AWS Service Catalog

EC2 Reference Blueprint

AWS Service Catalog EC2 Reference Blueprint

Table of Contents

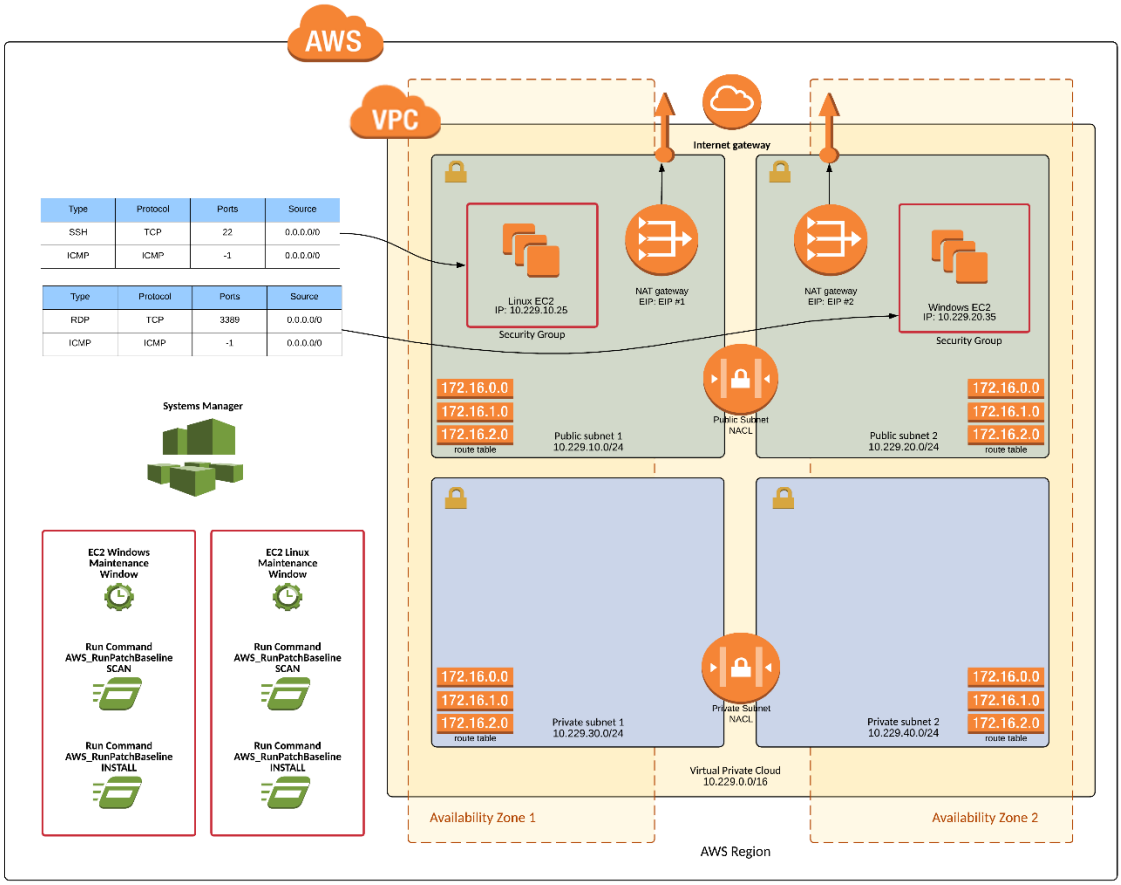
AWS Service Catalog EC2 Reference Blueprint

This reference blueprint demonstrates how an organization can use AWS Service Catalog to provide Amazon Elastic Cloud Compute (Amazon EC2) instances and Amazon Systems Manager instance patching for testing and integration.

After full implementation, this reference blueprint creates an AWS Service Catalog portfolio called “Service Catalog EC2 Reference Blueprint” with two associated products. The AWS Service Catalog product references AWS CloudFormation templates for the Amazon EC2 Linux and Windows instances that can be launched by end users through AWS Service Catalog. The AWS Service Catalog EC2 product creates either an Amazon Linux or Microsoft Windows EC2 instance in the VPC and subnets selected by the end user. A Systems Manager patch baseline, maintenance window, and task are created to allow for automated patching of the Amazon Linux and Microsoft Windows operating systems.

Service Catalog EC2 Reference Blueprint with Amazon Linux and Microsoft Windows instances

To deploy either of the EC2 products described in this document, you do not need to re-create the reference VPC shown in the following diagram. However, if you are interested in deploying a duplicate of the reference VPC shown here, you can first follow the steps outlined in the “Service Catalog VPC Reference Blueprint” documentation included in the ZIP file under the *./VPC directory.



Getting started

Note – Before you distribute the CloudFormation template to your organization, review the template and ensure that it is doing what you want it to do. Check IAM permissions, deletion policies, update stack behavior, and other aspects of the template and ensure that they are as per your expectations. These CloudFormation templates may need updates before you can use them in production.

There are two ways you can deploy this reference blueprint:

1. If you are familiar with the AWS SDK and you are comfortable executing code or scripts (that is, Python) to use AWS, we have included a simple Python script (**sc-ec2-ra-setup.py**) that you can use to deploy the reference blueprint. Follow the steps outlined in the "Script-based Installation Overview" section.

2. If you are not familiar with the AWS SDK, you can still perform the same steps manually, while learning some of the hands-on steps to create AWS Service Catalog portfolios and

products, assign constraints, apply AWS Identity and Access Management (IAM) policies, and so on. For step-by-step instructions, see the “Manual Installation Overview” section.

Regardless of the method, this reference blueprint creates an AWS Service Catalog portfolio with associated EC2 products. The AWS CloudFormation template for the EC2 reference blueprint can be launched by end users through AWS Service Catalog.

Script-based installation overview

The following prerequisites are required for a script-based installation:

- A Python development environment
- A system with permissions to execute the Python script
- Installation and configuration of the AWS CLI
- An AWS SDK to deploy the reference blueprints

Python Development Environment

To deploy this reference blueprint using the provided Python script, you must have Python, Boto, and the AWS CLI installed. We do not provide all necessary documentation on how to get your development environment ready to use the AWS SDK. For more information, see Start Developing with Amazon Web Services and the AWS SDK for Python (Boto3). For information about download options based on your preference of operating systems, see the following Python websites:

- Windows OS: https://www.python.org/downloads/windows/
- macOS: https://www.python.org/downloads/mac-osx/
- Documentation: https://www.python.org/doc/

System with Permissions to Execute the Python Script

For automatic installation, the provided script uses modules for “boto3” and “random”. For information about installing and configuring the boto3 python module, see the Boto 3 Quick Start documentation.

AWS Command Line Interface

Install and configure the AWS Command Line Interface (AWS CLI). It is important to ensure that the AWS CLI configuration contains the correct target AWS Region, as this Region is used to create the reference blueprint components within AWS Service Catalog.

For more information, see AWS Command Line Interface.

AWS SDK to deploy this reference blueprints

1. Download the reference blueprint zip file and expand its content into a folder.
    a. The location for the reference blueprint is: https://github.com/aws-samples/aws-service-catalog-reference-blueprints
2. Contents include:
    - ./README.pdf (this file)
    - ./COPYING.pdf
    - ./LICENSE.pdf
    - ./NOTICE.pdf
    - ./sc-ec2-ra-setup.py (Python script used during setup process)
    - ./sc-ec2-linux-ra.yml (EC2 CloudFormation template in YAML)
    - ./sc-ec2-linux-ra.json (EC2 CloudFormation template in JSON)
    - ./sc-ec2-windows-ra.yml (EC2 CloudFormation template in YAML)
    - ./sc-ec2-windows-ra.json (EC2 CloudFormation template in JSON)
    - ./sc-ec2-ra-blueprint.png (image of reference blueprint)
3. Provide execute permissions to the Python script.
4. Confirm the AWS Region for deployment.
5. Execute the Python setup script.

AWS Service Catalog portfolio access

After the setup script completes, there is a new AWS Service Catalog portfolio with new EC2 products associated in the specified Region. Before launching these products, grant access to the portfolio for the AWS Service Catalog admin and end users. To grant access to the portfolio, follow the steps outlined in the "Manual Installation Overview" section of this document.

Manual installation overview

Before you get started with AWS Service Catalog, you need to be familiar with its components. You also need to know about the initial workflows for administrators and end users, which are the two primary user types in AWS Service Catalog. You grant permissions to these users such that they can access the required functionality of AWS Service Catalog.

AWS Service Catalog supports the following types of users:

- **Catalog administrators (administrators)** – Manage a catalog of products (applications and services), organizing them into portfolios and granting access to end users. Catalog administrators prepare AWS CloudFormation[1] templates, configure constraints, and manage IAM roles that are assigned to products to provide for advanced resource management.

[1] https://aws.amazon.com/documentation/cloudformation/

- **End users** – Receive AWS credentials from their IT department or manager and use the AWS Management Console to launch products to which they have been granted access. Sometimes referred to as simply *users*, end users may be granted different permissions depending on your operational requirements. For example, a user may have the maximum permission level (to launch and manage all of the resources required by the products they use) or only permission to use particular service features.

Grant permissions to administrators and end users

Catalog administrators and end users require different IAM permissions to use AWS Service Catalog. As a catalog administrator, you must have IAM permissions that allow you to access the AWS Service Catalog administrator console and create and manage products.

Before your end users can use your products, you must grant them permissions that allow them to access the AWS Service Catalog end user console, launch products, and manage launched products as provisioned products.

AWS Service Catalog provides many of these permissions using managed policies. AWS maintains these policies and provides them in the AWS Identity and Access Management (IAM) service. You can use these policies by attaching them to the IAM users, groups, or roles that you and your end users use.

Create an AWS Service Catalog administrator and grant permissions

As a catalog administrator, you require access to the AWS Service Catalog administrator console view and IAM permissions that allow you to perform tasks such as the following:

- Creating and managing portfolios
- Creating and managing products
- Adding template constraints to control the options that are available to end users when launching a product
- Adding launch constraints to define the IAM roles that AWS Service Catalog assumes when end users launch products
- Granting end users access to your products

You, or an administrator who manages your IAM permissions, must attach policies to your IAM user, group, or role that are required to successfully deploy this product/solution.

1. Open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane, choose **Users**. If you have already created an IAM user to use as the catalog administrator, select the user name and choose **Add permissions**. Otherwise, create a user as follows:

a. Choose **Add user**.
b. For **User name**, type **ServiceCatalogAdmin**.
c. Select **Programmatic access** and **AWS Management Console access**.
d. Choose **Next: Permissions**.

3. Choose **Attach existing policies directly**.
4. Choose **Create policy** and enter the following values:

a. For **Create Your Own Policy**, choose **Select**.
b. For **Policy Name**, type **ServiceCatalogAdmin-AdditionalPermissions**.
c. For **Policy Document**, copy the following example policy and paste it in:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateKeyPair",
                "iam:AddRoleToInstanceProfile",
                "iam:AddUserToGroup",
                "iam:AttachGroupPolicy",
                "iam:CreateAccessKey",
                "iam:CreateGroup",
                "iam:CreateInstanceProfile",
                "iam:CreateLoginProfile",
                "iam:CreateRole",
                "iam:CreateUser",
                "iam:Get*",
                "iam:List*",
                "iam:PutRolePolicy",
                "iam:UpdateAssumeRolePolicy"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

d. (Optional) You must grant administrators additional permissions for Amazon S3 if they need to use a private CloudFormation template. For more information, see User Policy Examples[2] in the *Amazon Simple Storage Service Developer Guide*.
e. Choose **Create policy**.

5. Return to the browser window with the permissions page and choose **Refresh**.
6. In the search field, type **ServiceCatalog** to filter the policy list.
7. Select the check boxes for the **AWSServiceCatalogAdminFullAccess** and ServiceCatalogAdmin-**AdditionalPermissions** policies, and then choose **Next: Review**.
8. If you are updating a user, choose **Add permissions**.

[2] https://docs.aws.amazon.com/AmazonS3/latest/dev/example-policies-s3.html

9. If you are creating a user, choose **Create user**. You can download or copy the credentials and then choose **Close**.
10. To sign in as the catalog administrator, use your account-specific URL. To find this URL, choose **Dashboard** in the navigation pane and choose **Copy Link**. Paste the link in your browser, and use the name and password of the IAM user that you created or updated in this procedure.

Create an AWS Service Catalog end user and grant permissions

1. Open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane, choose **Users**. If you have already created an IAM user to use as the catalog end user, select the user name and choose **Add permissions**. Skip to step 3. Otherwise, create a user as follows:

    a. Choose **Add user**.
    b. For **User name**, type **ServiceCatalogUser**.
    c. Select **Programmatic access** and **AWS Management Console access**.
    d. Choose **Next: Permissions**.

3. In the navigation pane, choose **Attach existing policies directly**.
4. In the **Policy type** search box, type **AWSServiceCatalog** and select the **AWSServiceCatalogEndUserFullAccess** policy.
5. Choose **Next:Review**.
6. Choose **Create user**.

Required files

To provision and configure portfolios and products, you use AWS CloudFormation templates, which are JSON— or YAML—formatted text files. For more information, see Template Formats in the *AWS CloudFormation User Guide*. These templates describe the resources that you want to provision. You can use the CloudFormation editor or any text editor to create and save templates. For this reference blueprint, we've provided templates to get you started:

- Linux EC2 Reference Blueprint
    - **`sc-ec2-linux-ra.json`**
    - **`sc-ec2-linux-ra.yml`**
- Windows EC2 Reference Blueprint
    - **`sc-ec2-windows-ra.json`**
    - **`sc-ec2-windows-ra.`**

You only need either the JSON or YML files, not both.

Create an AWS Service Catalog portfolio

To provide users with products, begin by creating a portfolio for those products.

1. Open the AWS Service Catalog console at https://console.aws.amazon.com/servicecatalog/.
2. If you are using the AWS Service Catalog administrator console for the first time, choose **Get started**. Otherwise, choose **Create portfolio**.
3. Enter the following values:
    - **Portfolio name**: Service Catalog EC2 Reference Blueprint
    - **Description:** Service Catalog Portfolio that contains reference blueprint products for Amazon Elastic Compute Cloud.
    - **Owner:** IT Services
4. Choose **Create**.

Create EC2 products in AWS Service Catalog

1. In the AWS Service Catalog console, choose Service Catalog drop-down at the top-left. Under **Admin**, choose **Portfolios list**.
2. Choose the portfolio that you created in the previous step (Service Catalog EC2 Reference Blueprint).
3. Choose **Upload New Product**.
4. Enter the following product information and choose **Next**:
    a. **Product Name:** Amazon Elastic Compute Cloud (EC2) Windows
    b. **Description:** This product builds one Microsoft Windows EC2 instance and creates an SSM path baseline, maintenance windows, and patch task to scan for and install operating system updates on the EC2 instance.
    c. **Provided by:** IT Services
    d. **Vendor:** <blank>
5. Enter the following support information, and choose **Next**:
    - **Email contact:** it@yourcompany.com
    - **Support link:** http://helpdesk.yourcompany.com
    - **Support description:** Operations Team
6. Choose **Upload a template file**, **Browse**.
7. Navigate to the *sc-ec2-windows-ra.json* template on your system, and choose **Open**.
8. Fill in the remaining values on this page, and choose **Next**.
9. Verify the data on the review page, and choose **Create**.

Share the portfolio and variable VPC product with end users or accounts

1. In the AWS Service Catalog console, choose **Admin**, **Portfolios List**.

2. Open the portfolio to share (Service Catalog VPC Reference Blueprint).
3. Expand **Users, groups and roles** and cChoose **Add User, Group, or Role**.
4. On the respective tabs, select each user, group, or role to which to give portfolio access. In this example, share with the **`ServiceCatalogEndUser`** user. The same procedure can be followed to share with a Group or with Roles.
5. Choose **Add Access**.

Create a key pair

To enable your end users to launch the product that is based on the sample templates for this reference blueprint, you must create two Amazon EC2 key pairs, one for each EC2 product (that is, Linux and Windows).

A key pair is a combination of a public key used to encrypt data and a private key used to decrypt data. For more information, see Amazon EC2 Key Pairs in the *Amazon EC2 User Guide*.

The AWS CloudFormation templates for this reference blueprint, **sc-ec2-linux-ra.json** and **sc-ec2-windows-ra.json**, include the KeyName parameter:

```
. . .
"KeyName": {
      "Ref": "KeyPair"
                },
. . .
```

Specify the name of a key pair when you use AWS Service Catalog to launch the product that is based on the templates, for both Linux and Windows.

If you already have key pairs in your account that you would prefer to use, you can skip this section. Otherwise, complete the following steps.

To create a key pair

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the navigation pane, under **Network & Security**, choose **Key Pairs**.
3. On the Key Pairs page, choose **Create Key Pair**.
4. For **Key pair name**, type **SC-EC2-RA-Linux-Instance** for the name, and choose **Create**.
5. When the console prompts you to save the private key file, save it in a safe place.
6. Repeat steps 1–4 but use **SC-EC2-RA-Windows-Instance** for the name in step 4, and then choose **Create**.
7. When the console prompts you to save the private key file, save it in a safe place.

Important: This is the only chance for you to save the private key file.

Add a launch constraint to assign an IAM role

A launch constraint designates an IAM role that AWS Service Catalog assumes when an end user launches a product.

For this step, you add a launch constraint to the Linux and Windows Desktop products so that AWS Service Catalog can use the AWS resources that are part of the product's AWS CloudFormation template. This launch constraint enables the end user to launch the product and manage it as a provisioned product post-launch. For more information, see AWS Service Catalog Launch Constraints.

Without a launch constraint, you need to grant additional IAM permissions to your end users before they could use either the EC2 Linux or Windows instances. For example, the **AWSServiceCatalogEndUserFullAccess** policy grants the minimum IAM permissions required to access the AWS Service Catalog end user console view. By using a launch constraint, you can keep your end users' IAM permissions to a minimum, which is an IAM best practice. For more information, see Grant least privilege in the *IAM User Guide*.

To create the IAM role

1. Open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane, choose **Policies**, **Create policy**.
3. Enter the following values:

    - For **Create Your Own Policy**, choose **Select**.
    - For **Policy Name**, type **LinuxWindowsEC2-ra-Policy**.
    - For **Policy Document**[3], copy the following example policy and paste it in:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": "iam:AttachRolePolicy",
            "Resource": "*",
            "Condition": {
```

[3] This policy has been created exclusively for demonstration purposes. As shown here, this policy demonstrates how to allow and/or restrict access to different AWS Services (i.e. IAM, S3, SSM, CloudFormation, EC2, etc.) resources. In its current form, the policy should never be used for production environments without consulting your security or compliance team. Keep in mind that an IAM policy should only grant the least amount of privileges required to perform a task and nothing more. For more information, see IAM Best Practices (IAM Best Practices).

```
                "ArnEquals": {
                    "iam:PolicyARN": [
                        "arn:aws:iam::aws:policy/service-
role/AmazonEC2RoleforSSM",
                        "arn:aws:iam::aws:policy/AmazonSSMFullAccess",
                        "arn:aws:iam::aws:policy/service-
role/AmazonSSMMaintenanceWindowRole"
                    ]
                }
            }
        },
        {
            "Sid": "VisualEditor1",
            "Effect": "Allow",
            "Action": [
                "ec2:StartInstances",
                "ec2:RunInstances"
            ],
            "Resource": "*",
            "Condition": {
                "StringEqualsIfExists": {
                    "aws:RequestTag/Description": "Service-Catalog-EC2-
Reference-Architecture"
                }
            }
        },
        {
            "Sid": "VisualEditor2",
            "Effect": "Allow",
            "Action": [
                "ec2:AuthorizeSecurityGroupIngress",
                "ec2:ModifyVolumeAttribute",
                "ec2:DescribeInstances",
                "ec2:ResetInstanceAttribute",
                "ec2:ReportInstanceStatus",
                "iam:AddRoleToInstanceProfile",
                "ec2:UpdateSecurityGroupRuleDescriptionsIngress",
                "ssm:DescribeMaintenanceWindowExecutionTaskInvocations",
                "ssm:RegisterPatchBaselineForPatchGroup",
                "ec2:DescribeVolumeStatus",
                "cloudformation:DescribeStackEvents",
                "ec2:RevokeSecurityGroupEgress",
                "ec2:DescribeVolumes",
                "cloudformation:UpdateStack",
                "ec2:DescribeKeyPairs",
                "iam:ListRolePolicies",
                "ssm:UpdateManagedInstanceRole",
                "ssm:UpdatePatchBaseline",
                "iam:ListPolicies",
                "iam:GetRole",
                "ssm:GetMaintenanceWindowExecutionTask",
                "ssm:RegisterTaskWithMaintenanceWindow",
                "ec2:CreateTags",
                "ssm:GetMaintenanceWindowExecution",
                "iam:DeleteRole",
                "ec2:StopInstances",
                "ec2:DescribeVolumeAttribute",
```



```
                "ec2:CreateVolume",
                "ec2:RevokeSecurityGroupIngress",
                "ec2:CreateNetworkInterface",
                "ssm:UpdateMaintenanceWindow",
                "ec2:DisassociateIamInstanceProfile",
                "ssm:DescribeMaintenanceWindows",
                "ssm:DescribeEffectivePatchesForPatchBaseline",
                "cloudformation:DeleteStack",
                "ec2:DescribeSubnets",
                "cloudformation:ValidateTemplate",
                "ec2:AttachVolume",
                "iam:CreateInstanceProfile",
                "ssm:GetDefaultPatchBaseline",
                "ec2:DescribeRegions",
                "ssm:DescribePatchGroupState",
                "ssm:UpdateMaintenanceWindowTask",
                "ssm:CreatePatchBaseline",
                "ssm:DescribeMaintenanceWindowExecutionTasks",
                "iam:PassRole",
                "ssm:DescribeInstancePatchStatesForPatchGroup",
                "ssm:CreateMaintenanceWindow",
                "ssm:GetMaintenanceWindow",
                "ec2:DescribeInstanceStatus",
                "ssm:GetPatchBaseline",
                "ec2:RebootInstances",
                "cloudformation:SetStackPolicy",
                "iam:DeleteInstanceProfile",
                "iam:ListRoles",
                "ssm:DescribePatchBaselines",
                "ssm:DeregisterPatchBaselineForPatchGroup",
                "ec2:DescribeSecurityGroups",
                "ssm:DescribeInstanceInformation",
                "ec2:DescribeVpcs",
                "ssm:RegisterDefaultPatchBaseline",
                "ssm:DeregisterTargetFromMaintenanceWindow",
                "iam:RemoveRoleFromInstanceProfile",
                "ssm:DescribeInstancePatches",
                "iam:CreateRole",
                "ssm:UpdateMaintenanceWindowTarget",
                "ec2:AssociateVpcCidrBlock",
                "ssm:GetMaintenanceWindowExecutionTaskInvocation",
                "ec2:AssociateRouteTable",
                "ec2:DeleteVolume",
                "iam:DetachRolePolicy",
                "ec2:DescribeRouteTables",
                "ec2:DetachVolume",
                "ec2:ModifyVolume",
                "ec2:UpdateSecurityGroupRuleDescriptionsEgress",
                "ssm:RegisterTargetWithMaintenanceWindow",
                "cloudformation:DescribeStacks",
                "s3:GetObject",
                "ec2:AssociateSubnetCidrBlock",
                "ec2:DeleteTags",
                "ssm:DescribeMaintenanceWindowTasks",
                "ssm:GetPatchBaselineForPatchGroup",
                "ssm:DeletePatchBaseline",
                "ssm:DescribeMaintenanceWindowExecutions",
```



```
                "ec2:DescribeNetworkInterfaces",
                "ssm:DescribeInstancePatchStates",
                "ec2:CreateSecurityGroup",
                "ssm:DeregisterTaskFromMaintenanceWindow",
                "ec2:ModifyInstanceAttribute",
                "ssm:DescribeInstanceAssociationsStatus",
                "ssm:DescribeInstanceProperties",
                "ec2:AuthorizeSecurityGroupEgress",
                "ec2:TerminateInstances",
                "ec2:DescribeIamInstanceProfileAssociations",
                "ec2:DescribeTags",
                "ssm:DescribeMaintenanceWindowTargets",
                "ssm:DeleteMaintenanceWindow",
                "ssm:DeregisterManagedInstance",
                "cloudformation:GetTemplateSummary",
                "ec2:DescribeSecurityGroupReferences",
                "cloudformation:CreateStack",
                "ec2:DeleteSecurityGroup",
                "ec2:AssociateIamInstanceProfile",
                "ssm:DescribeAvailablePatches"
            ],
            "Resource": "*"
        },
        {
            "Sid": "VisualEditor3",
            "Effect": "Allow",
            "Action": "catalog-user:*",
            "Resource": "*"
        }
    ]
}
```

4. Choose **Create Policy**.
5. In the navigation pane, choose **Roles**, **Create role**.
6. For **Select role type**, choose **AWS service**, **Service Catalog**.
7. For **Select your use case heading,** choose **Service Catalog.** Choose **Next: Permissions**.
8. In the **Policy type** search box, type **LinuxWindows** and select the check box for the **LinuxWindowsEC2-ra-Policy** policy. Choose **Next: Review**.
9. For **Role name**, type **LinuxWindowsEC2-RA-LaunchRole**.
10. Choose **Create role**.

To add the launch constraint

1. Open the AWS Service Catalog console at https://console.aws.amazon.com/servicecatalog/.
2. Choose the Service Catalog EC2 Reference Blueprint portfolio.
3. On the portfolio details page, expand the **Constraints** section, and choose **Add constraints**.
4. For **Product**, choose **Amazon Elastic Compute Cloud (EC2) Windows**, and for **Constraint type**, choose **Launch**. Choose **Continue**.

5. On the Launch constraint page, for **IAM role**, choose **LinuxWindowsEC2-RA-LaunchRole**, and then choose **Submit**.
6. Repeats steps 5–8. In step 7, choose the **Amazon Elastic Compute Cloud (EC2) Linux** instead.

AWS Service Catalog product launch

After access has been provided to one or more end users, the EC2 reference blueprint products can be launched.

To launch an EC2 reference blueprint product, the user needs to log into AWS Service Catalog, select the EC2 Reference Blueprint Linux or Windows product and choose **Launch**. The launch process asks the end user for various details about how the EC2 product is to be configured. After the form fields are filled out and the product is launched, AWS Service Catalog executes a CloudFormation stack to build the product and provide the EC2 details to the end user.

Service Catalog EC2 Reference Blueprint cleanup

To remove the Service Catalog EC2 Reference Blueprint from AWS Service Catalog, perform the following steps:

1. Terminate all Service Catalog EC2 Reference Blueprint provisioned products.
2. Remove all products from the portfolio.
3. Remove all constraints from the portfolio.
4. Remove all access to users, groups, and roles from the portfolio.
5. Remove all shares associated with the portfolio.
6. Remove all tags from the portfolio.
7. Remove all tagOptions from the portfolio.
8. Delete all products from AWS Service Catalog.
9. Delete the portfolio from AWS Service Catalog.

Disclaimer

AWS has made efforts to create safe and repeatable blueprints for users to use. However, as expressed in the AWS Service Terms, Amazon.com specifically disclaims all warranties in respect of the artifacts provided in the SCIB program. Furthermore, user acknowledges that SCIB content may in fact have bugs and/or may malfunction. User expressly acknowledges that there is inherent risks associated with using or testing non-production or sample products or services.

Note – Before you distribute the CloudFormation template to your organization, review the template and ensure that it is doing what you want it to do. Check IAM permissions, deletion policies, update stack behavior, and other aspects of the template and ensure that they are as per your expectations. These CloudFormation templates may need updates before you can use them in production.

License

Authors

Israel Lawson – AWS Sr. Solutions Architect

Kanchan Waikar – AWS Solutions Architect

Abel Cruz – AWS Sr. Business Development Manager

Acknowledgments

The following AWS team members have provided guidance, code reviews, and other assistance throughout the design of this reference blueprint.

David Aiken – AWS Solutions Architect Manager

Mahdi Sajjadpour – AWS Principal Business Development Manager

Phil Chen – AWS Sr. Solutions Architect

Kenneth Walsh – AWS Solutions Architect

