

AWS GoldBase Implementation Guide

October 2015



© 2015, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Notices

This document is provided for informational purposes only. It represents AWS's current product offerings and practices as of the date of issue of this document, which are subject to change without notice. Customers are responsible for making their own independent assessment of the information in this document and any use of AWS's products or services, each of which is provided "as is" without warranty of any kind, whether express or implied. This document does not create any warranties, representations, contractual commitments, conditions or assurances from AWS, its affiliates, suppliers or licensors. The responsibilities and liabilities of AWS to its customers are controlled by AWS agreements, and this document is not part of, nor does it modify, any agreement between AWS and its customers.

Contents

AWS GoldBase Overview	4
AWS GoldBase Solution	4
How can GoldBase be used?	4
What are CloudFormation Templates	4
What is an AWS CloudFormation Stack?	5
Source Management	7
Assumptions	8
Technical Prerequisites	8
Deployment Scenarios	10
Quick Start Deployment: Common Issues	11
Uploading the Templates to S3	12
Quick Start Deployment: CLI	12
Quick Start Deployment: Console	15
Deleting a Stack	18
Integrating with AWS Service Catalog	19
Adding Users	19
Adding Constraints	20
Adding Tags	20
Service Catalog End Users	21
Appendix	23
Special Considerations	23
Bootstrapping Configuration Management	23
Working with the <i>aws-cf-tools</i> Package	29
Adding EC2 Instances	31

AWS GoldBase Overview

AWS GoldBase is a solution to enable customers to streamline, automate, and implement secure baselines in AWS, from initial design to operational security readiness. AWS GoldBase incorporates the expertise of AWS solutions architects and security and compliance personnel to build a secure and reliable architecture in an easy-to-implement solution through automation.

AWS GoldBase Solution

The AWS GoldBase solution includes the following items for customer use:

- Security Control Responsibility Matrix (CRM)
- Architecture diagrams
- AWS CloudFormation templates
- User Guides and deployment instructions

How can GoldBase be used?

You can build a “golden environment” that fits your requirements and implementation: define the configuration you require in the form of AWS configuration values, such as encryption requirements (forcing server side encryption for S3 objects); permissions to resources (which roles apply to certain environments); which compute images are authorized (based on gold images of servers you have authorized); and what kind of logging needs to be enabled (such as enforcing the use of CloudTrail on all resources).

As AWS provides a mature set of configuration options (and new services are being released all the time), we provide some templates for you to leverage for your own environment. These security templates ([in the form of AWS CloudFormation Templates](#)) provide a more comprehensive rule set that can be systematically enforced.

What are CloudFormation Templates?

A CloudFormation template is simply a JSON (JavaScript Object Notation)-formatted text file that describes the AWS infrastructure needed to run an application or service along with any interconnection between them. These

template packages deploy a base stack of resources, such as a three-tiered Virtual Private Network (VPC) structure, Identity Access Management (IAM) configuration, and S3 buckets and policies. EC2 and RDS instances can then be deployed into this base architecture while making use of standardized security groups, IAM policies, and other resources. The templates are designed to deploy architecture that is in alignment with AWS leading practices, as well as the identified security framework.

What is an AWS CloudFormation Stack?

When you use AWS CloudFormation, you manage related resources as a single unit called a [stack](#). In other words, you create, update, and delete a collection of resources by creating, updating, and deleting stacks. All the resources in a stack are defined by the stack's AWS CloudFormation template.

To update resources, you first modify the stack templates and then update your stack by submitting the modified template. You can work with stacks by using the AWS CloudFormation [console](#), [API](#), or [AWS CLI](#).

Stack Deployment

The template package uses nested CloudFormation templates. Each stack can be deployed independently, if needed, by specifying the required parameters for each upon deployment. To deploy the full package, the IAM user must have permissions to deploy each of the resources it creates, including IAM configuration for groups and roles.

Stacks included in the solution

There are four (4) CloudFormation template files included, as well as a master template, which can be used to deploy the full stack version deploying each individual stack in this solution:

Template	Description	Dependencies
Stack 1 – Access	Enables CloudTrail, S3 buckets and IAM settings for S3 bucket access. Creates IAM Roles and Groups.	None
Stack 2 – Network	Three tier VPCs (Management, Development and Production), Subnets, Gateways, Route Tables, NACLs	None
Stack 3 – Security	Elastic Load Balancers, S3 Buckets Policies, Security Groups, SNS, SQS, CloudWatch	None
Stack 4 – Server/Data	Instances – Proxy, WebApp and DB, or RDS DB, ELB, SNS Alarms for Cloudwatch, Auto Scaling Groups	Stack2 “Output” values
Main Stack	Primary template file which is used to deploy the full template package (multi-tier linux-based web application); passes parameters to nested templates	Stack 2 and Stack 3 “Output” values

This package includes the following directories and contents:

docs/	Documentation, SCTM, and this implementation guide
templates/	Template files for use case package
tools/	Tools and scripts useful in deployment or management
README.md	Readme file for basic information on this package

Source Management

It is expected that customers will be able to modify and update this package as needed over time. The AWS GoldBase archive can be managed on a source code repository such as Git, SVN, or by using [AWS Code Commit](#). This is recommended to ensure proper version control, developer collaboration, and documentation of updates.

Nested Stacks

This template relies on the use of nested [CloudFormation stacks](#). Stacks 1-4 JSON templates deploy the actual resources, with the main*.json templates specifying each of these template files as stack resources to launch. The main* templates are basically the entry points to launch the entire architecture. Main.json is used to launch the additional stacks and also allow parameters to be passed into each of the nested stacks.

To deploy the full package, the IAM user must have permissions to deploy each of the resources it creates, which includes IAM configuration for groups and roles.

Layered Deployments

To deploy the entire architecture (including IAM and VPC), use the **main-webapp-linux.json** template when launching the stacks. You can also launch main*.json and omit the S3 URLs for Stack3 and Stack4 to only deploy Stack1 and 2. This can be useful for provisioning teams who must deploy initial base architecture in accounts for application owners.

Additionally, each stack can be independently deployed. Keep in mind, in order to do this, individual parameters must be passed in upon launch of each template.

Technical Prerequisites

This implementation guide assumes the user has a moderate level of understanding of AWS. If not, there are several resources available which can help below.

In order to effectively use the AWS GoldBase package, the following minimum prerequisites are strongly recommended:

- Access to a current AWS account with IAM administrator-level access
- Basic understanding of AWS services and AWS CloudFormation
- Knowledge of architecting applications on AWS
- Understanding of security and compliance requirements in the customer organization

Recommended Training

AWS offers Training and Certification programs to help you develop skills to design, deploy, and operate infrastructure and applications on the AWS Cloud. Whether you are just getting started or looking to deepen your technical expertise, we have a variety of resources to meet your needs. Discover on-demand tools, technical classes, and certifications to help you on your journey with AWS.

Explore our offerings below. This information is also available as a PDF so you can download to read offline or print: [AWS Training and Certification Overview \(PDF\)](#).

Cost

CloudFormation itself is completely free, although the resources created by CloudFormation may have a cost associated with them. The basic deployment creates a minimum number of resources, which makes the cost of simply deploying this template very insignificant. In this architecture, the resources in `stack4` contribute to the most significant cost since this is where the compute resources and database are deployed.

AWS Service Catalog Deployment

The template package provided can be deployed as an AWS Service Catalog product. AWS Service Catalog enables a self-service model for deployment of applications and architecture in AWS. Service Catalog allows the creation of “portfolios” in which there can exist one or more products. These products are CloudFormation templates. IAM users/groups/roles can be granted access to specified portfolios, which they can then launch from a separate interface.

Deployment Scenarios

The following are potential deployment scenarios, which demonstrate how the included templates could be used:

Deployment	Description	Use Case
Full Deployment (all stacks)	Using the “main-webapp-linux.json” file, all stacks in package are deployed at once	Deployment of entire end-to-end architecture by administrator
Stacks 2,3,4 Only	Deployment of entire architecture except security/IAM Config (omit “stack1URL” parameter in main template)	Deployment of full application (including VPCs) into existing AWS account which already has security/IAM Config setup
Stack 1 Only	Individually deploy the stack1-security JSON template only	Initial setup of base IAM/security configuration for new AWS account (can be used with multiple apps)
Stack 1 and Stack 2 Only (or Stack2 Only)	Setup of existing architecture (VPC structure) for applications to be deployed into	Provisioning team (or Cloud Team) setting up VPCs for workload owners
Stack 3 and Stack 4	Templates available to workload owners with limited IAM access; templates specific to application	Workload owners deploy application who do not have permissions to create VPC or IAM configuration
Individual Stacks Only	Use each template file independently as needed	Can be reused independently or as part of another application in form of nested (or un-nested) stack

Quick Start Deployment: Common Issues

Before starting the Quick Start deployment, make note of these common issues:

CloudTrail	When launching Stack 1 (security stack 3) deployment may fail if CloudTrail is already enabled; ensure CloudTrail is not enabled or choose “no” to CreateCloudTrail parameter in template
Amazon Machine Images (AMI's)	AMI must exist in region or be accessible to account in the case of custom AMIs, otherwise errors may result on deployment of instances
Amazon Relational Database Service	RDS deployment may fail if DB/DB type does not support RDS encryption
Amazon Simple Storage Service (Amazon S3) URLs	Stack*URL parameters in main template must be valid and accessible locations of the nested templates stored on S3 buckets, or deployment will fail
AWS CLI (command line interface) Deployment	AWS Identity and Access Management (IAM) role the instance is launched into must have an IAM policy allowing access to resources stacks will deploy; if deploying stack1 this includes admin-level IAM permissions
AWS Management Console	You must be logged into the Management Console with IAM permissions for the resources and actions the templates deploy; if deploying stack1 you must have access to modify IAM configuration
S3 Buckets	If specifying S3 bucket name parameters, bucket names must be unique.

If unfamiliar with AWS CloudFormation, it can be useful to first review the [User Guide](#). An online lab listed on page 3 of this document can also help you become familiar with CloudFormation templates and deployment.

Uploading the Templates to S3

The main*.json template file lists S3 URLs for the nested stacks. You will want to upload these to an S3 bucket you have access to prior to deployment, and then modify these parameters in main-webapp-linux.json:

- Stack 1URL, Stack 2URL, Stack 3URL, Stack 4URL

This can be done through the console or the CLI.

Console:

1. Go to the CloudFormation Console, select a bucket which you want to store the templates
2. Click “Upload”, specify the local location of the file to upload
3. Upload all 4 template files in the same location
4. Make note of the URL of the files (by going to the “Properties” in the S3 bucket)

CLI

Use the following AWS CLI command to upload each template file:

```
aws cp <template file>.json s3://<s3bucketname>/
```

The AWS CLI package can be obtained here: <http://aws.amazon.com/cli/>

Quick Start Deployment: CLI

The Quick Start Deployment instructions below provide information on how to quickly test deploy the entire architecture (using nested stacks) in the AWS CLI/API. It references using the optional “cfdeploy” script included with the package. Alternatively [AWS CLI commands](#) can provide the same function if needed.

1. Launch an Amazon Elastic Compute Cloud ([Amazon EC2](#)) Linux instance in an existing VPC you have access to
 - Recommend using the [Amazon Linux AMI](#)

- The instance must be launched into an [IAM role](#) with “admin” rights

2. [Secure Shell \(SSH\) into the EC2](#) instance using the SSH key specified on launch in Step 1

- Install “git”

```
# sudo yum install git -y
```

3. Pull down the latest version of the package (Example)

- a. Option 1: Pull down from a Git repository

```
# sudo git clone git@<repository FQDN>: user/aws-goldbase.git
```

- b. Option 2: Pull down from an S3 URL link

```
# sudo wget <S3 URL>/aws-goldbase.tgz
```

```
# sudo tar xzf aws-goldbase.tgz
```

**aws-goldbase: modify the filename in these instructions if different in your package*

4. **Install the aws-cf-tools package**

```
# cd aws-goldbase
```

```
# sudo ./tools-install
```

5. **Edit the example parameters file**

- a. If deploying in us-west-1

```
# vi templates/parameters/example_uswest1.yaml
```

- b. If deploying in us-east-1*

```
# vi templates/parameters/example_useast1.yaml
```

** For us-east-1, please ensure you have access to the regions listed (us-east-1a, us-east-1c) in your AWS account dashboard and modify as needed*

Modify any parameters with <INPUT> to add appropriate values

s3bucketApplicationLogsName	Application Logs S3 bucket
S3bucketBackupName	(optional) S3 bucket for backup data
S3bucketWebContentName	(optional) S3 bucket for fixed/web content
Keyname	Existing SSH key in your account in region
DBName	Name to use for RDS DB
DBPassword	Password to use for RDS DB

Modify any other parameters as necessary for your specific deployment.

6. Launch the main.json template file using cfdeploy

a. If deploying in [identify region] e.g. us-east-1

```
# cfdeploy --deploy GoldBaseFull --yaml-parameters
templates/parameters/example_useast1.yaml --template templates/main-
webapp-linux.json --region us-east-1
```

b. If deploying us-west-1

```
# cfdeploy --deploy GoldBaseFull --yaml-parameters
templates/parameters/example_uswest1.yaml --template templates/main-
webapp-linux.json --region us-west-1
```

Quick Start Deployment: Console

The Quick Start Deployment instructions below provide information on how to quickly test deploy the entire architecture (using nested stacks) in the AWS console.

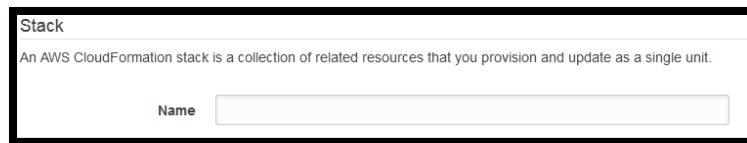
1. Login to the AWS Console and go to the AWS CloudFormation Console

<https://console.aws.amazon.com/cloudformation/>

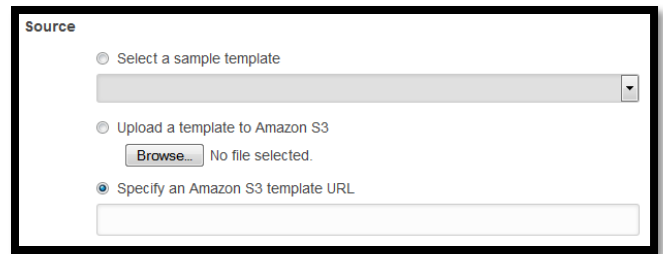
- a. “Create Stack”

2. On the Select Template page, enter the following required inputs:

- a. Provide a unique name to identify the stack to be launched



- b. Specify the source of the template as the S3 bucket URL where the **main.json** file is located.



- c. Click “Next” to Proceed

3. Enter the required parameters, and modify defaults as needed

- a. The following are required parameters. Those highlighted do not have default values that will automatically work and must be input or modified.

Resource Name	Description
AppInstanceType	Instance size to use for app tier
DBAllocatedStorage	Amount of storage in GB to allocate for RDS DB
DBClass	Size of RDS database
DBName	Name to use for RDS DB
DBPassword	Password for RDS DB
DBPrivateSubnetACIDR	CIDR block to use for RDS DB Subnet

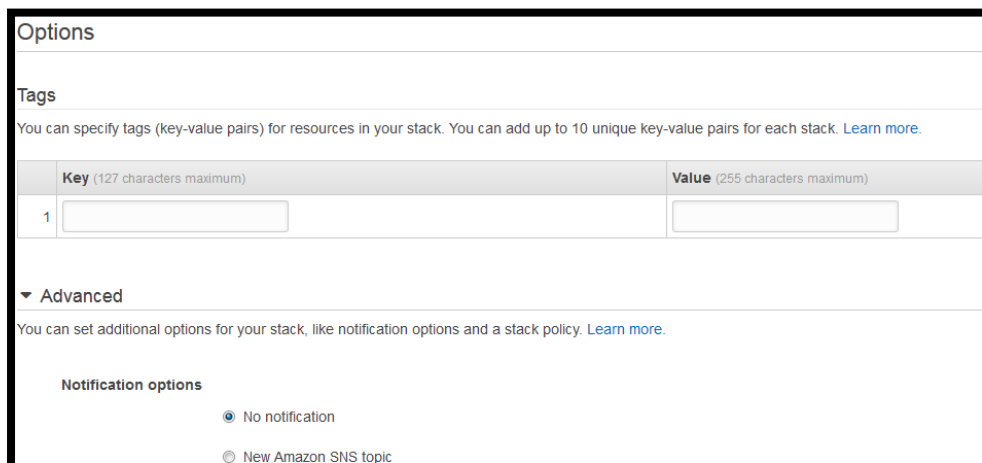
DBPrivateSubnetBCIDR	CIDR block to use for RDS DB Subnet
DBUser	Username to use for RDS DB
DMZSubnetACIDR	CIDR block to use for DMZ/Public Subnet
DMZSubnetBCIDR	CIDR block to use for DMZ/Public Subnet
DatabaseAmi	(optional for future use) leave blank if using RDS DB
DatabaseInstanceType	(optional for future use) leave blank if using RDS DB
DevelopmentCIDR	CIDR block of Development VPC (optional)
DevelopmentSubnetACIDR	CIDR block of Development SubnetA (optional)
DevelopmentSubnetBCIDR	CIDR block of Development SubnetB (optional)
DevelopmentVPCName	Name tag for Development VPC
Keyname	Name of existing SSH key to use in deployment
ManagementCIDR	CIDR block of Management VPC (optional)
ManagementSubnetACIDR	CIDR block of Management Subnet
ManagementSubnetBCIDR	CIDR block of Management Subnet
NatInstanceType	Size for NAT instance
ProductionCIDR	CIDR block for Production Subnet
ProductionVPCName	Production VPC Name
RegionAZ1Name	Availability Zone Name
RegionAZ2Name	Availability Zone Name
RegionDomain	Domain name of region
WebAppPrivateSubnetACIDR	Web App Subnet CIDR
WebAppPrivateSubnetBCIDR	Web App Subnet CIDR
WebInstanceType	Web Instance Type
appAmi	AMI to use for App Instance
createVPCDevelopment	Create Development VPC? (yes or no)
createVPCManagement	Create Management VPC? (yes or no)
pS3CloudTrailBucketExisting	Existing CloudTrail bucket (optional)
pS3CloudTrailLocal	Name of CloudTrail bucket to create (if not using existing)
s3bucketApplicationLogsName	Application Logs Bucket Name to create
s3bucketBackupName	Bucket name for backup logs to create
s3bucketWebContentName	Web/Application Content bucket name to create
stack1URL	URL of stack1.json
stack2URL	URL of stack2.json
stack3URL	URL of stack3.json
stack4URL	URL of stack4.json
webserverAMI	AMI of Web Server AMI to use

b. Click “Next” to Proceed

* S3 buckets have specific naming conventions and must be unique within a region
<http://docs.aws.amazon.com/AmazonS3/latest/dev/BucketRestrictions.html>

4. OPTIONAL: Specify Tags and advanced settings

Tags specified in this section will be applied to all resources created by the stack and can be used to both organize and control access to resources in the stack.



Options

Tags

You can specify tags (key-value pairs) for resources in your stack. You can add up to 10 unique key-value pairs for each stack. [Learn more.](#)

	Key (127 characters maximum)	Value (255 characters maximum)
1	<input type="text"/>	<input type="text"/>

▼ Advanced

You can set additional options for your stack, like notification options and a stack policy. [Learn more.](#)

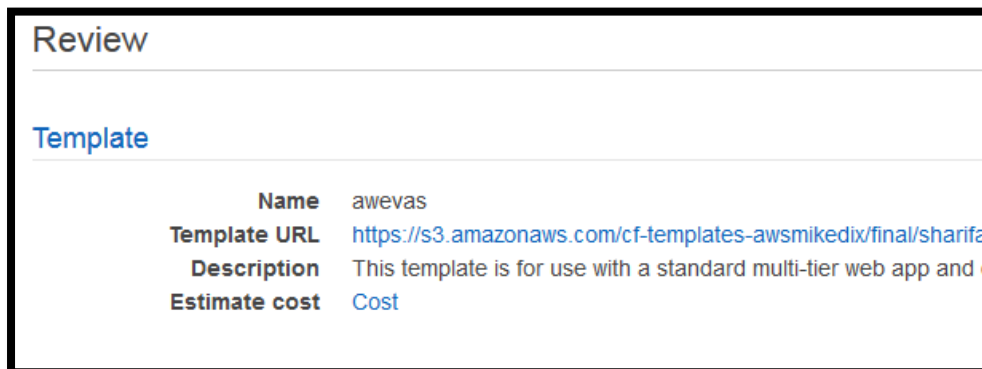
Notification options

☒ No notification

☐ New Amazon SNS topic

These features are optional. Click “Next” to proceed.

5. Review and click “Create” to deploy the template

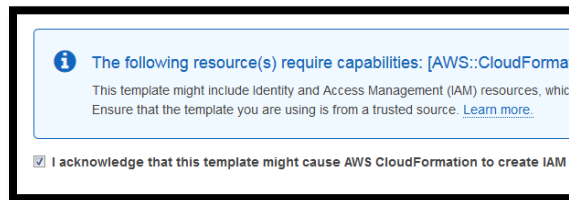


Review

Template

Name	awevas
Template URL	https://s3.amazonaws.com/cf-templates-awsmikedix/final/sharifa...
Description	This template is for use with a standard multi-tier web app and
Estimate cost	Cost

Make sure to select the acknowledgement checkbox. This simply states that the template will create IAM resources.



6. Validate launch successful

Monitor the status of the

stack being deployed:

- If deploying the full template, there will be 5 stacks listed at the end of deployment.
- When successful, the status will change to **CREATE_COMPLETE**

	Created Time	Status	Status Reason	Description
	2014-12-24 12:31:53 UTC-0500	CREATE_IN_PROGRESS	User Initiated	This template is for use with a

Deleting a Stack

Deleting a stack, either via CLI/API or through the CloudFormation console, will remove all resources created by the templates. The only exceptions are S3 buckets for logging and backup because the “DeletionPolicy” is set to “Retain” by default and would have to be deleted manually.

It is important to keep in mind that deleting the “main” stack in effect removes the nested stacks and associated resources.

Integrating with AWS Service Catalog

Creating a New Portfolio & Product

1. As an administrator, go into the [Service Catalog](https://console.aws.amazon.com/catalog) admin console:
<https://console.aws.amazon.com/catalog>

1. “[Create Portfolio](#)” to create a new portfolio

- a. Provide a useful name for the portfolio and description
- b. A portfolio can contain multiple products

2. Upload a new product

- a. Enter product name and description
- b. Enter support details
 - Support email should be distribution list of individuals/group who manage the product
 - Support link can optionally link to internal or external documentation

3. Specify the template to use for the product

- a. Specify the S3 URL of the main.json file
- b. Provide a version description
 - Versioning allows you to have multiple versions of the same product
 - Users can update/change a product version for an existing deployed product

4. On the final page, review your results and confirm the upload of the new product

Adding Users

After adding a product to your portfolio, the portfolio must have users assigned. These are users, groups, or roles in IAM, which will have access to this portfolio when logging into the end user console.

To grant portfolio access to users or groups:

1. In the portfolio details page, expand the Users, groups, and roles section, and then choose Add user, group or role.

2. Choose the Groups, Users, or Roles tab to add groups, users, or roles, respectively.
3. Choose one or more users, groups, or roles, and then choose Add Access to grant them access to the current portfolio.

Adding Constraints

Launch Constraints are a powerful feature of Service Catalog that designate an IAM role for the Service Catalog portfolio products to use when deploying AWS resources. Without a launch constraint, the workload owners or end users need elevated levels of IAM access.

The IAM role used, as a launch constraint for products, must have access to both CloudFormation, as well as any resources the products will deploy. This role is applied to the portfolio and not accessible by end users.

The Service Catalog Admin Guide provides information on configuring launch [constraints](#). Keep in mind that the IAM role used for launch must, along with an IAM policy, have a “Trust Relationship” allowing it to associate with the Service Catalog service. This can easily be added in the console. A simple example of this trust relationship is below:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": "servicecatalog.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Adding Tags

Tags are resource identifiers, which can be used for tracking resource ownership, controlling access, and most importantly allocating cost. Tags are a form of metadata, which consist of Key-Value pairs.

A Service Catalog portfolio can have up to 3 tags applied by administrators, and products.

To add tags to a portfolio:

1. Navigate to the Portfolios page, and then choose the portfolio.
2. On the portfolio details page, expand the Tags section.
3. Type a key and value for the tag.
4. Choose Add Tag.

Tags can be deleted from portfolios at any time, but the change will only apply when end users launch new products. Existing tags applied will remain.

Service Catalog End Users

The Service Catalog end user console allows end-users with access to IAM to launch and deploy the products you have created.

A standard Service Catalog End User IAM Policy can be assigned to users or groups without requiring additional permissions to deploy products.

Below is an example of this IAM policy:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "catalog-user:*",
        "cloudformation:DescribeStackEvents",
        "cloudformation:DescribeStackResource",
        "cloudformation:DescribeStackResources",
        "cloudformation:DescribeStacks",
        "cloudformation:ListStackResources*",
        "cloudformation:ListStacks",
        "cloudformation:GetStackPolicy",
        "cloudformation:GetTemplate",
        "cloudformation:GetTemplateSummary",
        "cloudformation:ValidateTemplate",
        "s3:GetObject",
        "servicecatalog:*"
      ]
    }
  ]
}
```

```
    },  
    "Resource": "*"    
  }  
]  
}
```

End users can be provided with the [Service Catalog End User Guide](#), which provides instructions on launching and managing products from an end user perspective.

Appendix

Special Considerations

Regions and Services

The AWS services used in this template package should exist in all commercial regions, but it is important to be aware of what is available in the region you choose to deploy. For information on US GovCloud service differences review this document:

<http://docs.aws.amazon.com/govcloud-us/latest/UserGuide/supported-services.html>

Instance AMIs

You must specify Amazon Machine Images (AMIs) by ID in the parameters of this template package when deploying if Stack4 is to be used. Existing AMIs available from AWS may be used or you can choose pre-built custom AMIs which your account has access to.

Bootstrapping Configuration Management

It is possible to modify the user data configuration in **stack4.json** to bootstrap the installation of custom applications, or configuration management solutions such as Chef or Puppet. The following sections can be modified as needed.

The “LaunchConfigurationWeb” is the Auto Scaling launch configuration where the Proxy/Web layer is bootstrapped. Presently this template installs and configures an ‘nginx’ proxy on these instances.

```
"LaunchConfigurationWeb": {  
  "Type": "AWS::AutoScaling::LaunchConfiguration",  
  "Condition" : "CreateWebInstance",  
  "DependsOn" : "elbWebApp",  
  "Metadata" : {  
    "AWS::CloudFormation::Init" : {  
      "config" : {  
        "packages" : {  
          "yum" : {  
            "nginx" : [],  
            "java-1.6.0-openjdk-devel":[],  
            "git":[]  
          }  
        },  
        "files" : {  
          "/tmp/nginx/default.conf" : {  
            "content" : { "Fn::Join" : [ "", [  
              "server {\n",  
              "listen 80;\n",  
            ] }  
          }  
        }  
      }  
    }  
  }  
}
```



```
        "listen [::]:80 default ipv6only=on;\n",\n\n        "charset utf-8;\n",\n\n        "location / {\n",\n\n        "proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;\n",\n\n        "proxy_set_header Host $http_host;\n",\n\n        "proxy_redirect off;\n",\n\n        "proxy_pass http://",{"Fn::GetAtt": ["elbWebApp", "DNSName" ]\n},"/;\n","\n","}\n","}\n"]], "mode" : "000755", "owner" : "root", "group" : "root"\n\n    }\n\n},\n\n    "services" : {\n\n        "sysvinit" : {\n\n            "nginx" : { "enabled" : "true", "ensureRunning" : "true", "files":\n["/etc/nginx/conf.d/default.conf"] }\n\n        }\n\n    }\n\n},\n\n},
```

“LaunchConfigurationApp” is the Auto Scaling launch configuration where the application layer is bootstrapped.

```
"LaunchConfigurationApp": {

  "Type": "AWS::AutoScaling::LaunchConfiguration",

  "DependsOn" : "MySQLRDSInstance",

  "Condition" : "CreateAppInstance",

  "Metadata" : {

    "AWS::CloudFormation::Init" : {

    "configSets" : {

      "wordpress_install" : ["install_cfn", "install_wordpress" ]

    },

    "install_cfn" : {

      "files": {

        "/etc/cfn/cfn-hup.conf": {

          "content": { "Fn::Join": [ "", [

            "[main]\n",

            "stack=", { "Ref": "AWS::StackId" }, "\n",

            "region=", { "Ref": "AWS::Region" }, "\n"

          ] ] },

          "mode" : "000400",

          "owner" : "root",
```

```
"group" : "root"

},

"/etc/cfn/hooks.d/cfn-auto-reloader.conf": {

  "content": { "Fn::Join": [ "", [

    "[cfn-auto-reloader-hook]\n",

    "triggers=post.update\n",

    "path=Resources.LaunchConfigurationApp.Metadata.AWS::CloudFormation::Init\n",

    "action=/opt/aws/bin/cfn-init -v ",

    "    --stack ", { "Ref" : "AWS::StackName" },

    "    --resource LaunchConfigurationApp ",

    "    --configsets wordpress_install ",

    "    --region ", { "Ref" : "AWS::Region" }, "\n"

  ] ] },

  "mode" : "000400",

  "owner" : "root",

  "group" : "root"

}

},

"services" : {
```

```
"sysvinit" : {  
  
  "cfn-hup" : { "enabled" : "true", "ensureRunning" : "true",  
  
    "files" : ["/etc/cfn/cfn-hup.conf", "/etc/cfn/hooks.d/cfn-auto-reloader.conf"]}  
  
  }  
  
}  
  
},
```

More information and example user data configurations for using Chef or Puppet with AWS CloudFormation can be found in the following guides:

[Using Chef with AWS CloudFormation](#)

[Integrating AWS CloudFormation with Puppet](#)

Working with the aws-cf-tools Package

The aws-cf-tools package is a set of Python-based tools to facilitate the deployment of the CloudFormation templates in this AWS GoldBase package. It consists of the following tools:

cfdeploy

This tool uses the AWS Boto API to deploy CloudFormation templates. It is an alternative to using the AWS CLI “create-stack” command and allows for added features such as the ability to manage input parameters from a YAML-based configuration file.

cfsecuritycheck

This tool allows customers to build compliance specific validation templates which can be used to verify CloudFormation templates against specific architecture rules. It can be especially useful in a CI/CD deployment.

Readme files in the tools/ directory of this package provide details on using these tools and the proper syntax required.

IAM Policy Examples

The following sample IAM policy can be used for IAM users, groups, and roles to deploy the entire architecture in the AWS GoldBase templates.

```
{
  "Statement": [
    {
      "Resource": "*",
      "Action": "iam:*",
      "Effect": "Allow"
    },
    {
      "Resource": "*",
      "Action": "cloudformation:*",
      "Effect": "Allow"
    },
    {
      "Resource": "*",
      "Action": "cloudtrail:*",
      "Effect": "Allow"
    },
    {
      "Resource": "*",
      "Action": "logs:*",
      "Effect": "Allow"
    },
    {
      "Resource": "*",
      "Action": "sns:*",
      "Effect": "Allow"
    },
    {
      "Resource": "*",
      "Action": "s3:*",
      "Effect": "Allow"
    },
    {
      "Resource": "*",
      "Action": "ec2:*",
      "Effect": "Allow"
    },
    {
      "Resource": "*",
      "Action": "aws-portal:*Billing",
      "Effect": "Deny"
    }
  ]
}
```

```
    },  
    "Version": "2012-10-17"  
}
```

Adding EC2 Instances

Instance-level resources including EC2 instances, RDS instances, Auto Scaling groups, and related configurations should be deployed in stack4.

Instances should be added within the Resources section of the template. The following is a very simple example of how to deploy an EC2 instance in a subnet.

```
"myInstance" : {  
  "Type" : "AWS::EC2::Instance",  
  "Properties" : {  
    "ImageId": "ami-20b65349",  
    "SecurityGroupIds" : [ { "Ref" : "mySecurityGroup" } ],  
    "SubnetId" : { "Ref" : "mySubnet" }  
  }  
}
```

In this example, “Ref” is making a reference to another resource in the CloudFormation template being deployed.

Stack4 Example Instances

The stack4.json included with the AWS GoldBase package already includes a sample architecture, which deploys EC2 instances as part of an Auto Scaling group accessed through an Elastic Load Balancer (ELB). The instance type, image, and detailed configuration information is referenced in the “LaunchConfiguration” for the Auto Scaling group.