Building More Powerful Templates with Built in Helpers

Overview

Creating multi-region template

- Mappings
- Fn::FindInMap

Using Pseudo Parameters

- AWS::AccountId, AWS::Region, ...
- Fn::GetAZs

Provisioning an EC2 Instance

- User Data
- Fn::Join and Fn::Base64

Common problems when creating a CloudFormation template …

- Multi-region support
- Install and configure software on EC2 instances automatically
- Reduce number of parameters

But how to do that in JSON?

- CloudFormation offers built in helpers for common problems

Adam, DevOps engineer at Globomantics

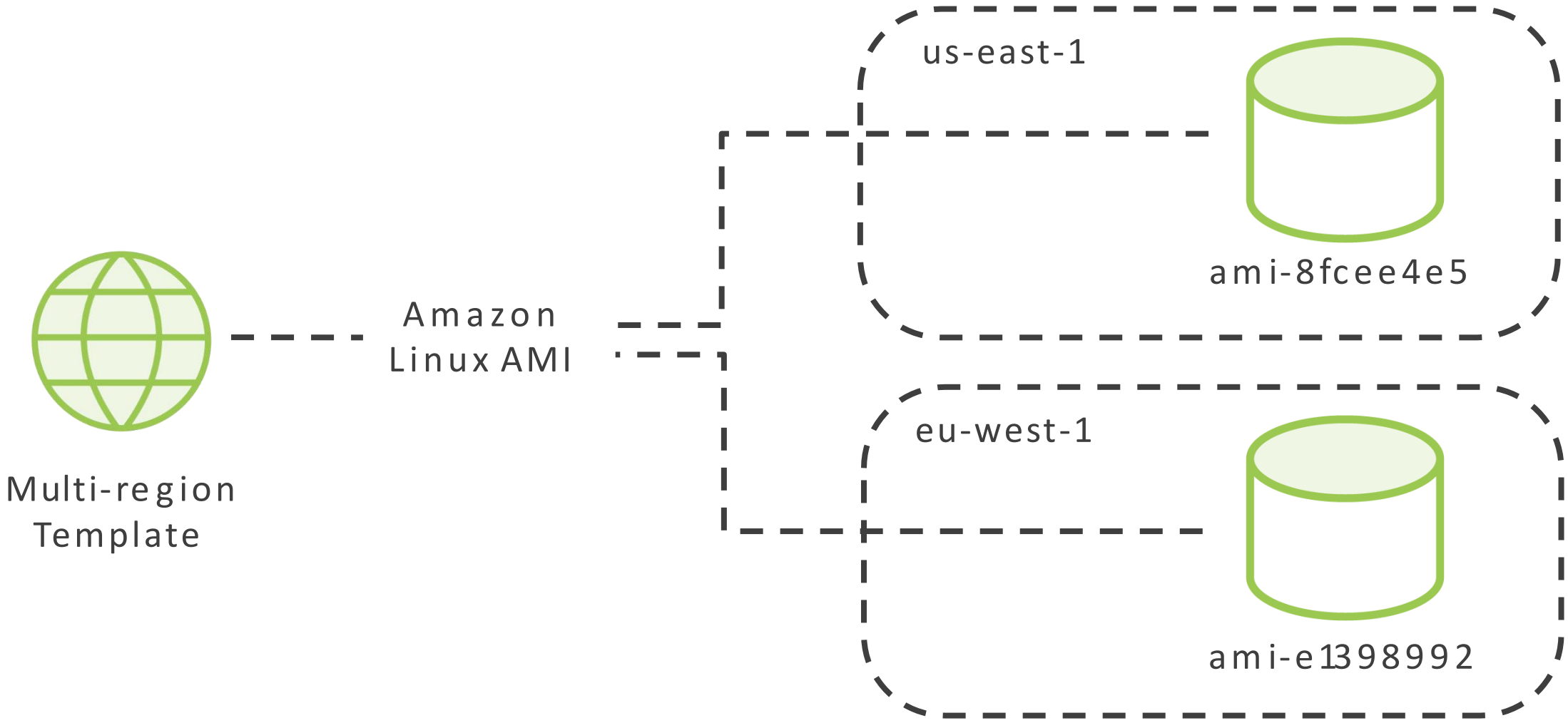
Create CloudFormation template

- Networking configuration
- EC2 Instance running web server
- Latest Amazon Linux AMI

Create stacks in the following regions

- us-east-1
- eu-west-1
- ap-northeast-1

Independent AMI per Region



Introducing Mappings

“A map is an object that maps keys to values.”

What is a map?

Mappings Section

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Description": "SSH Bastion Host",
    “Mappings": {...},
    "Resources": {...}
}
```

Defining a Mapping

```
"Mappings" : {
    "Mapping" : {
        "Key" : {
            "Name" : "Value"
        }
    }
}
```

Defining a Mapping

```
"Mappings" : {
        "Mapping" : {
                "Key" : {
                        "Name" : "Value"
                }
        }
}
```

Defining a Mapping

```
"Mappings" : {
        "Mapping" : {
                "Key" : {
                        "Name" : "Value"
                }
        }
}
```

```
"Mapping" : {
        "KeyA" : {},
        "KeyB" : {}
}
```

Unique Key

A key has to be unique within a mapping

```
"Key" : {
    "NameA" : "ValueA",
    "NameB" : "ValueB",
}
```

Named Values

A key can contain multiple named values

A name has to be unique within a key

Map AMIs per Region

```
"RegionAMI": {
        "us-east-1": {
                "AmazonLinux": "ami-8fcee4e5",
                "Ubuntu": "ami-fce3c696"
        },
        "eu-west-1": {
                "AmazonLinux": "ami-e1398992",
                "Ubuntu": "ami-f95ef58a"
}}
```

Map AMIs per Region

```
"RegionAMI": {
    "us-east-1": {
        "AmazonLinux": "ami-8fcee4e5",
        "Ubuntu": "ami-fce3c696"
    },
    "eu-west-1": {
        "AmazonLinux": "ami-e1398992",
        "Ubuntu": "ami-f95ef58a"
}}
```

Map AMIs per Region

```
"RegionAMI": {
    "us-east-1": {
        "AmazonLinux": "ami-8fcee4e5",
        "Ubuntu": "ami-fce3c696"
    },
    "eu-west-1": {
        "AmazonLinux": "ami-e1398992",
        "Ubuntu": "ami-f95ef58a"
}}
```

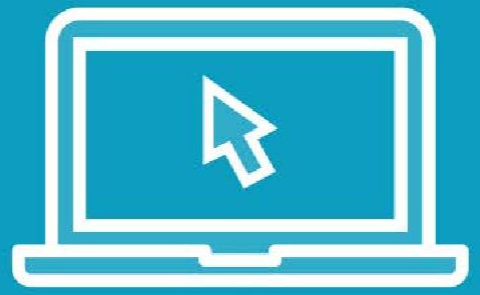
"Fn::FindInMap": ["RegionAMI", "eu-west-1", "AmazonLinux"]

Fn:: FindInMap

Allows to access values from a Mapping

Accepts Mapping name, Key, and Name as parameters

Demo



CloudFormation template

- Networking configuration
- EC2 Instance
- Security Group

Mapping contains AMI IDs for multiple regions

Create stack in multiple regions with the help of AWS CLI

Using Pseudo Parameters

Adam, DevOps engineer at Globomantics

CloudFormation template reusable for multiple regions

How to define the Availability Zone for a Subnet?

Define AZ for Multi-Region Template

```
"Subnet": {
    "Type": "AWS::EC2::Subnet",
    "Properties": {
    "AvailabilityZone": "???",
    "CidrBlock": "10.0.0.0/24",
    "VpcId": {"Ref": "VPC"},
    "MapPublicIpOnLaunch": "true"
}}
```

Pseudo Parameters

AWS::AccountId	AWS account ID
AWS::NotificationARNs	ARNs for notification topics
AWS::NoValue	Removes attribute
AWS::Region	Region of current stack
AWS::StackId	ID of current stack
AWS::StackName	Name of current stack

```
{"Ref": "AWS::Region"}
```

Accessing a Pseudo Parameter

Use built in Ref function to access pseudo parameters

```
{"Fn::GetAZs": {"Ref": "AWS::Region"}}
```

Fn::GetAZs

Returns all Availability Zones for a region

Uses current region of stack in this example

{"Fn::Select": ["0", ["a", "b", "c"]}

Fn::Select

Selects a value from a list

Selects first value of list in this example

Define AZ for Multi-region Template

```
"Type": "AWS::EC2::Subnet",
"Properties": {
    "AvailabilityZone": {
        "Fn::Select": [
            "0",
            {"Fn::GetAZs": {"Ref": "AWS::Region"}}
        ]
    }, ...
}
```

Define AZ for Multi-region Template

```
"Type": "AWS::EC2::Subnet",
"Properties": {
    "AvailabilityZone": {
        "Fn::Select": [
            "0",
            {"Fn::GetAZs": {"Ref": "AWS::Region"}}
        ]
    }, ...
}
```

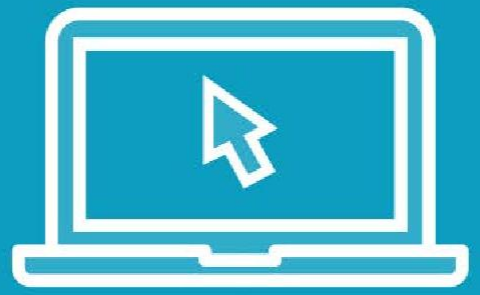
Define AZ for Multi-region Template

```
"Type": "AWS::EC2::Subnet",
"Properties": {
    "AvailabilityZone": {
        "Fn::Select": [
            "0",
            {"Fn::GetAZs": {"Ref": "AWS::Region"}}
        ]
    }, ...
}
```

Template Reference

http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/template-reference.html

Demo



CloudFormation Template

- Pseudo Parameters
- Fn::GetAZs
- Fn::Select

Provision EC2 Instance Automatically

Adam, DevOps engineer at Globomantics

CloudFormation template includes EC2 Instance

Launch Web Server on EC2 Instance automatically

Necessary Steps During Bootstrap

Install

Install aHTTP server on the EC2 Instance

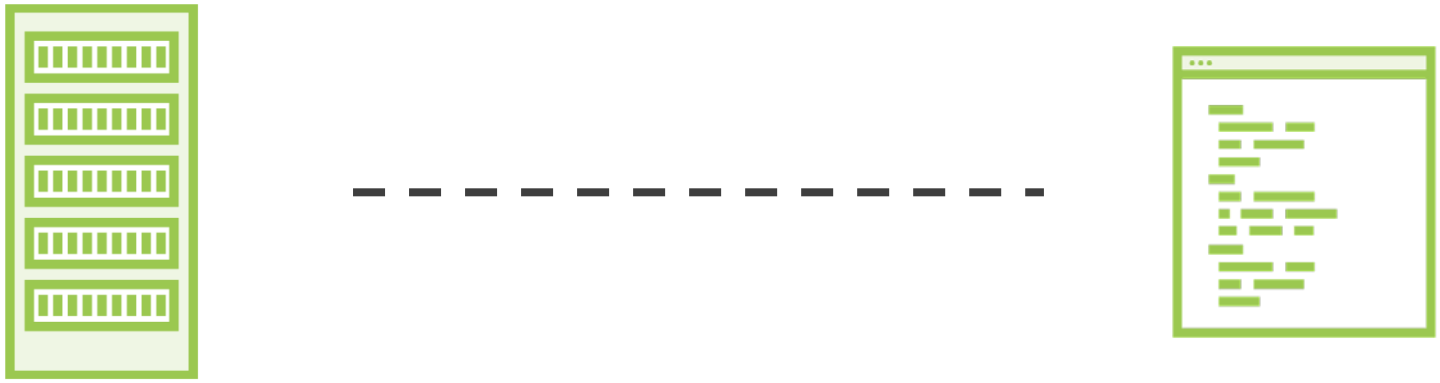
Configure

Configure the HTTP server

Start

Start the HTTP server

User Data of EC2 Instance



EC2 Instance

Is able to access User Data. Executes Shell script during bootstrap.

User Data

Defined when EC2 Instance is launched. Contains a Shell script

```
"Type": "AWS::EC2::Instance",
        "Properties": {
        "UserData": "..."
}
```

User Data Property

User Data can be described as property of an EC2 Instance

But User Data needs to beencoded in base64

{"Fn::Base64": "A String"}

Fn::Base64

There's a built in function for that

Encodes a String in base64

{"Fn::Join": [";", ["a", "b", "c"]]}

Fn::Join

Allows you to concatenate Strings

Accepts a delimiter and a list of Strings as input

User Data Containing Shell Script

```
"UserData": {"Fn::Base64": {"Fn::Join": ["\n", [
        "#!/bin/bash -ex",
        "yum install -y httpd",
        "cd /var/www/html",
        "echo '<html><body>...</body></html>' > index.html",
        "service httpd start"
]]}}
```

User Data Containing Shell Script

```
"UserData": {"Fn::Base64": {"Fn::Join": ["\n", [
        "#!/bin/bash -ex",
        "yum install -y httpd",
        "cd /var/www/html",
        "echo '<html><body>...</body></html>' > index.html",
        "service httpd start"
]]}}
```

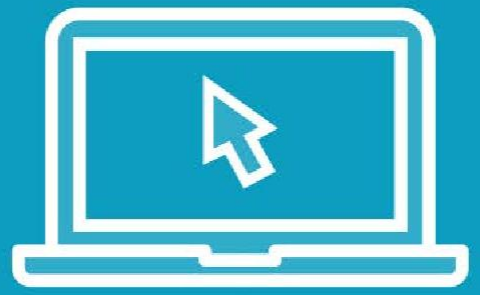
User Data Containing Shell Script

```
"UserData": {"Fn::Base64": {"Fn::Join": ["\n", [
        "#!/bin/bash -ex",
        "yum install -y httpd",
        "cd /var/www/html",
        "echo '<html><body>...</body></html>' > index.html",
        "service httpd start"
]]}}
```

User Data Containing Shell Script

```
"UserData": {"Fn::Base64": {"Fn::Join": ["\n", [
        "#!/bin/bash -ex",
        "yum install -y httpd",
        "cd /var/www/html",
        "echo '<html><body>...</body></html>' > index.html",
        "service httpd start"
]]}}
```

Demo



CloudFormation Template

- EC2 Instance
- User Data containing Shell Script

Create stack with the help of AWS CLI and access the created Web Server

Summary

Mappings section

- Map between region and AMI

Pseudo Parameters

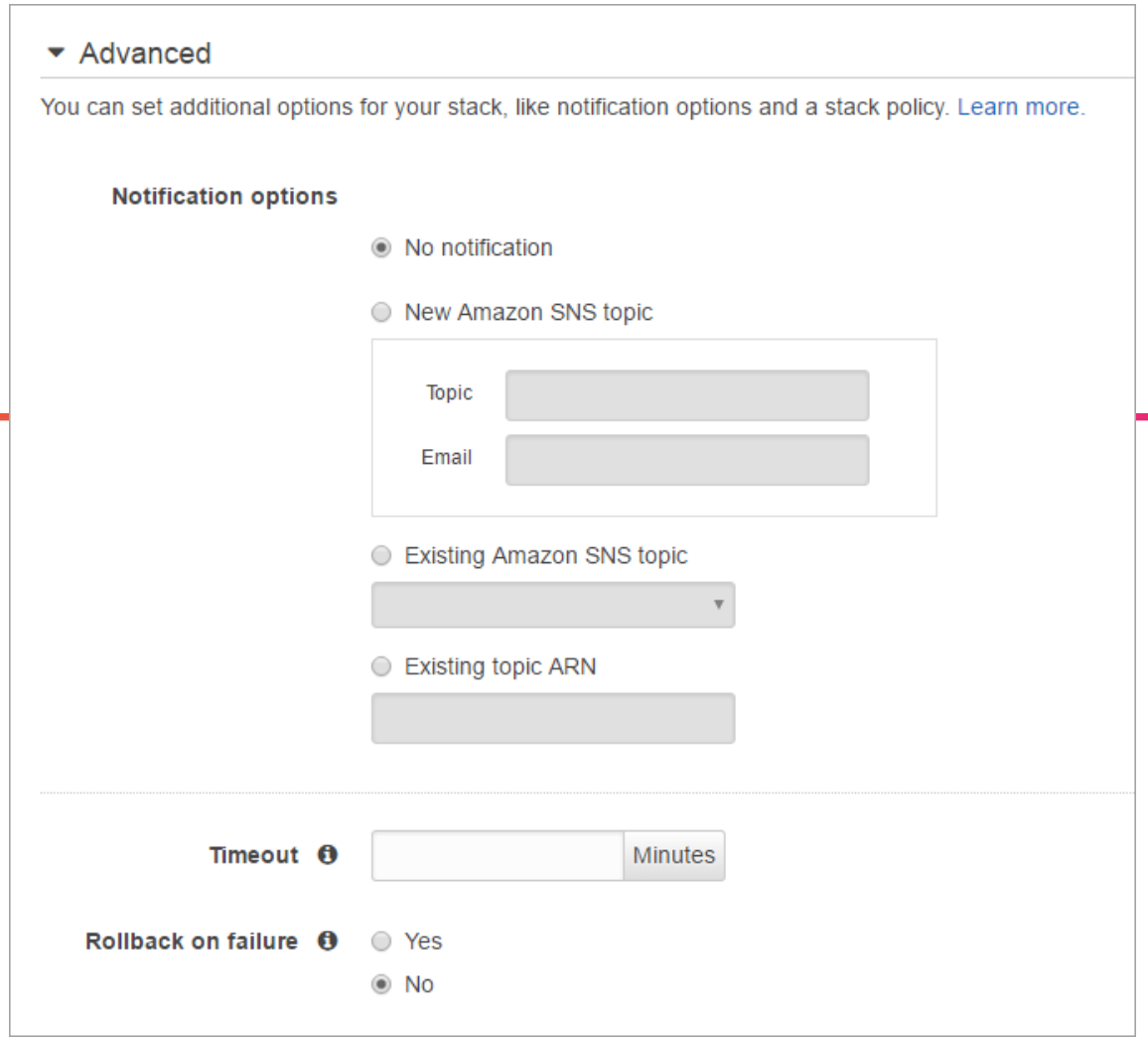
Built in functions

- Fn::GetAZs
- Fn::Select
- Fn::Join
- Fn::Base64

User Data

- Provision EC2 Instance automatically

Troubleshooting Failed Stacks



Troubleshooting Failed Stacks

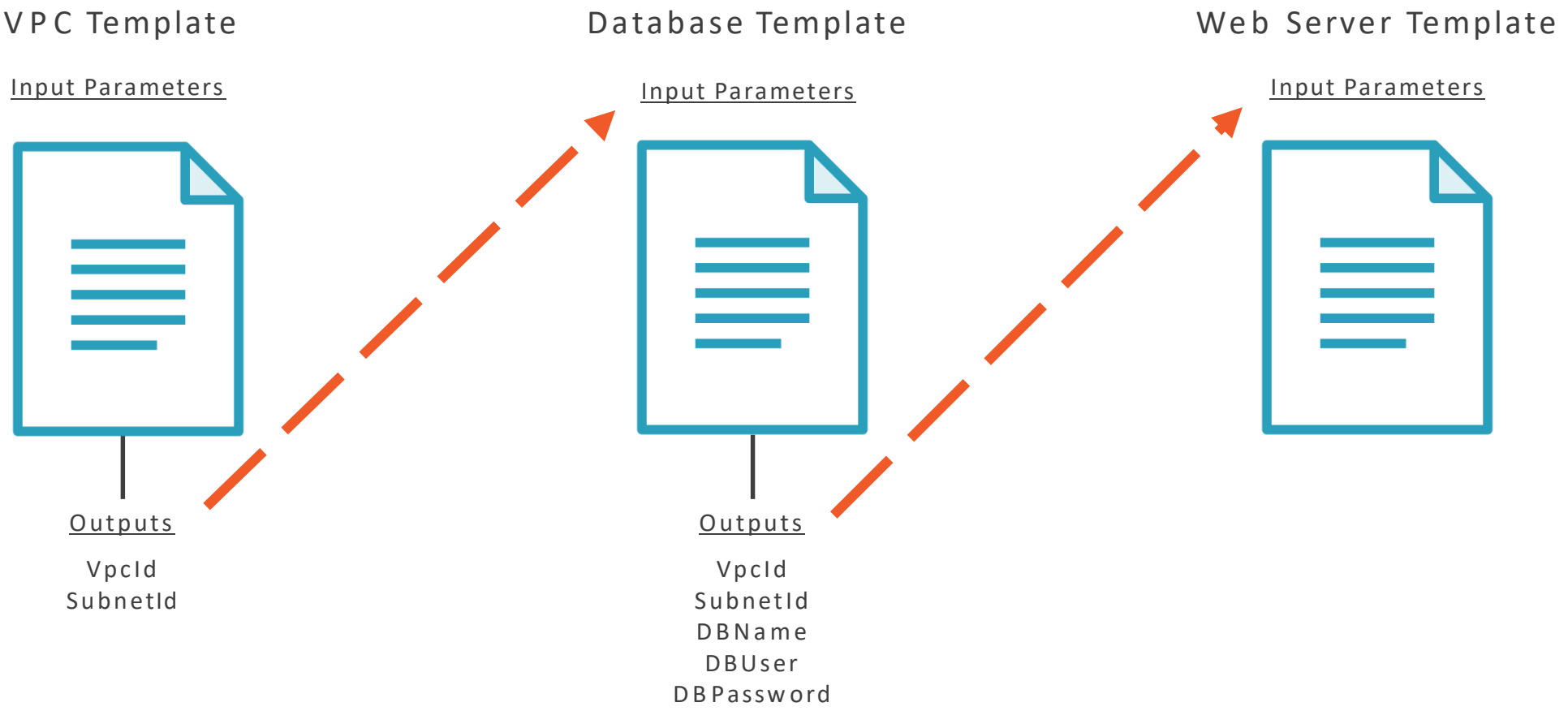
Linux

Review log files in /var/log

Windows

Review log files in c:\cfn\log

Creating Nested Templates

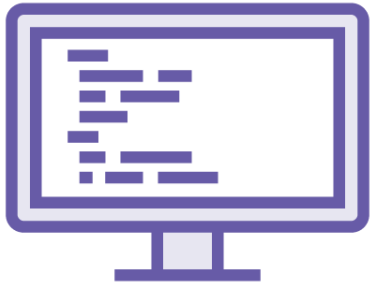


CloudFormation Best Practices

Don't embed credentials

Validate templates

Manage Everything from CloudFormation



Use constraints & AWS-specific parameter types

Use nested stacks to reduce code duplication

Use code reviews and store templates in version control

Summary

Template authoring

Bootstrapping

Nested templates

Troubleshooting failed stacks

Best practices