General Template Snippets

The following examples show different AWS CloudFormation template features that aren't specific to an AWS service.

Topics

Base64 Encoded UserData Property

functions. The references `MyValue` and `MyName` are parameters that must be defined in the Parameters section of the template. The literal string `Hello World` is just another value this example passes in as part of the `UserData`.

JSON

```
"UserData" : {
    "Fn::Base64" : {
        "Fn::Join" : [ ",", [
            { "Ref" : "MyValue" },
            { "Ref" : "MyName" },
            "Hello World" ] ]
    }
}
```

YAML

```
UserData:
  Fn::Base64: !Sub |
     Ref: MyValue
     Ref: MyName
     Hello World
```

Base64 Encoded UserData Property with AccessKey and SecretKey

This example shows the assembly of a UserData property using the Fn::Base64 and Fn::Join functions. It includes the `AccessKey` and `SecretKey` information. The references `AccessKey` and `SecretKey` are parameters that must be defined in the Parameters section of the template.

JSON

```
"UserData" : {
    "Fn::Base64" : {
        "Fn::Join" : [ "", [
            "ACCESS_KEY=", { "Ref" : "AccessKey" },
            "SECRET_KEY=", { "Ref" : "SecretKey" } ]
        ]
    }
}
```

YAML

```
UserData:
```

```
  Fn::Base64: !Sub |
     ACCESS_KEY=${AccessKey}
     SECRET_KEY=${SecretKey}
```

Parameters Section with One Literal String Parameter

The following example depicts a valid Parameters section declaration in which a single `String` type parameter is declared.

JSON

```
"Parameters" : {
    "UserName" : {
        "Type" : "String",
        "Default" : "nonadmin",
        "Description" : "Assume a vanilla user if no command-line spec provided"
    }
}
```

YAML

```
Parameters:
  UserName:
    Type: String
    Default: nonadmin
    Description: Assume a vanilla user if no command-line spec provided
```

Parameters Section with String Parameter with Regular Expression Constraint

The following example depicts a valid Parameters section declaration in which a single `String` type parameter is declared. The AdminUserAccount parameter has a default of admin. The parameter value must have a minimum length of 1, a maximum length of 16, and contains alphabetic characters and numbers but must begin with an alphabetic character.

JSON

```
"Parameters" : {
    "AdminUserAccount": {
      "Default": "admin",
      "NoEcho": "true",
      "Description" : "The admin account user name",
      "Type": "String",
```

```
      "MinLength": "1",
      "MaxLength": "16",
      "AllowedPattern" : "[a-zA-Z][a-zA-Z0-9]*"
    }
}
```

YAML

```
Parameters:
  AdminUserAccount:
    Default: admin
    NoEcho: true
    Description: The admin account user name
    Type: String
    MinLength: 1
    MaxLength: 16
    AllowedPattern: '[a-zA-Z][a-zA-Z0-9]*'
```

Parameters Section with Number Parameter with MinValue and MaxValue Constraints

The following example depicts a valid Parameters section declaration in which a single `Number` type parameter is declared. The WebServerPort parameter has a default of 80 and a minimum value 1 and maximum value 65535.

JSON

```
"Parameters" : {
    "WebServerPort": {
      "Default": "80",
      "Description" : "TCP/IP port for the web server",
      "Type": "Number",
      "MinValue": "1",
      "MaxValue": "65535"
    }
}
```

YAML

```
Parameters:
  WebServerPort:
    Default: 80
    Description: TCP/IP port for the web server
    Type: Number
    MinValue: 1
```

```
    MaxValue: 65535
```

Parameters Section with Number Parameter with AllowedValues Constraint

The following example depicts a valid Parameters section declaration in which a single `Number` type parameter is declared. The WebServerPort parameter has a default of 80 and allows only values of 80 and 8888.

JSON

```
"Parameters" : {
    "WebServerPortLimited": {
      "Default": "80",
      "Description" : "TCP/IP port for the web server",
      "Type": "Number",
      "AllowedValues" : ["80", "8888"]
    }
}
```

YAML

```
Parameters:
  WebServerPortLimited:
    Default: 80
    Description: TCP/IP port for the web server
    Type: Number
    AllowedValues:
    - 80
    - 8888
```

Parameters Section with One Literal CommaDelimitedList Parameter

The following example depicts a valid Parameters section declaration in which a single `CommaDelimitedList` type parameter is declared. The NoEcho property is set to `TRUE`, which will mask its value with asterisks (*****) in the `aws cloudformation describe-stacks` output.

JSON

```
"Parameters" : {
    "UserRoles" : {
        "Type" : "CommaDelimitedList",
        "Default" : "guest,newhire",
```

```
        "NoEcho" : "TRUE"
    }
}
```

YAML

```
Parameters:
  UserRoles:
    Type: CommaDelimitedList
    Default: "guest,newhire"
    NoEcho: true
```

Parameters Section with Parameter Value Based on Pseudo Parameter

The following example shows commands in the EC2 user data that use the pseudo parameters `AWS::StackName` and `AWS::Region`. For more information about pseudo parameters, see Pseudo Parameters Reference.

JSON

```
        "UserData"       : { "Fn::Base64" : { "Fn::Join" : ["", [
           "#!/bin/bash -xe\n",
           "yum install -y aws-cfn-bootstrap\n",

           "/opt/aws/bin/cfn-init -v ",
           "         --stack ", { "Ref" : "AWS::StackName" },
           "         --resource LaunchConfig ",
           "         --region ", { "Ref" : "AWS::Region" }, "\n",

           "/opt/aws/bin/cfn-signal -e $? ",
           "         --stack ", { "Ref" : "AWS::StackName" },
           "         --resource WebServerGroup ",
           "         --region ", { "Ref" : "AWS::Region" }, "\n"
      ]]}}
    }
```

YAML

```
UserData:
  Fn::Base64: !Sub |
     #!/bin/bash -xe
     yum update -y aws-cfn-bootstrap
     /opt/aws/bin/cfn-init -v --stack ${AWS::StackName} --resource LaunchConfig --region
${AWS::Region}
     /opt/aws/bin/cfn-signal -e $? --stack ${AWS::StackName} --resource WebServerGroup --
```

```
region ${AWS::Region}
```

Mapping Section with Three Mappings

The following example depicts a valid Mapping section declaration that contains three mappings. The map, when matched with a mapping key of `Stop`, `SlowDown`, or `Go`, provides the RGB values assigned to the corresponding `RGBColor` attribute.

JSON

```
"Mappings" : {
    "LightColor" : {
        "Stop" : {
            "Description" : "red",
            "RGBColor" : "RED 255 GREEN 0 BLUE 0"
        },
        "SlowDown" : {
            "Description" : "yellow",
            "RGBColor" : "RED 255 GREEN 255 BLUE 0"
        },
        "Go" : {
            "Description" : "green",
            "RGBColor" : "RED 0 GREEN 128 BLUE 0"
        }
    }
}
```

YAML

```
Mappings:
  LightColor:
    Stop:
      Description: red
      RGBColor: "RED 255 GREEN 0 BLUE 0"
    SlowDown:
      Description: yellow
      RGBColor: "RED 255 GREEN 255 BLUE 0"
    Go:
      Description: green
      RGBColor: "RED 0 GREEN 128 BLUE 0"
```

Description Based on Literal String

The following example depicts a valid Description section declaration where the value is based on a

literal string. This snippet can be for templates, parameters, resources, properties, or outputs.

JSON

```
"Description" : "Replace this value"
```

YAML

```
Description: "Replace this value"
```

Outputs Section with One Literal String Output

This example shows a output assignment based on a literal string.

JSON

```
"Outputs" : {
    "MyPhone" : {
        "Value" : "Please call 555-5555",
        "Description" : "A random message for aws cloudformation describe-stacks"
    }
}
```

YAML

```
Outputs:
  MyPhone:
    Value: Please call 555-5555
    Description: A random message for aws cloudformation describe-stacks
```

Outputs Section with One Resource Reference and One Pseudo Reference Output

This example shows an Outputs section with two output assignments. One is based on a resource, and the other is based on a pseudo reference.

JSON

```
"Outputs" : {
   "SNSTopic" : { "Value" : { "Ref" : "MyNotificationTopic" } },
```

```
    "StackName" : { "Value" : { "Ref" : "AWS::StackName" } }
}
```

YAML

```
Outputs:
  SNSTopic:
    Value: Ref: MyNotificationTopic
  StackName:
    Value: Ref: AWS::StackName
```

Outputs Section with an Output Based on a Function, a Literal String, a Reference, and a Pseudo Parameter

This example shows an Outputs section with one output assignment. The Join function is used to concatenate the value, using a percent sign as the delimiter.

JSON

```
"Outputs" : {
    "MyOutput" : {
        "Value" : { "Fn::Join" :
            [ "%", [ "A-string", {"Ref" : "AWS::StackName" } ] ]
        }
    }
}
```

YAML

```
Outputs:
  MyOutput:
    Value: !Join [ %, [ 'A-string', !Ref 'AWS::StackName' ]]
```

Template Format Version

The following snippet depicts a valid Template Format Version section declaration.

JSON

```
"AWSTemplateFormatVersion" : "2010-09-09"
```

YAML

```
AWSTemplateFormatVersion: '2010-09-09'
```

AWS Tag Property

This example shows an AWS Tag property. You would specify this property within the Properties section of a resource. When the resource is created, it will be tagged with the tags you declare.

JSON

```
"Tags" : [
      {
        "Key" : "keyname1",
        "Value" : "value1"
      },
      {
        "Key" : "keyname2",
        "Value" : "value2"
      }
    ]
```

YAML

```
Tags:
  -
    Key: "keyname1"
    Value: "value1"
  -
    Key: "keyname2"
    Value: "value2"
```


« Previous Next »