcho": "true",
     "Description" : "The guest account user name",
     "Type": "String",
     "MinLength": "1",
     "MaxLength": "16",
     "AllowedPattern" : "[a-zA-Z][a-zA-Z0-9]*"
   }
}
"Parameters" : {
   "WebServerPort": {
     "Default": "80",
     "Description" : "TCP/IP port for the web server",
     "Type": "Number",
     "MinValue": "1",
     "MaxValue": "65535"
   }
}
"Parameters" : {
   "WebServerPort": {
     "Default": "80",
     "Description" : "Port for the web server",
     "Type": "Number",
     "AllowedValues" : ["80", "8888"]
   }
}
```

PRACTICE #13:
USE NOECHO PROPERTY FOR SENSITIVE INFORMATION

When the cloudformation is in progress we can use describe-stacks to check the progress of the stack, in this case we can see the parameters values being returned. If we use the NoEcho Property like below, the sensitive information will not be returned.

```
"AdminPassword": {
    "NoEcho": "true",
    "Description" : "The Joomla! admin account password",
    "Type": "String",
    "MinLength": "1",
    "MaxLength": "41",
    "AllowedPattern" : "[a-zA-Z0-9]*",
    "ConstraintDescription" : "must contain only alphanumeric characters."
  }
```

PRACTICE #14:
USE SELECT FUNCTION AND REDUCE THE PARAMETERS IN THE TEMPLATE

Fn::Select intrinsic function: Using the above function in the in the Resources section of your template will help you to combine related parameters which can reduce the total number of parameters in your template.

```
"Parameters" : {
 "DbSubnetBlocks": {
   "Description": "Three CIDR blocks",
   "Type": "CommaDelimitedList",
    "Default": "10.0.58.0/24, 10.0.59.0/24, 10.0.60.0/24"
 }
}
```

To specify one of the three CIDR blocks, use Fn::Select in the Resources section of the same template, as shown in the following sample snippet:

```
"Subnet1": {
 "Type": "AWS::EC2::Subnet",
   "Properties": {
    "VpcId": { "Ref": "VPC" },
    "CidrBlock": { "Fn::Select" : [ "0", {"Ref": "DbSubnetBlocks"} ] }
   }
}
```

PRACTICE #15:
USE INTRINSIC FUNCTIONS TO MANAGE AUTOMATION

Fn::FindInMap intrinsic function

The function Fn::FindInMap returns the value corresponding to keys in a two-level map that is declared in the Mappingssection.

```
{
  ...
 "Mappings" : {
  "RegionMap" : {
    "us-east-1" : { "32" : "ami-6451e20d", "64" : "ami-7a11e213" },
    "us-west-1" : { "32" : "ami-d5gc7978c", "64" : "ami-cfc7978a" }
     }
 },

 "Resources" : {
   "myEC2Instance" : {
     "Type" : "AWS::EC2::Instance",
     "Properties" : {
       "ImageId" : { "Fn::FindInMap" : [ "RegionMap", { "Ref" : "AWS::Region" },
"32"]},
        "InstanceType" : "m3.large"
      }
    }
 }
}
```

You can use intrinsic functions, such as Fn::If, Fn::Equals, and Fn::Not, to conditionally create stack resources. These conditions are evaluated based on input parameters that you declare when you create or update a stack. After you define all your conditions, you can associate them with resources or resource properties in the Resources and Outputs sections of a template

PRACTICE #16:
PROPERTIES SECTION

If a resource does not require any properties to be declared, you can omit the Properties section of that resource. This will keep the template code simple and manageable

```
"Resources" : {
   "WebInstance" : {
     "Type" : "AWS::EC2::Instance",
     "Properties" : {
       "UserData" : {
         "Fn::Base64" : {
           "Fn::Join" : [ "", [ "Queue=", { "Ref" : "MyQueue" } ] ]
          } },
        "ImageId" : "ami-20b65349"
     }
   },

   "MyQueue" : {
     "Type" : "AWS::SQS::Queue",
     "Properties" : {
     }
   }
}
```

PRACTICE #17:
PLANNING WINDOWS BOOTSTRAPPING USING AMAZON CFT

In order for the Windows bootstrapping to work the CloudFormation the windows instance should be setup with the EC2ConfigService. The AWS CloudFormation helper script cfn-init is used to perform each of these actions, based on information in the AWS::CloudFormation::Init resource.

```
"SharePointFoundation": {
  "Type" : "AWS::EC2::Instance",
  "Metadata" : {
   "AWS::CloudFormation::Init" : {
    "config" : {

"files" : {
 "c:\\cfn\\cfn-hup.conf" : {
  "content" : { "Fn::Join" : ["", [
   "[main]\n",
   "stack=", { "Ref" : "AWS::StackName" }, "\n",
   "region=", { "Ref" : "AWS::Region" }, "\n"
   ]]}
 },
 "c:\\cfn\\hooks.d\\cfn-auto-reloader.conf" : {
  "content": { "Fn::Join" : ["", [
   "[cfn-auto-reloader-hook]\n",
   "triggers=post.update\n",
```

```
    "path=Resources.SharePointFoundation.Metadata.AWS::CloudFormation::Init\n",
    "action=cfn-init.exe -v -s ", { "Ref" : "AWS::StackName" },
                                 " -r SharePointFoundation",
                                 " --region ", { "Ref" : "AWS::Region" }, "\n"
  ]]}
 },
 "C:\\SharePoint\\SharePointFoundation2010.exe" : {
  "source" : "http://d3adzpja92utk0.cloudfront.net/SharePointFoundation.exe"
 }
},
```

PRACTICE #18:
USE NESTED STACKS

You can declare only Maximum of 200 resources in your AWS CloudFormation template. For complex environments and automation stacks you will need more than that, it is recommended to separate your template into multiple templates using the nested stacks feature.

```
{
  "AWSTemplateFormatVersion" : "2010-09-09",
  "Resources" : {
    "myStack" : {
      "Type" : "AWS::CloudFormation::Stack",
      "Properties" : {
        "TemplateURL" : "https://s3.amazonaws.com/cloudformation-templates-us-east-1/S3_Bucket.template",
        "TimeoutInMinutes" : "60"
      }
    }
  },
  "Outputs": {
    "StackRef": {"Value": { "Ref" : "myStack"}},
    "OutputFromNestedStack" : {
        "Value" : { "Fn::GetAtt" : [ "myStack", "Outputs.BucketName" ] }
    }
  }
}
```

PRACTICE #19:
INTEGRATE AMAZON CFT WITH AMAZON CLOUDTRAIL

It is recommended that AWS CloudFormation is integrated with the CloudTrail. The actions related to CloudFormation like **CreateStack, DeleteStack,** and **ListStacks** actions generate entries in CloudTrail log files. These logs can be processed for audit compliance purpose later.

San Ramon, CA (HQ)
12647 Alcosta Boulevard,
Suite 450,San Ramon, CA 94583, USA
www.securekloud.com
info@securekloud.com
Direct : 925-270-4800
Toll-Free : 855-856-4537

Chicago, IL
1827 Walden Office Square
Suite #460
Schaumburg, IL 60173
Phone : 708-289-5111

Dallas, TX
17740 Preston Road
Suite #200
Dallas, TX -75252
Phone : 214-272-2404

Chennai, India
Srinivasa Towers
New No.5, Old No. 11,
Cenotaph Road,
Alwarpet,Chennai – 600 018
Phone : +91-44-6602-8000
Fax : +91-44-4300-9049

Ontario, Canada
4 Robert Speck Parkway,
Suite 1500,
Mississauga, Ontario L4Z 1S1
Phone : 416-366-7762

Sharjah, UAE
Q1-05-109/C
SAIF Zone
PO Box 121213
Sharjah-UAE