Service Catalog

Lab – Using Service Catalog as a Preventive Control

January 2020

Rev 2019-10-16

Table of Contents

Overview

Large enterprises try to find a balance between controlling risk and empowering their developers in alignment with DevOps practices. While developers are required to use AWS services to create optimized architectures for their applications, organizations are concerned that the setup/configuration of infrastructure services might not be done in alignment with the organization's policies.

This solution will demonstrate how AWS Service Catalog can be used to allow development teams to leverage AWS services while enforcing an organization's policies. The solution will create a modest architecture comprised of an autoscaling web application with a load balancer. We will send the application/web/middle tier logs to an Amazon Kinesis Firehose for analysis. The solution will use the following AWS Services; EC2 autoscaling, Application Load Balancer and Kinesis Firehose. The AWS services will be created by using AWS Service Catalog products that have been created for a single AWS service. These AWS Service Catalog products will have security and governance controls built into the products. By creating Service Catalog products for each AWS service; developers can create their own architectures with a self-service experience, and at the same time ensuring that they are complying with organization standards and policies

This solution provides CloudFormation templates to make it easier for developers to automate the provisioning of the Service Catalog products. For more information about this solution including full documentation, AWS Lambda functions code and AWS CloudFormation templates visit: https://github.com/aws-samples/aws-service-catalog-preventive-control.

In this lab we walk you through how to launch a scalable web server built from the individual AWS Service Catalog products chained together and deploy from single CloudFormation template.

The lab is split for two workflows to show the interaction with solution from administrator and developer standpoint.

Prerequisites

You need access to Linux/Mac OS system in order to run shell script in step 2.

In addition, make sure you have latest version of AWS CLI installed and configured on your system. For more information visit:

- AWS CLI Installation/Upgrade Instructions
- AWS CLI Configure Instructions

Finally, you need to install "jq". jq is a lightweight and flexible command-line JSON processor. You can check if "jq" was installed on your machine by running following command:

```
jq --version
```

If "jq" is not present, install it by running one of the following OS dependent commands:

```
Red Hat/Centos: yum install jq
Ubuntu: apt-get install jq
Mac OS: brew install jq
```

Development Environment (Optional)

If you do not have access to Linux/Mac OS, alternatively you can launch an AWS Cloud9 environment.

AWS Cloud9 is a cloud-based integrated development environment (IDE) that lets you write, run, and debug your code with just a browser.

Create Development Environment

Using the AWS CloudFormation template, create the AWS Cloud9 environment.

1. **View the template** – familiarize yourself with the contents of the template by downloading it from here: https://s3.amazonaws.com/aws-service-catalog-reference-architectures/labs/preventive-control/code9-workshop-environment-cfn.yml
2. Log in to the AWS Management Console for the account you plan on deploying the Landing Zone into.
3. **Launch the Stack** – click on the following link to launch the Stack.
4. The template has to be launched in the us-east-1 (N. Virginia) Region
5. On the **Specify Details** page, assign a name for your stack.
6. Under **Parameters**, review the parameters for the template and modify them as necessary. This solution uses the following default values.

Resources Deployment Stack Configuration		
Parameter	**Default**	**Description**
ResourcePrefix	service-catalog-workshop	Prefix for resources created by CloudFormation template
InstanceType	t2.small	Instance type for the Cloud9 environment

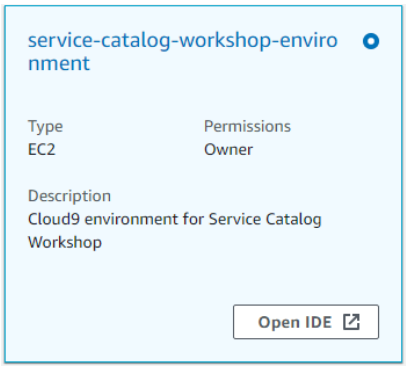
7. Under **Capability** check:
    a. I acknowledge that AWS CloudFormation might create IAM resources with custom names.
8. Choose **Create Stack** to deploy the stack.

9. You can view the status of the stack in the **AWS CloudFormation Console** in the **Status** column. You should see a status of CREATE_COMPLETE in approximately two minutes. Confirm the stack was created successfully before moving to next section.

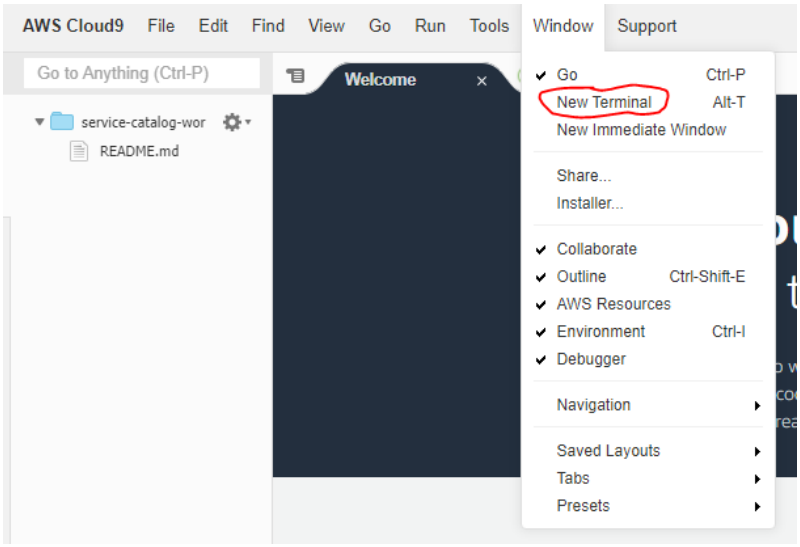
Connect to Cloud9

Navigate to the Cloud9 service within the AWS console.

Click “Open IDE” on the environment created in the previous step



From the “Window” menu select “New Terminal”



In the new terminal window run the following command:

```
sudo yum install jq –y
```

The working environment is ready to execute shell scripts as indicate in workshop. Proceed to the next section.

Administrator Workflow

Create Lab Environment

In this section, we build secure lab environment where we deploy our solution.

Using the AWS CloudFormation template, the following AWS resources will be created:

- A VPC with two private subnets, each in different Availability Zone. In order to prevent direct access to Internet from the VPC, template won't create an Internet Gateway or a NAT Gateway.
- VPC Amazon S3 Endpoint to allow EC2 download from Amazon S3 necessary packages during the bootstrap process.
- VPC Amazon Kinesis Endpoint to allow Kinesis Agent running on the EC2 instance upload logs.
- Amazon S3 bucket to store AWS Service Catalog products deployment templates and configuration files
- A Security Group allow inbound communication to the web server over port 433 from within the VPC
- AWS Lambda functions:
  - ❖ Product Deployment – this function is use to deploy products to AWS Service Catalog
  - ❖ Product Selector – this function is use to return AWS Service Catalog provisioning product id
  - ❖ Resource Selector – this function is use to quickly find resources id by tags
  - ❖ Resource Compliance – this function is use to validate certain security aspects of deploying products

Note: Link to full documentation of each lambda function as well as source code can be find in **Overview** section.

10. **View the template** – familiarize yourself with the contents of the template by downloading it from here: https://s3.amazonaws.com/aws-service-catalog-reference-architectures/labs/preventive-control/deployment-cfn.yml
11. Log in to the AWS Management Console for the account you plan on deploying the Landing Zone into.
12. **Launch the Stack** – click on the following link to launch the Stack.
13. The template has to be launched in the us-east-1 (N. Virginia) Region
14. Click **Next**.
15. On the **Specify Details** page, assign a name to your solution stack.

16. Under **Parameters**, review the parameters for the template and modify them as necessary. This solution uses the following default values.

Resources Deployment Stack Configuration		
Parameter	**Default**	**Description**
PortfolioAccessRole	<Optional Input>	Name of the IAM role that will have admin access to portfolio. Require if you deploying solution under IAM role.
PortfolioAccessUser	<Optional Input>	Name of the IAM use that will have admin access to portfolio. Require if you deploying solution under IAM user.
DeploymentBucketName	*<Requires input>*	Name of S3 bucket where deployment files will be stored
DeploymentConfigSuffix	deployer	Deployment configuration file suffix e.g. deployer
DeploymentLambdaFunctionName	sc-product-deployment	Deployment Lambda function name
DeploymentLambdaRoleName	sc-product-deployment-lambda-role	Deployment Lambda function IAM role name
PolicyName	service-catalog-product-policy	Service Catalog product IAM policy name
PortfolioDescription	Service Catalog Lab Portfolio	Service Catalog portfolio description
PortfolioName	security-products	Service Catalog portfolio name
ProductSelectorLambdaRoleName	sc-product-selector-lambda-role	Product Selector Lambda function IAM role name
ResourceComplianceLambdaRoleName	sc-resource-compliance-lambda-role	Resource Compliance Lambda function IAM role name
ResourceSelectorLambdaRoleName	sc-resource-selector-lambda-role	Resource Selector Lambda function IAM role name
UserName	sc-lab-user	Name of IAM user. This user will have access to provision Service Catalog product as well as launch CloudFormation template
UserPassword	*<Requires input>*	IAM user password. Check AWS account password policy: (https://console.aws.amazon.com/iam/home?region=us-east-1#/account_settings)

17. Under **Capability** check both options:
    b. I acknowledge that AWS CloudFormation might create IAM resources with custom names.
    c. I acknowledge that AWS CloudFormation might require the following capability: CAPABILITY_AUTO_EXPAND
18. Choose **Create Stack** to deploy the stack.
19. You can view the status of the stack in the **AWS CloudFormation Console** in the **Status** column. You should see a status of CREATE_COMPLETE in approximately five minutes. Confirm stack was created successfully before moving to next section.

Deploy Products to Service Catalog

In this section, we will use shell script to create and import self-signed SSL certificate to AWS Certificate Manager as well as deploy products to AWS Service Catalog.

The SSL certificate will be created for domain www.example.com. You can change the domain name by editing shell script. If you decided change domain name, you will need to change it in Step 3 in CloudFormation template under parameter ***DomainName***.

In addition, the deployment script will try to create two service linked roles: *AWSServiceRoleForElasticLoadBalancing* and *AWSServiceRoleForAutoScaling*

Note: The script will output an error if any of these roles already exist,– you can safely ignore this error.

To start deployment, run the following command on the system configured as indicated in prerequisites section:

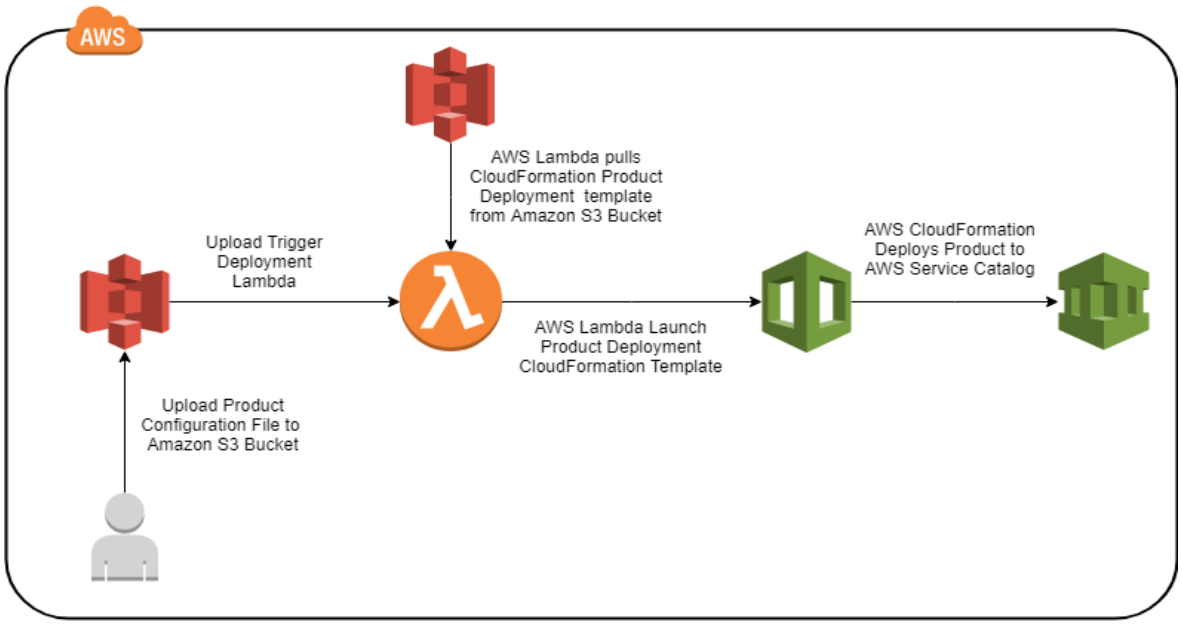
```
wget https://s3.amazonaws.com/aws-service-catalog-reference-architectures/labs/preventive-control/deploy.sh -O deploy.sh

chmod +x deploy.sh

./deploy.sh <cli profile name>
```

CLI Profile Name – is option parameter. If provided all AWS CLI command will be executed under the provided profile. Otherwise the CLI commands will be executed under default profile

Login to AWS Management Console and under AWS CloudFormation check if the product templates were deployed. If not, run the deployment script once again, but this time skip import SSL by calling script with argument “nocert”:

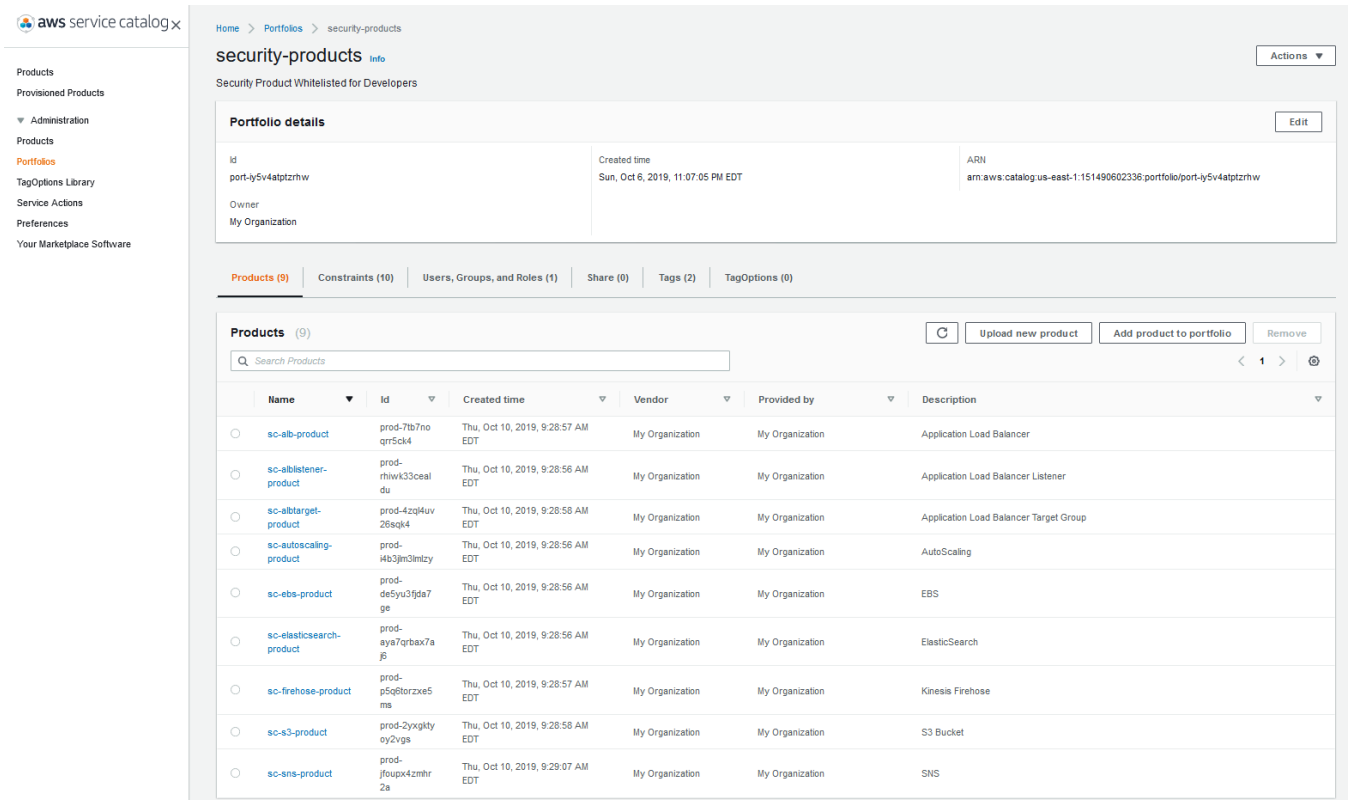
```
./deploy.sh nocert <cli profile name>
```

The following diagram shows the process of deploying products to the AWS Service Catalog.



AWS Service Catalog Product Deployment Diagram

After deployment completed, navigate to the Service Catalog console to validate if products were created successfully. Select portfolio and then select security-products. You should see a listing of created products.



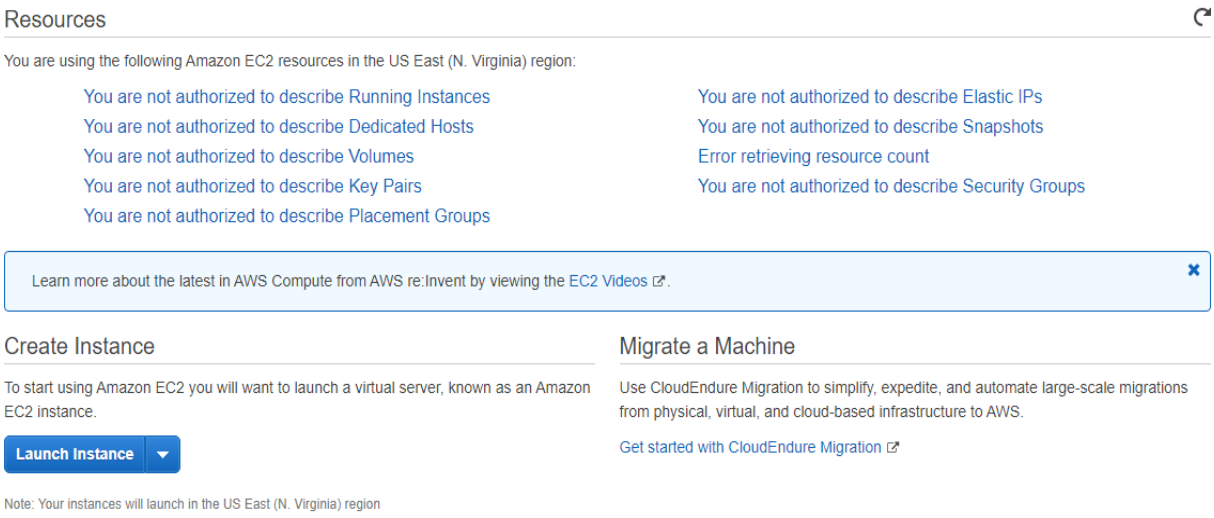
Developer Workflow

Review Developer Permissions

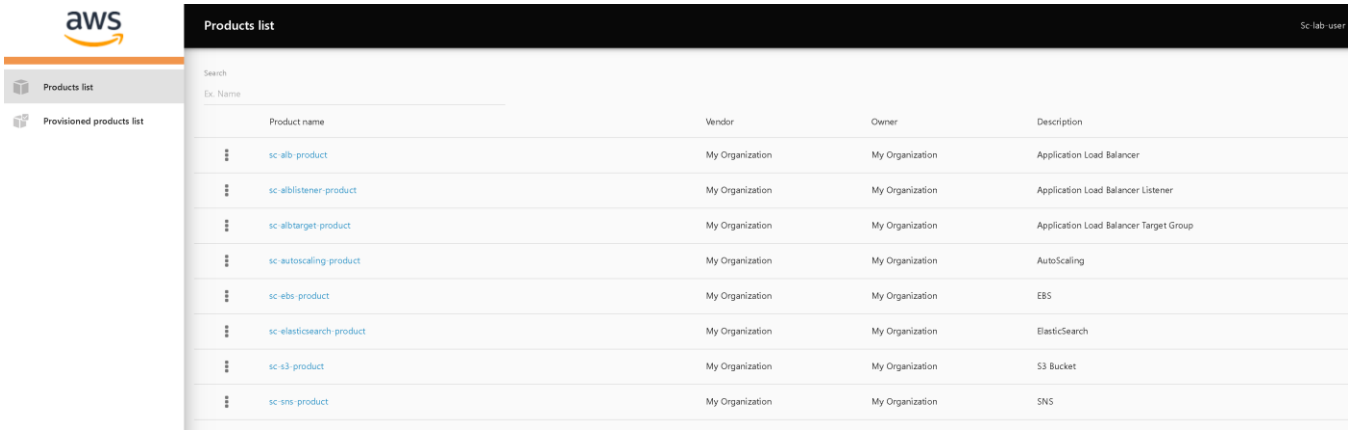
In the previous section as an admin we created an IAM user account named sc-lab-user, deployed Service Catalog products and granted the sc-lab-user user access to Service Catalog portfolio.

The sc-lab-user IAM account is intended to demonstrate a developer persona. To confirm that developers don't have permission to create AWS resource directly but can still provision product from Service Catalog; follow these steps:

1. Login to AWS console as sc-lab-user.
2. Go to EC2 console (you should see several "You are not authorized" errors:



3. Select Load Balancers from the lower half of the left hand navigation menu
   - Click the blue "Create Load Balancer" button.
   - Select the Application Load Balancer
   - At the top you should see an error message (User arn:aws:iam::<account number>sc-lab-user is not authorized …)
4. Navigate to the Service Catalog dashboard (select the Service menu from the top navigation bar on the left
   - Under "Products list" you should see the list of deployed products that developer has permission to provision.



Now we will launch an Apache based application by chaining Service Catalog products exposed by administrator to developers.

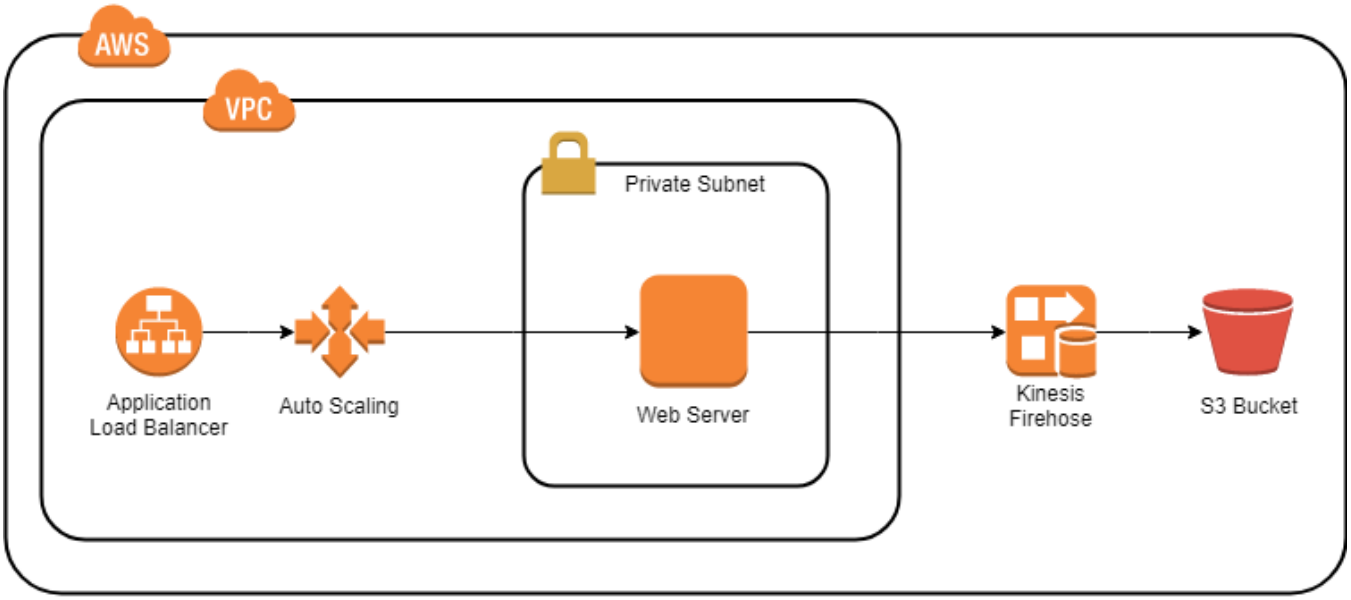
Deploy Web Application

In this section, using an AWS CloudFormation template, we will deploy scalable Apache application using multiple web servers running on the EC2 instance. In addition, we will deploy Amazon Kinesis Firehose to automatically upload Apache logs to Amazon S3 bucket.

Here is the full list of AWS resources that will be created by CloudFormation template:

- EC2 and Kinesis IAM roles
- Application Load Balancer from AWS Service Catalog
- Amazon Kinesis Firehose
- Amazon S3 Bucket to store Apache logs
- KMS Key
- SNS Topic
- AutoScaling

The following diagram shows the architecture of deployed Web Server using AWS Service Catalog products.



Web Application Architecture

With the exception of the KMS Key, all of the resources listed above will be provisioned from the AWS Service Catalog. This will help us satisfy the organization security objectives.

1. **View the template** – familiarize yourself with the contents of the template by downloading it from here: https://s3.amazonaws.com/aws-service-catalog-reference-architectures/labs/preventive-control/web-server-deployment-cfn.yml
2. Log in to the AWS Management Console for the account you plan on deploying the Landing Zone into.
3. **Launch the Stack** – click on the following link to launch the Stack.
4. The template has to launched in the us-east-1 (N. Virginia) Region
5. Click **Next**.
6. On the **Specify Details** page, assign a name to your solution stack.
7. Under **Parameters**, review the parameters for the template and modify them as necessary. This solution uses the following default values.

Web Server Deployment Stack Configuration		
Parameter	**Default**	**Description**
ALBHealthCheckThresholdsList	2,5,5,2	Comma delimited list of ALB health check thresholds: HealthyThreshold, UnhealthyThreshold, IntervalThreshold, TimeoutThreshold
AppHealthCheckGracePeriod	300	Number of seconds after instance launch ALB begins health checks
AppInstanceType	t2.medium	EC2 instance type
AppMinMaxCount	1, 2	Comma delimited list of min then max number of EC2 instances for ASG: min, max
BucketName	*<Requires input>*	S3 Bucket Name to store web server logs. This bucket will be created and cannot already exist.
DomainName	www.example.com	Domain name for which you created SSL certificated in step 2
HealthCheckType	ELB	The service you want the health status from, Amazon EC2 or Elastic Load Balancing Service
ImageId	ami-0de53d8956e8dcf80	AMI ImageId
PortList	443,443	Comma-delimited list if ALB ports: WebALBIn, WebEC2In
TopicName	sc-lab-sns-topic	SNS Notification topic name
KMSAlias	sc-lab-key	The KMS alias for encryption key.

8. Under **Configuration Stack**, If you get “Failed to retrieves sns topics’ or ‘Failed to retrieves IAM roles’, ignore it and click **Next**
9. Under **Capability** check both options:
    a. I acknowledge that AWS CloudFormation might create IAM resources with custom names.
    b. I acknowledge that AWS CloudFormation might require the following capability: CAPABILITY_AUTO_EXPAND
10. Choose **Create Stack** to deploy the stack.
11. You can view the status of the stack in the **AWS CloudFormation Console** in the **Status** column. You should see a status of CREATE_COMPLETE in approximately ten minutes.

You should see traffic on Amazon Kinesis and Apache logs stored on the Amazon S3 bucket after successfully launching the stack.

Cleanup

To delete all resources deployed as a part of this lab, follow these steps:

(Run all steps as admin)

1. Manually delete Amazon S3 bucket created in the section Developer Workflow – Deploy Web Application*
2. Delete AWS Cloud Formation stack created in the section Developer Workflow – Deploy Web Application. The default name of this stack is sc-lab-webserver-stack.
3. Delete ACM certificate and products uploaded to AWS Service Catalog by running this commands:

```
wget https://s3.amazonaws.com/aws-service-catalog-reference-architectures/labs/preventive-control/cleanup.sh -O cleanup.sh

chmod +x cleanup.sh

./cleanup.sh <cli profile name>
```

4. Manually delete Amazon S3 bucket created in the section Administrator Workflow – Create Lab Environment.
5. Delete AWS Cloud Formation stack created in the section Administrator Workflow – Create Lab Environment. The default name for this stack is sc-lab-deployment-stack.
6. If you created Cloud9, delete AWS CloudFormation stack created in the section Development Environment.

* CloudFormation will fail to delete S3 bucket, if bucket contain any objects.

Summary

This exercise showed how to leverage AWS Service Catalog to allow Developers to create their own architecture and enforce the organization security controls in a preventative manner.

