Integrating AWS CloudFormation with Puppet

AWS CloudFormation gives you an easy way to create the set of resources such as Amazon EC2 instance, Amazon RDS database instances and Elastic Load Balancers needed to run your application. The template describes *what* resources you need and AWS CloudFormation takes care of *how:* provisioning the resources in an orderly and predictable fashion, handling and recovering from any failures or issues. For more details of using AWS CloudFormation see the AWS CloudFormation product detail page.

While AWS CloudFormation takes care of provisioning all the resources, it raises the obvious question of how your application software is deployed, configured and executed on the Amazon EC2 instances. There are many options, each of which has implications on how quickly your application is *ready* and how flexible you need to be in terms of deploying new versions of the software. The remainder of this document shows how to use the AWS CloudFormation bootstrap helper scripts to deploy applications using Puppet. It builds on the features of AWS CloudFormation introduced in the whitepaper *Bootstrapping Applications With AWS CloudFormation*.

Overview

Puppet is an open source platform for provisioning, configuring and patching applications and operating system components. A Puppet deployment consists of 2 basic building blocks:

Puppet Master*:* The puppet master is a centralized configuration server that holds the definitions and instructions needed to install applications as well as server roles.

Puppet Client: A Puppet client connects to a Puppet server to download the necessary instructions to install, update and patch the software running on it.

For more details on using Puppet see the Puppet Labs documentation site.

The remainder of this section describes how to bootstrap a Puppet Master and Puppet clients using AWS CloudFormation on the base Amazon Linux AMI.

Note: This document assumes some familiarity with Puppet; it is not intended to be a tutorial on Puppet.

Creating a Puppet Master

The Puppet Master is the central location for managing you application and OS component configuration. The following template shows how to bootstrap Puppet Master on a base Amazon Linux AMI:

```
{
  "AWSTemplateFormatVersion" : "2010-09-09",

  "Description": "Sample template to bring up Puppet Master instance that can be used to bootstrap and manage Puppet Clients. The Puppet Master is
populated from an embedded template that defines the set of applications to load. **WARNING** This template creates one or more Amazon EC2 instances.
You will be billed for the AWS resources used if you create a stack from this template.",
```

```
  "Parameters" : {
    "InstanceType" : {
      "Description" : "EC2 instance type for PuppetMaster",
      "Type" : "String",
      "Default" : "t1.micro",
      "AllowedValues" : [ "t1.micro", "m1.small", "m1.large", "m1.xlarge", "m2.xlarge", "m2.2xlarge", "m2.4xlarge", "c1.medium", "c1.xlarge",
"cc1.4xlarge" ],
      "ConstraintDescription" : "must contain only alphanumeric characters."
    },
    "KeyName" : {
      "Description" : "Name of an existing EC2 KeyPair to enable SSH access to the PuppetMaster",
      "Type" : "String"
    },
    "ContentManifest" : {
      "Default" : "/wordpress/: { include wordpress }",
      "Description" : "Manifest of roles to add to nodes.pp",
      "Type" : "String"
    },
    "ContentLocation" : {
      "Default" : "https://s3.amazonaws.com/cloudformation-examples/wordpress-puppet-config.tar.gz",
      "Description" : "Location of package (Zip, GZIP or Git repository URL) that includes the PuppetMaster content",
      "Type" : "String"
    }
  },


  "Mappings" : {
    "AWSInstanceType2Arch" : {
      "t1.micro"    : { "Arch" : "32" },
      "m1.small"    : { "Arch" : "32" },
      "m1.large"    : { "Arch" : "64" },
      "m1.xlarge"   : { "Arch" : "64" },
      "m2.xlarge"   : { "Arch" : "64" },
      "m2.2xlarge"  : { "Arch" : "64" },
      "m2.4xlarge"  : { "Arch" : "64" },
      "c1.medium"   : { "Arch" : "32" },
      "c1.xlarge"   : { "Arch" : "64" },
      "cc1.4xlarge" : { "Arch" : "64" }
    },
    "AWSRegionArch2AMI" : {
      "us-east-1"      : { "32" : "ami-7f418316", "64" : "ami-7341831a" },
      "us-west-1"      : { "32" : "ami-951945d0", "64" : "ami-971945d2" },
      "eu-west-1"      : { "32" : "ami-24506250", "64" : "ami-20506254" },
      "ap-southeast-1" : { "32" : "ami-74dda626", "64" : "ami-7edda62c" },
      "ap-northeast-1" : { "32" : "ami-dcfa4edd", "64" : "ami-e8fa4ee9" }
    }
  },

  "Resources" : {
    "CFNInitUser" : {
      "Type" : "AWS::IAM::User",
      "Properties" : {
        "Policies": [{
          "PolicyName": "AccessForCFNInit",
          "PolicyDocument" : {
            "Statement": [{
              "Effect"   : "Allow",
              "Action"   : "cloudformation:DescribeStackResource",
              "Resource" : "*"
            }]
          }
        }]
      }
    },

    "CFNKeys" : {
      "Type" : "AWS::IAM::AccessKey",
      "Properties" : {
        "UserName" : { "Ref": "CFNInitUser" }
      }
    },

    "PuppetMasterInstance" : {
      "Type" : "AWS::EC2::Instance",
      "Metadata" : {
        "AWS::CloudFormation::Init" : {
          "config" : {
            "packages" : {
              "yum" : {
                "puppet"        : [],
                "puppet-server" : [],
                "ruby-devel"    : [],
                "gcc"           : [],
                "make"          : [],
                "rubygems"      : []
              },
              "rubygems" : {
                "json"          : []
              }
            },
            "sources" : {
              "/etc/puppet" : { "Ref" : "ContentLocation" }
```

```
          },
          "files" : {
            "/etc/yum.repos.d/epel.repo" : {
              "source" : "https://s3.amazonaws.com/cloudformation-examples/enable-epel-on-amazon-linux-ami",
              "mode"   : "000644",
              "owner"  : "root",
              "group"  : "root"
            },
            "/etc/puppet/autosign.conf" : {
              "content" : "*.internal\n",
              "mode"    : "100644",
              "owner"   : "root",
              "group"   : "wheel"
            },
            "/etc/puppet/fileserver.conf" : {
              "content" : "[modules]\n   allow *.internal\n",
              "mode"    : "100644",
              "owner"   : "root",
              "group"   : "wheel"
            },
            "/etc/puppet/puppet.conf" : {
              "content" : { "Fn::Join" : ["", [
                "[main]\n",
                "   logdir=/var/log/puppet\n",
                "   rundir=/var/run/puppet\n",
                "   ssldir=$vardir/ssl\n",
                "   pluginsync=true\n",
                "[agent]\n",
                "   classfile=$vardir/classes.txt\n",
                "   localconfig=$vardir/localconfig\n"]] },
              "mode"    : "000644",
              "owner"   : "root",
              "group"   : "root"
            },
            "/etc/puppet/modules/cfn/manifests/init.pp" : {
              "content" : "class cfn {}",
              "mode"    : "100644",
              "owner"   : "root",
              "group"   : "wheel"
            },
            "/etc/puppet/modules/cfn/lib/facter/cfn.rb" : {
              "source"  : "https://s3.amazonaws.com/cloudformation-examples/cfn-facter-plugin.rb",
              "mode"    : "100644",
              "owner"   : "root",
              "group"   : "wheel"
            },
            "/etc/puppet/manifests/nodes.pp" : {
              "content" : {"Fn::Join" : ["", [
                "node basenode {\n",
                "  include cfn\n",
                "}\n",
                "node /^.*internal$/ inherits basenode {\n",
                "  case $cfn_roles {\n",
                "    ", { "Ref" : "ContentManifest" }, "\n",
                "  }\n",
                "}\n"]]},
              "mode"    : "100644",
              "owner"   : "root",
              "group"   : "wheel"
            },
            "/etc/puppet/manifests/site.pp" : {
              "content" : "import \"nodes\"\n",
              "mode"    : "100644",
              "owner"   : "root",
              "group"   : "wheel"
            }
          },
          "services" : {
            "sysvinit" : {
              "puppetmaster" : {
                "enabled" : "true",
                "ensureRunning" : "true"
              }
            }
          }
        }
      }
    },
    "Properties" : {
      "InstanceType" : { "Ref" : "InstanceType" },
      "SecurityGroups" : [ { "Ref" : "PuppetGroup" } ],
      "ImageId" : { "Fn::FindInMap" : [ "AWSRegionArch2AMI", { "Ref" : "AWS::Region" },
                  { "Fn::FindInMap" : [ "AWSInstanceType2Arch", { "Ref" : "InstanceType" }, "Arch" ] } ] },
      "KeyName" : { "Ref" : "KeyName" },
      "UserData" : { "Fn::Base64" : { "Fn::Join" : ["", [
        "#!/bin/bash\n",
        "/opt/aws/bin/cfn-init --region ", { "Ref" : "AWS::Region" },
        "     -s ", { "Ref" : "AWS::StackName" }, " -r PuppetMasterInstance ",
        "     --access-key ", { "Ref" : "CFNKeys" },
        "     --secret-key ", { "Fn::GetAtt" : ["CFNKeys", "SecretAccessKey"]}, "\n",
        "/opt/aws/bin/cfn-signal -e $? '", { "Ref" : "PuppetMasterWaitHandle" }, "'\n"]]}}
    }
```

```
    },

    "EC2SecurityGroup" : {
      "Type" : "AWS::EC2::SecurityGroup",
      "Properties" : {
        "GroupDescription" : "Group for clients to communicate with Puppet Master"
      }
    },

    "PuppetGroup" : {
      "Type" : "AWS::EC2::SecurityGroup",
      "Properties" : {
        "GroupDescription" : "Group for puppet communication",
        "SecurityGroupIngress" : [
          { "IpProtocol" : "tcp", "FromPort" : "8140", "ToPort" : "8140", "SourceSecurityGroupName" : { "Ref" : "EC2SecurityGroup" }},
          { "IpProtocol" : "tcp", "FromPort": "22", "ToPort": "22", "CidrIp": "0.0.0.0/0" }
        ]
      }
    },

    "PuppetMasterWaitHandle" : {
      "Type" : "AWS::CloudFormation::WaitConditionHandle"
    },

    "PuppetMasterWaitCondition" : {
      "Type" : "AWS::CloudFormation::WaitCondition",
      "DependsOn" : "PuppetMasterInstance",
      "Properties" : {
        "Handle" : { "Ref" : "PuppetMasterWaitHandle" },
        "Timeout" : "600"
      }
    }
  },

  "Outputs" : {
    "PuppetMasterDNSName" : {
      "Value" : { "Fn::GetAtt" : [ "PuppetMasterInstance", "PrivateDnsName" ] },
      "Description" : "DNS Name of PuppetMaster"
    },
    "PuppetClientSecurityGroup" : {
      "Value" : { "Ref" : "EC2SecurityGroup" },
      "Description" : "Clients of the Puppet Master should be part of this security group"
    }
  }
}
```

Things to note about the template:

- The Puppet Master is available in the Amazon Linux Yum repository.
- The Amazon Linux AMI comes with *Extra Packages for Enterprise Linux (EPEL)* pre-installed, but not enabled. The template enables it by creating the file */etc/yum.repos.d/epel.repo*.
- The template has 2 parameters *ContentLocation* and *ContentManifest*. These parameters are used to populate the Puppet master with a set of modules or application and OS component configurations that can be downloaded to Puppet clients. The *ContentLocation* parameters points to an archive (.zip or tar'd and zipped) file that contains the manifests, templates and other artifacts making up the modules. The *ContentManifest* parameter contains the mapping between server roles and the modules needed.

  By default, the *ContentLocation* points to a sample archive that contains a module to install a WordPress application that makes use of an Amazon RDS database. The *ContentManifest* maps the Wordpress role to the Wordpress module as follows:

  ```
  /wordpress/: { include wordpress }
  ```

- The template can be run in any region, with any instance type being used to host the Puppet Master. Mappings are used to define the architecture of the instance and to select the correct EC2 AMI based on architecture and region.

- The template defines an IAM user that only has permissions to call the CloudFormation DescribeStackResource API. This is passed to the Cloud-init script for the Puppet Master EC2 instance.
- The Cloud-init script calls *cfn-init* to install the Puppet Master and populate it with an initial set of modules and then signals completion when the Puppet Master is ready to accept requests from Puppet clients.
- A Facter plug-in is also deployed to the Puppet Master at */etc/puppet/modules/cfn*. As we shall see later in this section, the Facter plug-in allows the client template to specify the server role and any properties needed by the modules in template metadata. From nodes.pp, you can see that the Facter plug-in is installed on all clients.
- The Puppet Master is locked down so that only instances in the EC2SecurityGroup can access it.
- The template returns both the Puppet Master DNS name and the EC2 security group for clients via outputs.

Facter plug-in

Facter is a Puppet plug-in that collects attributes about the operating system or other environmental aspects of the client host. These attributes can be used to influence how modules are customized and installed. AWS CloudFormation provides a Facter plug-in that interprets the template metadata, enabling you to use template metadata to configure applications and roles deployed via Puppet.

```
# cfn.rb
require 'rubygems'
require 'json'
filename = "/var/lib/cfn-init/data/metadata.json"
if not File.exist?(filename)
   return
end
parsed = JSON.load(File.new(filename))
parsed.default = Hash.new
parsed[\"Puppet\"].each do |key, value|
   actual_value = value
   if value.is_a? Array
    actual_value = value.join(',')
   end
   Facter.add(\"cfn_\" + key) do
     setcode do
       actual_value
     end
   end
end
```

Once installed, you can define attributes in your template metadata as follows:

```
  "Resources": {
     :
    "PuppetClient": {
      "Type": "AWS::EC2::Instance",
      "Metadata" : {
        "AWS::CloudFormation::Init" : {
```

```
          "config" : {
            :
          }
        },
        "Puppet" : {
          "roles"    : [ "wordpress" ],
          "host"     : {"Fn::GetAtt" : ["WordPressDatabase", "Endpoint.Address"]},
          "database" : "WordPressDB",
          "user"     : {"Ref" : "DatabaseUser"},
          "password" : {"Ref" : "DatabasePassword" }
        }
      },
      "Properties": {
        :
      }
    },
```

The roles key is interpreted in the */etc/puppet/manifests/nodes.pp* file to map the role to a set of modules. Your Puppet module templates can then make use of the other metadata values as follows:

```
<?php
define('DB_NAME', '<%= cfn_database %>');
define('DB_USER', '<%= cfn_user %>');
define('DB_PASSWORD', '<%= cfn_password %>');
define('DB_HOST', '<%= cfn_host %>');
:
```

By using the template metadata in this way, you can boot a base Amazon Linux AMI, bootstrap the Puppet client, configure one or more roles for the client and install and run the appropriate software packages all without manual intervention.

Installing the Puppet Client

That brings us to the Puppet Client. The following template creates a WordPress installation using an RDS database instance as the datastore.

```
{
  "AWSTemplateFormatVersion" : "2010-09-09",

  "Description": "Sample template to bring up WordPress using the Puppet client to install server roles. A WaitCondition is used to hold up the stack
creation until the application is deployed. **WARNING** This template creates one or more Amazon EC2 instances and CloudWatch alarms. You will be billed
for the AWS resources used if you create a stack from this template.",

  "Parameters" : {
    "KeyName": {
      "Type": "String",
      "Description" : "Name of an existing EC2 KeyPair to enable SSH access to the web server"
    },
    "PuppetClientSecurityGroup": {
      "Description" : "The EC2 security group for the instances",
      "Type": "String"
    },
    "PuppetMasterDNSName": {
      "Description" : "The PuppetMaster DNS name",
      "Type": "String"
    },
    "InstanceType": {
      "Default": "m1.small",
      "Description" : "Type of EC2 instance for web server",
      "Type": "String",
      "AllowedValues" : [ "t1.micro", "m1.small", "m1.large", "m1.xlarge", "m2.xlarge", "m2.2xlarge", "m2.4xlarge", "c1.medium", "c1.xlarge",
"cc1.4xlarge" ],
      "ConstraintDescription" : "must contain only alphanumeric characters."
```

```
    },
    "DatabaseType": {
      "Default": "db.m1.small",
      "Description" : "The database instance type",
      "Type": "String",
      "AllowedValues" : [ "db.m1.small", "db.m1.large", "db.m1.xlarge", "db.m2.xlarge", "db.m2.2xlarge", "db.m2.4xlarge" ],
      "ConstraintDescription" : "must contain only alphanumeric characters."
    },
    "DatabaseUser": {
      "Default" : "admin",
      "NoEcho": "true",
      "Type": "String",
      "Description" : "Test database admin account name",
      "MinLength": "1",
      "MaxLength": "16",
      "AllowedPattern" : "[a-zA-Z][a-zA-Z0-9]*",
      "ConstraintDescription" : "must begin with a letter and contain only alphanumeric characters."
    },
    "DatabasePassword": {
      "Default" : "admin",
      "NoEcho": "true",
      "Type": "String",
      "Description" : "Test database admin account password",
      "MinLength": "1",
      "MaxLength": "41",
      "AllowedPattern" : "[a-zA-Z0-9]*",
      "ConstraintDescription" : "must contain only alphanumeric characters."
    }
  },

  "Mappings" : {
    "AWSInstanceType2Arch" : {
      "t1.micro"    : { "Arch" : "32" },
      "m1.small"    : { "Arch" : "32" },
      "m1.large"    : { "Arch" : "64" },
      "m1.xlarge"   : { "Arch" : "64" },
      "m2.xlarge"   : { "Arch" : "64" },
      "m2.2xlarge"  : { "Arch" : "64" },
      "m2.4xlarge"  : { "Arch" : "64" },
      "c1.medium"   : { "Arch" : "32" },
      "c1.xlarge"   : { "Arch" : "64" },
      "cc1.4xlarge" : { "Arch" : "64" }
    },
    "AWSRegionArch2AMI" : {
      "us-east-1"      : { "32" : "ami-7f418316", "64" : "ami-7341831a" },
      "us-west-1"      : { "32" : "ami-951945d0", "64" : "ami-971945d2" },
      "eu-west-1"      : { "32" : "ami-24506250", "64" : "ami-20506254" },
      "ap-southeast-1" : { "32" : "ami-74dda626", "64" : "ami-7edda62c" },
      "ap-northeast-1" : { "32" : "ami-dcfa4edd", "64" : "ami-e8fa4ee9" }
    }
  },

  "Resources" : {

    "CFNInitUser" : {
      "Type" : "AWS::IAM::User",
      "Properties" : {
        "Policies": [{
          "PolicyName": "AccessForCFNInit",
          "PolicyDocument" : {
            "Statement": [{
              "Effect"   : "Allow",
              "Action"   : "cloudformation:DescribeStackResource",
              "Resource" : "*"
            }]
          }
        }]
      }
    },

    "CFNKeys" : {
      "Type" : "AWS::IAM::AccessKey",
      "Properties" : {
        "UserName" : { "Ref": "CFNInitUser" }
      }
    },

    "WebServer": {
      "Type": "AWS::EC2::Instance",
      "Metadata" : {
        "AWS::CloudFormation::Init" : {
          "config" : {
            "packages" : {
              "yum" : {
                "puppet"     : [],
                "ruby-devel" : [],
                "gcc"        : [],
                "make"       : [],
                "rubygems"   : []
              },
              "rubygems" : {
                "json"       : []
```

```
            }
          },
          "files" : {
            "/etc/yum.repos.d/epel.repo" : {
              "source" : "https://s3.amazonaws.com/cloudformation-examples/enable-epel-on-amazon-linux-ami",
              "mode"   : "000644",
              "owner"  : "root",
              "group"  : "root"
            },
            "/etc/puppet/puppet.conf" : {
              "content" : { "Fn::Join" : ["", [
                "[main]\n",
                "   logdir=/var/log/puppet\n",
                "   rundir=/var/run/puppet\n",
                "   ssldir=$vardir/ssl\n",
                "   pluginsync=true\n",
                "[agent]\n",
                "   classfile=$vardir/classes.txt\n",
                "   localconfig=$vardir/localconfig\n",
                "   server=",{ "Ref" : "PuppetMasterDNSName" },"\n"
              ]] },
              "mode" : "000644",
              "owner" : "root",
              "group" : "root"
            }
          },
          "services" : {
            "sysvinit" : {
              "puppet" : {
                "enabled" : "true",
                "ensureRunning" : "true"
              }
            }
          }
        }
      },
      "Puppet" : {
        "roles"    : [ "wordpress" ],
        "host"     : {"Fn::GetAtt" : ["WordPressDatabase", "Endpoint.Address"]},
        "database" : "WordPressDB",
        "user"     : {"Ref" : "DatabaseUser"},
        "password" : {"Ref" : "DatabasePassword" }
      }
    },
    "Properties": {
      "SecurityGroups": [ { "Ref": "PuppetClientSecurityGroup" }, { "Ref" : "EC2SecurityGroup" } ],
      "ImageId": { "Fn::FindInMap": [ "AWSRegionArch2AMI", { "Ref": "AWS::Region" }, { "Fn::FindInMap": [ "AWSInstanceType2Arch", { "Ref":
"InstanceType" }, "Arch" ] } ]
      },
      "UserData" : { "Fn::Base64" : { "Fn::Join" : ["", [
        "#!/bin/bash\n",
        "/opt/aws/bin/cfn-init --region ", { "Ref" : "AWS::Region" },
        "    -s ", { "Ref" : "AWS::StackName" }, " -r WebServer ",
        "    --access-key ", { "Ref" : "CFNKeys" },
        "    --secret-key ", { "Fn::GetAtt" : ["CFNKeys", "SecretAccessKey"]}, "\n",
        "/opt/aws/bin/cfn-signal -e $? '", { "Ref" : "ApplicationWaitHandle" }, "'\n"
      ]]}},
      "KeyName": { "Ref": "KeyName" },
      "InstanceType": { "Ref": "InstanceType" }
    }
  },


  "EC2SecurityGroup" : {
    "Type" : "AWS::EC2::SecurityGroup",
    "Properties" : {
      "GroupDescription" : "Enable HTTP access for Wordpress plus SSH access via port 22",
      "SecurityGroupIngress" : [
        {"IpProtocol" : "tcp", "FromPort" : "22", "ToPort" : "22", "CidrIp" : "0.0.0.0/0" },
        {"IpProtocol" : "tcp", "FromPort" : "80", "ToPort" : "80", "CidrIp" : "0.0.0.0/0" }
      ]
    }
  },

  "ApplicationWaitHandle" : {
    "Type" : "AWS::CloudFormation::WaitConditionHandle"
  },

  "ApplicationWaitCondition" : {
    "Type" : "AWS::CloudFormation::WaitCondition",
    "DependsOn"  : "WebServer",
    "Properties" : {
      "Handle"   : { "Ref" : "ApplicationWaitHandle" },
      "Timeout"  : "600"
    }
  },

  "WordPressDatabase" : {
    "Type" : "AWS::RDS::DBInstance",
    "Properties" : {
      "AllocatedStorage"   : "5",
      "DBName"             : "WordPressDB",
```

```
        "Engine"             : "MySQL",
        "DBInstanceClass"    : { "Ref" : "DatabaseType" },
        "DBSecurityGroups"   : [ { "Ref": "DBSecurityGroup" } ],
        "MasterUsername"     : { "Ref" : "DatabaseUser" },
        "MasterUserPassword" : { "Ref" : "DatabasePassword" }
      }
    },

    "DBSecurityGroup": {
      "Type": "AWS::RDS::DBSecurityGroup",
      "Properties": {
        "DBSecurityGroupIngress": {
          "EC2SecurityGroupName": { "Ref": "EC2SecurityGroup" }
        },
        "GroupDescription": "database access"
      }
    }
  },

  "Outputs": {
    "WebsiteURL": {
      "Value": { "Fn::Join": [ "", [ "http://", { "Fn::GetAtt": [ "WebServer", "PublicDnsName" ] }, "/wordpress" ] ] },
      "Description" : "URL of the WordPress website"
    },
    "InstallURL": {
      "Value": { "Fn::Join": [ "", [ "http://", { "Fn::GetAtt": [ "WebServer", "PublicDnsName" ] }, "/wordpress/wp-admin/install.php" ] ] },
      "Description" : "URL to install WordPress"
    }
  }
}
```

Things to note about the template:

- The Puppet Master is installed from the Amazon Linux Yum repository.
- The *PuppetMasterDNSName* and *PuppetClientSecurityGroup* parameters are used to locate and provide access to the Puppet Master.
- The template can be run in any region, with any instance type being used to host the WordPress installation. Mappings are used to define the architecture of the instance and to select the correct EC2 AMI based on architecture and region.
- The template defines an IAM user that only has permissions to call the CloudFormation DescribeStackResource API. This is passed to the Cloud-init script for the WebServer Amazon EC2 instance.
- Template metadata defines the Wordpress role and parameters needed to configure WordPress for this installation.
- Both the initial WordPress install URL and the subsequent WordPress website URL are returned using template outputs.

Abstracting out the "muck" to create a re-usable Puppet Client template

With AWS CloudFormation embedded templates, it is possible to create a re-usable template that can bootstrap the Puppet client, making it simple to use Puppet to deploy your applications. To do this, we will split up the previous example into common pieces needed to bootstrap the Puppet client and pieces needed specifically to deploy WordPress.

The following is a re-usable template that creates an Amazon EC2 instance running the Puppet client that can be configured to install a server role from the Puppet Master:

```
{
  "AWSTemplateFormatVersion" : "2010-09-09",

  "Description": "Sample template to use the PuppetLabs Puppet client to install server roles. A WaitCondition is used to hold up the stack creation
until the application is deployed. **WARNING** This template creates one or more Amazon EC2 instances and CloudWatch alarms. You will be billed for the
AWS resources used if you create a stack from this template.",

  "Parameters" : {
    "KeyName": {
      "Type": "String",
      "Description" : "Name of an existing EC2 KeyPair to enable SSH access to the web server"
```

```
    },
    "EC2SecurityGroup": {
      "Default": "default",
      "Description" : "The EC2 security group that contains instances that need access to the database",
      "Type": "String"
    },
    "StackNameOrId" : {
      "Description" : "The StackName or StackId containing the Puppet configuration metadata",
      "Type": "String",
      "MinLength": "1",
      "MaxLength": "128",
      "AllowedPattern" : "[a-zA-Z][a-zA-Z0-9]*"
    },
    "ResourceName" : {
      "Description" : "The Logical Resource Name in the stack defined by StackName containing the resource with the Puppet configuration metadata",
      "Type": "String",
      "MinLength": "1",
      "MaxLength": "128",
      "AllowedPattern" : "[a-zA-Z][a-zA-Z0-9]*"
    },
    "PuppetClientSecurityGroup": {
      "Description" : "The EC2 security group for the instances",
      "Type": "String"
    },
    "PuppetMasterDNSName": {
      "Description" : "The PuppetMaster DNS name",
      "Type": "String"
    },
    "InstanceType": {
      "Default": "m1.small",
      "Description" : "Type of EC2 instance for web server",
      "Type": "String",
      "AllowedValues" : [ "t1.micro", "m1.small", "m1.large", "m1.xlarge", "m2.xlarge", "m2.2xlarge", "m2.4xlarge", "c1.medium", "c1.xlarge",
"cc1.4xlarge" ],
      "ConstraintDescription" : "must contain only alphanumeric characters."
    }
  },

  "Mappings" : {
    "AWSInstanceType2Arch" : {
      "t1.micro"    : { "Arch" : "32" },
      "m1.small"    : { "Arch" : "32" },
      "m1.large"    : { "Arch" : "64" },
      "m1.xlarge"   : { "Arch" : "64" },
      "m2.xlarge"   : { "Arch" : "64" },
      "m2.2xlarge"  : { "Arch" : "64" },
      "m2.4xlarge"  : { "Arch" : "64" },
      "c1.medium"   : { "Arch" : "32" },
      "c1.xlarge"   : { "Arch" : "64" },
      "cc1.4xlarge" : { "Arch" : "64" }
    },
    "AWSRegionArch2AMI" : {
      "us-east-1"      : { "32" : "ami-7f418316", "64" : "ami-7341831a" },
      "us-west-1"      : { "32" : "ami-951945d0", "64" : "ami-971945d2" },
      "eu-west-1"      : { "32" : "ami-24506250", "64" : "ami-20506254" },
      "ap-southeast-1" : { "32" : "ami-74dda626", "64" : "ami-7edda62c" },
      "ap-northeast-1" : { "32" : "ami-dcfa4edd", "64" : "ami-e8fa4ee9" }
    }
  },

  "Resources" : {

    "CFNInitUser" : {
      "Type" : "AWS::IAM::User",
      "Properties" : {
        "Policies": [{
          "PolicyName": "AccessForCFNInit",
          "PolicyDocument" : {
            "Statement": [{
              "Effect"   : "Allow",
              "Action"   : "cloudformation:DescribeStackResource",
              "Resource" : "*"
            }]
          }
        }]
      }
    },

    "CFNKeys" : {
      "Type" : "AWS::IAM::AccessKey",
      "Properties" : {
        "UserName" : { "Ref": "CFNInitUser" }
      }
    },

    "PuppetClient": {
      "Type": "AWS::EC2::Instance",
      "Metadata" : {
        "AWS::CloudFormation::Init" : {
          "config" : {
            "packages" : {
              "yum" : {
```

```
              "puppet"     : [],
              "ruby-devel" : [],
              "gcc"        : [],
              "make"       : [],
              "rubygems"   : []
            },
            "rubygems" : {
              "json"       : []
            }
          },
          "files" : {
            "/etc/yum.repos.d/epel.repo" : {
              "source" : "https://s3.amazonaws.com/cloudformation-examples/enable-epel-on-amazon-linux-ami",
              "mode"   : "000644",
              "owner"  : "root",
              "group"  : "root"
            },
            "/etc/puppet/puppet.conf" : {
              "content" : { "Fn::Join" : ["", [
                "[main]\n",
                "   logdir=/var/log/puppet\n",
                "   rundir=/var/run/puppet\n",
                "   ssldir=$vardir/ssl\n",
                "   pluginsync=true\n",
                "[agent]\n",
                "   classfile=$vardir/classes.txt\n",
                "   localconfig=$vardir/localconfig\n",
                "   server=",{ "Ref" : "PuppetMasterDNSName" },"\n"
              ]] },
              "mode" : "000644",
              "owner" : "root",
              "group" : "root"
            }
          },
          "services" : {
            "sysvinit" : {
              "puppet" : {
                "enabled" : "true",
                "ensureRunning" : "true"
              }
            }
          }
        }
      }
      },
      "Properties": {
        "SecurityGroups": [ { "Ref": "PuppetClientSecurityGroup" }, { "Ref" : "EC2SecurityGroup" } ],
        "ImageId": { "Fn::FindInMap": [ "AWSRegionArch2AMI", { "Ref": "AWS::Region" },
                 { "Fn::FindInMap": [ "AWSInstanceType2Arch", { "Ref": "InstanceType" }, "Arch" ] } ]
        },
        "UserData" : { "Fn::Base64" : { "Fn::Join" : ["", [
            "#!/bin/bash -v\n",

            "function error_exit\n",
            "{\n",
            "  cfn-signal -e 1 -r \"$1\" '", { "Ref" : "ApplicationWaitHandle" }, "'\n",
            "}\n",

            "/opt/aws/bin/cfn-init --region ", { "Ref" : "AWS::Region" },
            "    -s ", { "Ref" : "AWS::StackName" }, " -r PuppetClient ",
            "    --access-key ", { "Ref" : "CFNKeys" },
            "    --secret-key ", { "Fn::GetAtt" : ["CFNKeys", "SecretAccessKey"]}, " || error_exit 'Failed to initialize Puppet client using cfn-
init'\n",

            "/opt/aws/bin/cfn-init --region ", { "Ref" : "AWS::Region" },
            "    -s ", { "Ref" : "StackNameOrId" }, " -r ", { "Ref" : "ResourceName" },
            "    --access-key ", { "Ref" : "CFNKeys" },
            "    --secret-key ", { "Fn::GetAtt" : ["CFNKeys", "SecretAccessKey"]}, " || error_exit 'Failed to initialize server role using cfn-init'\n",

            "/opt/aws/bin/cfn-signal -e $? '", { "Ref" : "ApplicationWaitHandle" }, "'\n"
        ]]}},
        "KeyName": { "Ref": "KeyName" },
        "InstanceType": { "Ref": "InstanceType" }
      }
    },

    "ApplicationWaitHandle" : {
      "Type" : "AWS::CloudFormation::WaitConditionHandle"
    },

    "ApplicationWaitCondition" : {
      "Type" : "AWS::CloudFormation::WaitCondition",
      "DependsOn"  : "PuppetClient",
      "Properties" : {
        "Handle"   : { "Ref" : "ApplicationWaitHandle" },
        "Timeout"  : "9000"
      }
    }
  },

  "Outputs": {
    "ServerDNSName": {
```

```
      "Value": { "Fn::GetAtt": [ "PuppetClient", "PublicDnsName" ] },
      "Description" : "Public DNS name of new server"
    }
  }
}
```

Things to note about the template:

- While this template creates a single instance server with the Puppet Client installed, you could create a building-block template using these same techniques to create a highly available, multi-AZ application by using an Auto Scaling group rather than a single Amazon EC2 instance. The chef-solo building block template shown previous shows how you might do that.
- The template takes 2 input parameters *PuppetClientSecurityGroup* and *PuppetMasterDNSName* to identify and access the Puppet Master. This assumes the Puppet Master is configured either using the template described earlier in this section or in a similar way.
- The template takes 2 input parameters *StackNameOrId* and *ResourceName* that are used to reference the server roles defined in resource metadata in the calling template. For details, see the WordPress example below.
- The template can be run in any region, with any instance type being used to host the application. Mappings are used to define the architecture of the instance and to select the correct Amazon EC2 AMI based on architecture and region.
- The template defines an IAM user that only has permissions to call the CloudFormation DescribeStackResource API. This is passed to the Cloud-init script for the Amazon EC2 instance.
- The PuppetClient metadata defines all of the packages necessary to run, deploy and configure the Puppet client software to access the Puppet Master.
- The Cloud-init script calls *cfn-init* twice, the first time using the metadata defined in this template to install the Puppet client and a second time to configure the roles defined in the calling template.
- A WaitCondition is used to delay the stack being created until the Puppet client is configured and running.
- The public DNS name of the Amazon EC2 instance is returned via the template output values.

The following template shows how you can use this (and an RDS best practices template) to install and configure WordPress (assuming the roles is predefined in the Puppet Master).

```
{
  "AWSTemplateFormatVersion": "2010-09-09",

  "Description": "This template demonstrates using embedded templates to build an end to end solution from basic building blocks. It builds a WordPress
installation using an RDS database backend configured via a Puppet Master. **WARNING** This template creates one or more Amazon EC2 instances and
CloudWatch alarms. You will be billed for the AWS resources used if you create a stack from this template.",

  "Parameters": {
    "KeyName": {
      "Type": "String",
      "Description" : "Name of an existing EC2 KeyPair to enable SSH access to the web server"
    },
    "PuppetClientSecurityGroup": {
      "Description" : "The EC2 security group for the instances",
      "Type": "String"
    },
    "PuppetMasterDNSName": {
      "Description" : "The PuppetMaster DNS name",
      "Type": "String"
    },
```

```
    "InstanceType": {
      "Default": "m1.small",
      "Description" : "Type of EC2 instance for web server",
      "Type": "String",
      "AllowedValues" : [ "t1.micro", "m1.small", "m1.large", "m1.xlarge", "m2.xlarge", "m2.2xlarge", "m2.4xlarge", "c1.medium", "c1.xlarge",
"cc1.4xlarge" ],
      "ConstraintDescription" : "must be a valid EC2 instance type."
    },
    "DatabaseType": {
      "Default": "db.m1.small",
      "Description" : "The database instance type",
      "Type": "String",
      "AllowedValues" : [ "db.m1.small", "db.m1.large", "db.m1.xlarge", "db.m2.xlarge", "db.m2.2xlarge", "db.m2.4xlarge" ],
      "ConstraintDescription" : "must be a valid RDS DB Instance type."
    },
    "DatabaseUser": {
      "Default" : "admin",
      "NoEcho": "true",
      "Type": "String",
      "Description" : "Test database admin account name",
      "MinLength": "1",
      "MaxLength": "16",
      "AllowedPattern" : "[a-zA-Z][a-zA-Z0-9]*",
      "ConstraintDescription" : "must begin with a letter and contain only alphanumeric characters."
    },
    "DatabasePassword": {
      "Default" : "admin",
      "NoEcho": "true",
      "Type": "String",
      "Description" : "Test database admin account password",
      "MinLength": "1",
      "MaxLength": "41",
      "AllowedPattern" : "[a-zA-Z0-9]*",
      "ConstraintDescription" : "must contain only alphanumeric characters."
    },
    "OperatorEmail": {
      "Description": "EMail address to notify if there are operational issues",
      "Type": "String"
    }
  },

  "Mappings" : {
    "RegionMap" : {
      "us-east-1" :      { "s3Bucket" : "https://s3.amazonaws.com/cloudformation-templates-us-east-1" },
      "us-west-1" :      { "s3Bucket" : "https://s3.amazonaws.com/cloudformation-templates-us-west-1" },
      "eu-west-1" :      { "s3Bucket" : "https://s3.amazonaws.com/cloudformation-templates-eu-west-1" },
      "ap-northeast-1" : { "s3Bucket" : "https://s3.amazonaws.com/cloudformation-templates-ap-northeast-1" },
      "ap-southeast-1" : { "s3Bucket" : "https://s3.amazonaws.com/cloudformation-templates-ap-southeast-1" }
    }
  },

  "Resources" : {

    "AlarmTopic" : {
      "Type" : "AWS::SNS::Topic",
      "Properties" : {
        "Subscription" : [ { "Endpoint" : { "Ref": "OperatorEmail" }, "Protocol" : "email" } ]
      }
    },

    "EC2SecurityGroup" : {
      "Type" : "AWS::EC2::SecurityGroup",
      "Properties" : {
        "GroupDescription" : "Open up SSH and HTTP access",
        "SecurityGroupIngress" : [
          {"IpProtocol": "tcp", "FromPort": "22", "ToPort": "22", "CidrIp": "0.0.0.0/0" },
          {"IpProtocol": "tcp", "FromPort": "80", "ToPort": "80", "CidrIp": "0.0.0.0/0" }
        ]
      }
    },

    "WebServer" : {
      "Type" : "AWS::CloudFormation::Stack",
      "Metadata" : {
        "Puppet" : {
          "roles"    : [ "wordpress" ],
          "host"     : { "Fn::GetAtt" : [ "AppDatabase", "Outputs.DBAddress" ] },
          "database" : "WordPressDB",
          "user"     : {"Ref" : "DatabaseUser"},
          "password" : {"Ref" : "DatabasePassword" }
        }
      },
      "Properties" : {
        "TemplateURL" : { "Fn::Join" : ["/", [{ "Fn::FindInMap" : [ "RegionMap", { "Ref" : "AWS::Region" }, "s3Bucket" ]},
                          "puppet-client-configuration.template" ]]},
        "Parameters" : {
          "KeyName"             : { "Ref" : "KeyName" },
          "InstanceType"        : { "Ref" : "InstanceType"},
          "EC2SecurityGroup"    : { "Ref" : "EC2SecurityGroup" },
          "PuppetClientSecurityGroup" : { "Ref" : "PuppetClientSecurityGroup" },
          "PuppetMasterDNSName" : { "Ref" : "PuppetMasterDNSName" },
          "StackNameOrId"       : { "Ref" : "AWS::StackName" },
```

```
          "ResourceName"        : "WebServer"
        }
      }
    },

    "AppDatabase" : {
      "Type" : "AWS::CloudFormation::Stack",
      "Metadata" : {
          "Comment" : "Application database."
      },
      "Properties" : {
        "TemplateURL" : { "Fn::Join" : ["/", [{ "Fn::FindInMap" : [ "RegionMap", { "Ref" : "AWS::Region" }, "s3Bucket" ]},
                          "RDS_MySQL_55.template" ]]},
        "Parameters" : {
          "DBName"           : "WordPressDB",
          "DBUser"           : { "Ref" : "DatabaseUser" },
          "DBPassword"       : { "Ref" : "DatabasePassword" },
          "DBInstanceClass"  : { "Ref" : "DatabaseType" },
          "AlarmTopic"       : { "Ref" : "AlarmTopic" },
          "EC2SecurityGroup" : { "Ref" : "EC2SecurityGroup" },
          "MultiAZ"          : "false"
        }
      }
    }
  },

  "Outputs": {
    "URL": {
      "Value": { "Fn::Join" : ["", [ "http://", { "Fn::GetAtt": [ "WebServer", "Outputs.ServerDNSName" ] }, "/wordpress" ]]},
      "Description" : "URL of the website"
    }
  }
}
```

Things to note from this template:

- There are only 2 main resources defined in the template: A WebServer configured using the Puppet client building block template defined previously and an RDS database defined using a pre-existing RDS database best practice configuration template that includes alarming configuration.
- The WebServer defines metadata to define and configure the WordPress role for this WebServer.

Getting the AWS CloudFormation helper scripts and templates

The AWS CloudFormation helper scripts are available from the following locations:

- The latest version of the Amazon Linux AMI has the AWS CloudFormation helper scripts installed by default in */opt/aws/bin.*
- The AWS helper scripts are available in the Amazon Linux Yum repository (the package name is *aws-cfn-bootstrap*) for previous versions of the Amazon Linux AMI.
- In addition, the helpers are available in other formats:
  - *https://s3.amazonaws.com/cloudformation-examples/aws-cfn-bootstrap-1.0-4.noarch.rpm*
  - *https://s3.amazonaws.com/cloudformation-examples/aws-cfn-bootstrap-1.0.tar.gz* to install the helper scripts via the Python *easy-install* tools.
  - *https://s3.amazonaws.com/cloudformation-examples/aws-cfn-bootstrap-1.0.zip*
  - *https://s3.amazonaws.com/cloudformation-examples/aws-cfn-bootstrap-1.0.win32.msi* for installation on Windows.
- The source for the scripts is available at *https://s3.amazonaws.com/cloudformation-examples/aws-cfn-bootstrap-1.0-4.src.rpm*

- The Facter plugin used in this document is available at https://s3.amazonaws.com/cloudformation-examples/cfn-facter-plugin.rb
- The templates used in this document are available, along with many other sample templates, on the AWS CloudFormation sample template site at http://aws.amazon.com/cloudformation/aws-cloudformation-templates/