Amazon Web Services CloudFormation

tecRACER

Cloud Enabling Your Business

amazon web services | Partner Network

PREMIER CONSULTING PARTNER

tecRACER

Cloud Enabling Your Business

- AWS Consulting
- AWS Training
- AWS Managed Hosting
- Software development on AWS

Andreas Wittig

- software engineer
- senior consultant for AWS
- author of AWS in Action
- DevOps enthusiast
- migrated first German bank to AWS

Deploying on AWS

- AWS CodeDeploy
- AWS Elastic Beanstalk
- AWS OpsWorks
- AWS CloudFormation

CodeDeploy

- Deploy from S3 or GitHub to EC2 or on-premise servers
- rolling update and rollback supported
- low-level service focusing on software deployment
- agent based (Amazon Linux, Ubuntu, RHEL, Windows Server)
- integrated with CodePipeline

CodeDeploy

Application

bundle of revisions and deployments

Revision

bundled source code accessible via S3 or GitHub

Deployment

revision + deployment config (e.g. rolling update)

CodeDeploy

Deployment Group

EC2 instances filtered by tags

AppSpec

instructions for deployment part of revision

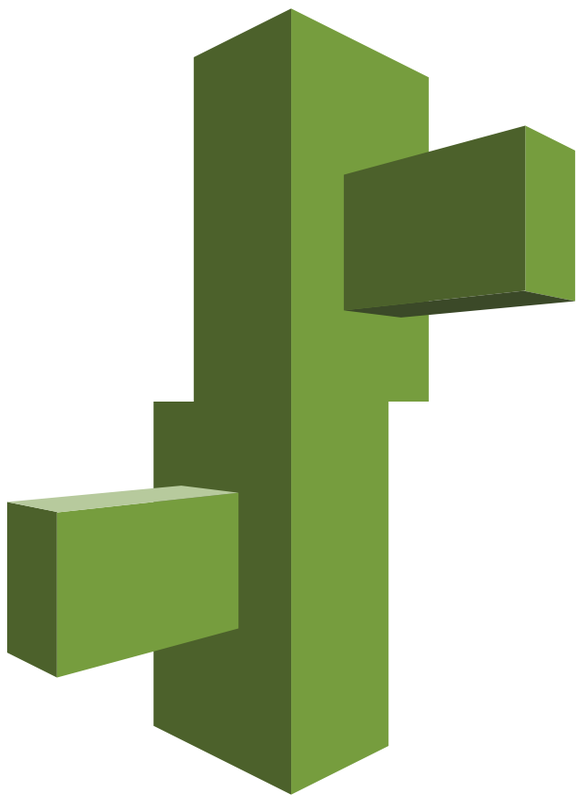
Comparing tools

Elastic Beanstalk

OpsWorks

CloudFormation
with custom scripts

Elastic Beanstalk



- supports PHP, Java, .NET, Ruby, Node.js, Python, Go, and Docker
- designed for common web applications
- easy to use: three steps from zero to running applicaiton

OpsWorks

- designed for more complex environments with multiple layers
- uses Chef (configuration management)
- standard layers: HAProxy, static web server, Ruby on Rails, PHP, Java, MySQL, Memcached and Ganglia

OpsWorks: app sources

- Git
- Subversion
- Amazon S3 bundle
- HTTP bundle

Infrastructure as Code

- automate changes of infrastructure
- bundle infrastructure and application
- description of infrastructure available on Git/SVN/…
- automated test for infrastructure changes
- deploy infrastructure and application at once
- testing system = production system

CloudFormation

- template: blueprint written in JSON
- stack: instance of a template in a region
- supports 90 % of services on AWS
- create, update and delete infrastructure

Labs

git clone

https://github.com/widdix/learn-cloudformation

Lab 0: Create Stack

Create a CloudFormation stack based on a template.

Lab: learn-cloudformation/lab0-create-stack

Combine them …

One tool to rule them all …

Use CloudFormation to control OpsWorks and Elastic Beanstalk

Agenda

- Introducing Amazon Web Services
- Computing and Networking
- Storing data
- Deploying applications
- **Deep Dive: CloudFormation**

Minimal CF template

- Version
- Description
- Resources

Example: CF template

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Description": "Empty template",
  "Resources": {
    [...]
  }
}
```

Resources

AWS::EC2::Instance

AWS::S3::Bucket

AWS::IAM::Role

AWS::RDS::DBInstance

…

http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-template-resource-type-ref.html

Example: Resource

```
"WebServerSecurityGroup": {
  "Type": "AWS::EC2::SecurityGroup",
  "Properties": {
    "GroupDescription": "web server",
    "VpcId": [...],
    "SecurityGroupIngress": [{
      "CidrIp": "0.0.0.0/0",
      "FromPort": 80,
      "IpProtocol": "tcp",
      "ToPort": 80
    }]
  }
}
```

Lab 1: Simple Template

Create your first CloudFormation template.

Lab: learn-cloudformation/lab1-simple-template

Reuse templates

String, Number, List, CommaDelimitedList
AWS specific types (e.g. AWS::EC2::KeyPair::KeyName)

http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/parameters-section-structure.html

Define parameters

```
"Parameters": {
  "Environment": {
    "Description": "name of environment",
    "Type": "String"
  },
  "Subnet": {
    "Description": "subnet to launch virtual server in",
    "Type": "AWS::EC2::Subnet::Id"
  }
}
```

Reference parameters

```
{
  "Parameters": {
    "AMI": {
      "Description": "AMI to start virtual server",
      "Type": "String",
    }
  },
  "Resources": {
    "EC2Instance": {
      "Type": "AWS::EC2::Instance",
      "Properties": {
        "ImageId": {"Ref": "AMI"}
      }
    }
  }
}
```

Validate parameters

```
"Parameters": {
  "AMI": {
    "Description": "AMI to start virtual server",
    "Type": "String",
    "MaxLength": 12,
    "MinLength": 12
  },
  "InstanceType": {
    "Description": "instance type of virtual server",
    "Type": "String",
    "AllowedValues": ["t2.micro", "t2.small",
"t2.medium"]
  }
}
```

Lab 2: Parameters

Use parameters to be able to reuse your templates.

Lab: learn-cloudformation/lab2-parameters

Functions

Fn::Base64
Fn::FindInMap
Fn::GetAtt
Fn::GetAZs
Fn::Join
Fn::Select

Fn::And
Fn::Equals
Fn::If
Fn::Not
Fn::Or

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

Use functions

```
"UserData": {"Fn::Base64": {"Fn::Join": ["", [
  "#!/bin/bash -ex\n",
  "yum install -y ", {"Ref": "package"}, "\n"
]]}}
```

Lab 3: Functions

Use build-in functions within your template.

Lab: learn-cloudformation/lab3-functions

Publish results

- API/HTTPS endpoint
- public IP addresses for DNS entries
- …

http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/outputs-section-structure.html

Define outputs

```
"Outputs": {
  "InstanceId": {
    "Value": {"Ref": "EC2Instance"},
    "Description": "ID of virtual server"
  },
  "PublicIPAddress": {
    "Value": {"Fn::GetAtt": ["EC2Instance",
"PublicIp"]},
    "Description": "public IP address of virtual server"
  }
}
```

Lab 4: Outputs

Use outputs from CloudFormation stacks.

Lab: learn-cloudformation/lab4-outputs

Dependencies

Automagically

CloudFormation resolves dependencies between resources automatically.

Describe dependencies

```
"SecurityGroup": {
  "Type": "AWS::EC2::SecurityGroup",
  "Properties": {
    "GroupDescription": "allow-ssh",
    "VpcId": {"Ref": "VPC"}
  }
},
"EC2Instance": {
  "Type": "AWS::EC2::Instance",
  "Properties": {
    "InstanceType": "t2.micro",
    "SecurityGroupIds": [{"Ref": "SecurityGroup"}]
  }
}
```

Lab 5: Dependencies

Define dependencies between resources.

Lab: learn-cloudformation/lab5-dependencies

Mappings

Map Key -> Value

Useful to define parameters based on region, environment, …

Define mappings

```
"Mappings": {
  "RegionAMIMap": {
    "ap-northeast-1": {"AmazonLinux": "ami-cbf90ecb"},
    "ap-southeast-1": {"AmazonLinux": "ami-68d8e93a"},
    "ap-southeast-2": {"AmazonLinux": "ami-fd9cecc7"},
    "eu-central-1": {"AmazonLinux": "ami-a8221fb5"},
    "eu-west-1": {"AmazonLinux": "ami-a10897d6"},
    "sa-east-1": {"AmazonLinux": "ami-b52890a8"},
    "us-east-1": {"AmazonLinux": "ami-1ecae776"},
    "us-west-1": {"AmazonLinux": "ami-d114f295"},
    "us-west-2": {"AmazonLinux": "ami-e7527ed7"}
  }
}
```

Access mappings

```
{"Fn::FindInMap": ["RegionAMIMap", {"Ref":
"AWS::Region"}, "AmazonLinux"]}
```

Lab 6: Mappings

Use mappings to map from keys to named values.

Lab: learn-cloudformation/lab6-mappings

Nested Stacks

Stack stacks into stacks

Modularize your infrastructure. Powerful but heavy.
Consider loose coupling.

Describe stacks

```
"Resources": {
  "S3Stack": {
    "Type" : "AWS::CloudFormation::Stack",
    "Properties" : {
      "TemplateURL" : "https://.../1-simple.json",
      "TimeoutInMinutes" : "2"
    }
  }
}
```

Lab 7: Nested stacks

Use nested stacks to modulize your templates.

Lab: learn-cloudformation/lab7-nested-stacks

cfn-init

Initialize EC2 instance based on metadata

Components

- Small tool running on your EC2 instance
- Pre-installed on Amazon Linux
- Uses metadata from CloudFormation stack to initialize EC2 instance

Features

- Install packages
- Control services
- Create/Modify files
- Execute commands
- Create users and groups
- Download sources

cfn-init: metadata

```
"AWS::CloudFormation::Init" : {
  "config" : {
    "packages": {
      "yum" : {"httpd": []}
    },
    "files" : {
      "/var/www/html/index.html": {
        "content": "<html><body>Hello World!</body></
html>",
        "mode"  : "000644"
      }
    }
  }
}
```

Lab 8: cfn-init

Use cfn-init to initialize EC2 instance.

Lab: learn-cloudformation/lab8-cfn-init

Rolling Update

Blue-Green-Deployment

Rolling Update

Components

- Auto Scaling Group
- CloudFormation
  - Update-Policy
  - Wait-Condition (cfn-signal)

Auto Scaling Group

```
"AutoScalingGroup": {
  "Type": "AWS::AutoScaling::AutoScalingGroup",
  "Properties": {
    "MinSize": "1",
    "MaxSize": "2",
    "DesiredCapacity": "1",
    "LaunchConfigurationName": "..."
  }
}
```

Update Policy

```
"AutoScalingGroup": {
  "Type": "AWS::AutoScaling::AutoScalingGroup",
  "Properties": {},
  "UpdatePolicy": {
    "AutoScalingRollingUpdate": {
      "MaxBatchSize": "2",
      "MinInstancesInService": "1",
      "PauseTime": "PT10M",
      "SuspendProcesses": ["AlarmNotification"],
      "WaitOnResourceSignals": true
    }
  }
}
```

Creation Policy

```
"AutoScalingGroup": {
  "Type": "AWS::AutoScaling::AutoScalingGroup",
  "Properties": {},
  "CreationPolicy": {
    "ResourceSignal": {
      "Count": 1,
      "Timeout": "PT10M"
    }
  }
}
```

cfn-signal

```
/opt/aws/bin/cfn-signal -e 0 --region eu-west-1
  --stack test --resource AutoScalingGroup
```

Lab 9: Rolling Update

Explore a rolling update in action.

Lab: learn-cloudformation/lab9-rolling-update

Andreas Wittig

awittig@tecracer.de

@andreaswittig