Automating Governance

A Managed Service Approach to Security and Compliance on AWS

August 2015

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

Abstract

This whitepaper is intended for existing and potential Amazon Web Services (AWS) customers who are implementing security controls for applications running on AWS. It provides guidelines for developing and implementing a managed service approach to deploying applications in AWS. The guidelines described provide enterprise customers with greater control over their applications while accelerating the process of deploying, authorizing, and monitoring these applications.

This paper is targeted at IT decision makers and security personnel and assumes familiarity with basic networking, operating system, data encryption, and operational control security practices.

Introduction

Governance encompasses an organization's mission, long-term goals, responsibilities, and decision making. Gartner describes governance as "the processes that ensure the effective and efficient use of IT in enabling an organization to achieve its goals"[1]. An effective governance strategy defines both the frameworks for achieving goals and the decision makers who create them:

- Frameworks – The policies, principles, and guidelines that drive consistent IT decision making.
- Decision makers – The entities or individuals who are responsible and accountable for IT decisions.

Well-developed frameworks ultimately can yield an efficient, secure, and compliant technology environment. This paper describes how to develop and automate these frameworks by introducing the following concepts and practices:

- A managed service organization (MSO) that is part of a centralized cloud governance model
- Roles and responsibilities of the MSO on the customer side of the AWS shared responsibility model

- Shared services and the use of Amazon Virtual Private Cloud (Amazon VPC) within AWS
- Architectural baselines for establishing minimum configuration requirements for applications being deployed in AWS
- Automation methods that can facilitate application deployment and simplify compliance accreditation

Shared Responsibility Environment

Moving IT infrastructure to services in AWS creates a model of shared responsibility between the customer and AWS. This shared model helps relieve the operational burden on the customer because AWS operates, manages, and controls the IT components from the host operating system and virtualization layer down to the physical security of the facilities in which the services operate.

The customer assumes responsibility for and management of the guest operating system (including responsibility for updates and security patches) and other associated application software, and the configuration of the AWS-provided security group firewall. Customers must carefully consider the services they choose because their responsibilities vary depending on the services they use, the integration of those services into their IT environment, and applicable laws and regulations.

Figure 1: The AWS Shared Responsibility Model

This customer/AWS shared responsibility model also extends to IT controls. Just as AWS and its customers share the responsibility for operating the IT environment, they also share the management, operation, and verification of IT controls. AWS can help relieve the customer of the burden of operating controls by managing those controls associated with the physical infrastructure deployed in the AWS environment that might previously have been managed by the customer. Customers can shift the management of certain IT controls to AWS, which results in a (new) distributed control environment. Customers can then use the AWS control and compliance documentation to perform their control evaluation and verification procedures as required under the applicable compliance standard.

Compliance Requirements

The infrastructure and services provided by AWS are approved to operate under several compliance standards and industry certifications. These certifications cover only the AWS side of the shared responsibility model; customers retain the responsibility for certifying and accrediting workloads that are deployed on top of the AWS-provided services that they run.

The following common compliance standards have unique requirements that customers must consider:

- **NIST SP 800-53**[2]–Published by the National Institute of Standards in Technology (NIST), NIST SP 800-53 is a catalog of security controls which most U.S. federal agencies must comply with and which are widely used within private sector enterprises. Provides a risk management framework that adheres to the Federal Information Processing Standard (FIPS).
- **FedRAMP**[3]–A U.S. government program for ensuring standards in security assessment, authorization, and continuous monitoring. FedRAMP follows the NIST 800-53 security control standards.
- **DoD Cloud Security Model (CSM)**[4]–Standards for cloud computing issued by the U.S. Defense Information Systems Agency (DISA) and documented in the Department of Defense (DoD) Security Requirements Guide (SRG). Provides an authorization process for DoD workload owners who have unique architectural requirements depending on impact level.

- **HIPAA**[5] – The Health Insurance Portability and Accountability Act (HIPAA) contains strict security and compliance standards for organizations processing or storing Protected Health Information (PHI).
- **ISO 27001**[6] – ISO 27001 is a widely adopted global security standard that outlines the requirements for information security management systems. It provides a systematic approach to managing company and customer information that's based on periodic risk assessments.
- **PCI DSS**[7] – Payment Card Industry (PCI) Data Security Standards (DSS) are strict security standards for preventing fraud and protecting cardholder data for merchants that process credit card payments.

Evaluating systems in the cloud can be a challenge unless there are architectural standards that align with compliance requirements. These architectural standards are especially critical for customers who must prove their systems meet strict compliance standards before they are permitted to go into production.

Compliance and Governance

AWS customers are required to continue to maintain adequate governance over the entire IT control environment regardless of whether it is deployed in a traditional data center or in the cloud. Leading governance practices include:

- Understanding required compliance objectives and requirements (from relevant sources)
- Establishing a control environment that meets those objectives and requirements
- Understanding the validation required, based on the organization's risk tolerance
- Verifying the operational effectiveness of the control environment

Deployment in the AWS cloud gives organizations options to apply various types of controls and verification methods.

Workload owners can follow these basic steps to ensure strong governance and compliance:

1. Review information from AWS and other sources to understand the entire IT environment.

2. Document all compliance requirements.
3. Design and implement control objectives to meet the organization's compliance requirements.
4. Identify and document controls owned by outside parties.
5. Verify that all control objectives are met and all key controls are designed and operating effectively.

Approaching compliance governance in this manner will help customers gain a better understanding of their control environment and help clearly define the verification activities that must be performed.

For more information on governance in the cloud, see Security at Scale: Governance in AWS[8].

Challenges in Architecting for Governance

AWS provides a high level of flexibility in how customers can design architectures for their applications in the cloud. AWS has documented best practices in the whitepapers, user guides, API references, and other resources that describe how to design for elasticity, availability, and security. But these resources alone do not prevent bad design and improper configuration. Architectural decisions that impact security can put customer data or personal information at risk and create liability.

Consider the following challenges:

- Building a single workload with different architecture choices that is still compliant
- The need to individually assess each of these unique architectures
- The high level of flexibility leaves room for error, and serious mistakes can be resolved only by redeployment of the application
- Security analysts may not understand the differences between the many architectural decisions

Learning Curve

By deploying applications in AWS, workload owners and developers have a much greater level of control over and access to resources beyond the operating system and software. However, the number of decisions required when building an architecture can be overwhelming for those new to AWS. Some of these architectural decisions include how to address:

- Amazon VPC structure and network controls
- AWS Identity and Access Management (IAM) configuration, policies, permissions; Amazon Simple Storage Service (S3) bucket policies
- Storage and database options
- Load balancing
- Monitoring options, alerts, tagging
- Aggregation, analysis, and storage considerations for logging produced by a workload or AWS service

Implementing a Managed Services Organization

To implement governance, AWS customers have begun establishing centralized teams within their organizations that facilitate the migration of legacy applications and the development of new applications. Such a team can be called a provisioning team, a center of excellence, a broker, and, most commonly, the managed service organization (MSO), which is the term we use. Customers use an MSO to establish repeatable processes and templates for deploying applications to AWS while maintaining organizational control over their enterprise's applications. When the MSO function is outsourced, it is generally referred to as a managed service partner (MSP). Many MSPs are validated by AWS under our Managed Service Program.[9]

Understanding the enterprise's cloud governance model is key to determining the provisioning strategy for accounts, Amazon VPCs, and applications, and for deciding how to automate these processes. Large enterprises generally centrally manage cloud operations at some level. It is important to find the optimal balance between central management and decentralized control.[10]

In a centralized governance model, an MSO provides the minimum requirements for workload owners who are deploying applications in the cloud:

- Guardrails for security, data protection, and disaster recovery
- Shared services for security, continuous monitoring, connectivity, and authentication
- Auditing the deployments of workload owners to ensure adherence to security and compliance standards

For most large enterprises, there are typically two sets of cloud governance roles involved in the deployment of applications:

- **MSO** – As previously mentioned, a component of centralized cloud governance; responsibilities can include account provisioning, establishment of connectivity and Amazon VPC networking, security auditing, hosting of shared services, billing and cost management.
- **Workload Owners** – Those who are directly responsible for the deployment, development, and maintenance of applications; a workload owner can be a cost center or a department and may include system administrators, developers, and others directly responsible directly for one or more applications.

Enterprise customers establish an MSO when there are common functions that can be centralized to ensure that applications are deployed in a secure and compliant fashion. The MSO can also accelerate the rate of migration through reuse of approved configurations, which minimizes development and approval time while ensuring compliance through the automated implementation of organizational security requirements.

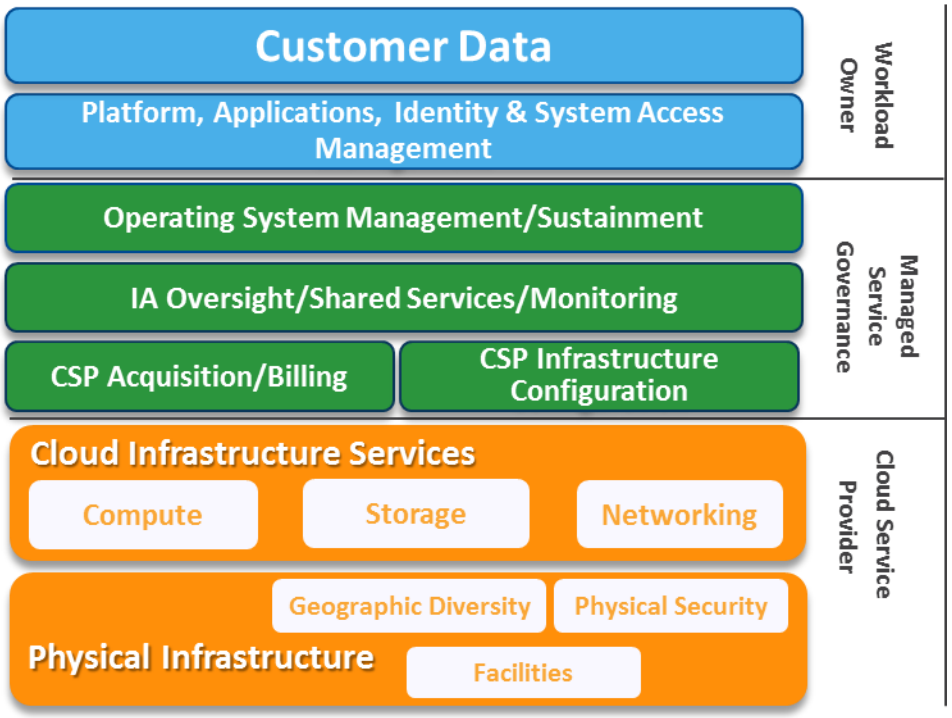


Figure 2: Shared Responsibility Between the CSP, the MSO, and the Workload Owner

Adding an MSO allows the authorization documentation of the workload owner to be scoped down to only the configuration and installation of software specific to a particular application, because the workload owner inherits a significant portion of the security control implementation from AWS and the organization's MSO. Establishing an MSO requires some up front work, but this investment provides enhanced control over applications, increased speed to deployment, decreased time to authorization, and overall enhancement of the enterprise's security posture.

Common Activities of the MSO

MSOs implemented by AWS customers often perform the following activities:

- **Account provisioning.** After reviewing the workload owner's use case, the MSO establishes the initial account, connects it to the appropriate account for consolidated billing, and configures basic security functionality prior to granting access to the workload owner.

- **Security oversight.** Centralized account provisioning allows the MSO to implement features that enable security personnel to monitor the application as it is deployed and managed; the MSO might perform activities such as establishing an auditor group with cross-account access and linking the application VPC to a shared services VPC that is controlled by the MSO.
- **Amazon VPC configuration.** Deploying the VPC and its subnets, including configuring security groups and network ACLs. To maintain tighter control over the application VPCs, the MSO may retain control of VPC configuration and require the workload owner to request desired changes to network security.
- **IAM configuration.** Creating user groups and assignment of rights, including creation of groups for internal auditors, an IAM superuser, and application administrative groups segregated by functionality (e.g., database and Unix administrators).
- **Development and approval of templates.** Creating preapproved AWS CloudFormation templates for common use cases. Using templates allows workload owners to inherit the security implementation of the approved template, thereby limiting their authorization documentation to the features that are unique to their application. Templates can be reused to shorten the time required to approve and deploy new applications.
- **AMI creation and management.** Creating a library of common, approved Amazon Machine Images (AMIs) for the organization, allowing centralized management and updating of machine images. Creating common templates allows the MSO to enforce the use of approved AMIs.
- **Development of a shared services VPC.** A shared service VPC allows the MSO to receive continuous monitoring feeds from the organization's application VPC and to provide common, shared services that are required for their organization. This often includes a shared access management platform, logging endpoints, and the aggregation of configuration information.

Standardizing Architecture for Compliance

The solution to the challenge of implementing security controls for applications running on AWS is to build standardized, automated, and repeatable architectures that can be deployed for common use cases. Automation can help customers easily meet the foundational requirements for building a secure application in the AWS cloud, while providing a level of uniformity that follows proven best practices.

Architectural Baselines

To determine the best method for standardizing and automating architecture in AWS, establish baseline requirements up front. These are the minimum common requirements to which most (or all) workloads must adhere. An enterprise's baseline requirements normally follow pre-existing compliance controls, regulatory guidelines, security standards, and best practices. Typically, a central department or group of individuals who are also involved in the monitoring, auditing, and evaluation of systems that are being deployed establish standard architectures based upon their baseline compliance and operational requirements.

Standard architectures can be shared among multiple applications and use cases within an organization. This provides efficiency and uniformity, and reduces the time and effort spent in designing architectures for new applications on AWS. In an organization with a centralized cloud model, these standard architectures are deployed during the account provisioning or application onboarding process.

Access Control/IAM Configuration

IAM is central to securely controlling access to AWS resources. Administrators can create users, groups, and roles with specific access policies to control which actions users and applications can perform through the AWS Management Console or AWS API. Federation allows IAM roles to be mapped to permissions from central directory services.

The enterprise should determine how to implement the following IAM controls:

- Standard users, groups, or both that will exist in every account

- Cross-account roles or federated roles
- Roles for EC2 instances and application access to the AWS API
- Roles requiring access to S3 buckets and other shared resources
- Security requirements, such as password policies and multi-factor authentication (MFA)

Networking/VPC Configuration

Network boundaries and components are critical to deploying a secure architecture in the cloud. An Amazon VPC is a logically isolated section of the AWS cloud which can be configured to enforce these network boundaries. An AWS account can have one or more Amazon VPCs. Subnets are logical groupings of IP address space within an Amazon VPC and exist within a single Availability Zone (AZ).

A VPC strategy depends on the requirements of a common use case. Amazon VPCs can be designated based on application lifecycle (production, development) or on role (management, shared services). A well-documented Amazon VPC strategy will also take into account:

- The number of Amazon VPCs per AWS account
- The subnet structure within an Amazon VPC: the number of subnets and routing capabilities of each subnet
- High availability requirements: Amazon VPC subnets across availability zones (AZs)
- Connectivity options: internet gateways, virtual private gateways, and routing

AWS provides the components necessary for controlling the network boundaries of an application in an Amazon VPC. The following table lists examples of Amazon VPC networking controls that can be utilized in AWS.

Control	Implementation	Protection Provided
VPC Routing Tables	Control which VPC subnets may communicate directly with the Internet	Provides segmentation and broad reduction of attack surface area per subnet
VPC Network	Subnet-level, all traffic allowed by	Provides blacklist protection for ports

Access Control Lists (NACLs)	default, stateless filtering designed and implemented across one or more VPC subnets	and protocols with security concerns, such as TFTP and NetBIOS
VPC Security Group(s)	Hypervisor-level, all inbound connections denied by default, stateful filtering designed for one or more instances	Provides whitelist abilities for ingress and egress traffic, opening services and protocols required by the instance and applications
Host-based Protection	Customer-selected software to provide intrusion detection and prevention, and firewall and/or logging capabilities.	Depending on product implemented, can provide scalable protection and detection capabilities and security behavior visibility across your virtual fleet

Because VPC networking configuration is critical to ensure the confidentiality, integrity, and availability of an application, enterprises should define standards that adhere to security and AWS best practices. MSOs should follow these standards, or in the case of decentralized deployment, workload owners should have a blueprint to follow when building a VPC structure.

Resource Tagging

Almost all AWS resources allow the addition of user-defined tags. These tags are metadata and are irrelevant to the functionality of the resource, but are critical for cost management and access control. When multiple groups of users or multiple workload owners exist within the same AWS account, restricting access to resources based on tagging is important.

Regardless account structure, tag-based IAM policies can be used to place extra security restrictions on critical resources. The following example of an IAM policy specifies a condition that restricts an IAM user to changing the state of an EC2 instance that has the resource tag of “project = 12345”.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "ec2:StartInstances",
        "ec2:StopInstances",
```

```
          "ec2:RebootInstances",
          "ec2:TerminateInstances"
        ],
        "Condition": {
          "StringEquals": {
            "ec2:ResourceTag/project":"12345"
          }
        },
        "Resource": [

"arn:aws:ec2:your_region:your_account_ID:instance/*"
        ],
        "Effect": "Allow"
      }
    ]
}
```

AWS recommends the following to effectively use resource tagging:

- Establish tagging baselines that define common keys and expected values across all accounts.
- Implement tag enforcement through both auditing and automation methods.
- Use automated deployment with AWS CloudFormation to automatically tag resources.

AMI Configuration

Organizations commonly ensure security and compliance by centrally providing workload owners with pre-built Amazon Machine Images (AMIs). These "golden" AMIs can be preconfigured with host-based security software and be hardened based on predetermined security guidelines. Workload owners and developers can then use the AMIs as starting images on which to install their own software and configuration, knowing the images are already compliant.

Note that managing centrally distributed AMIs can be an involved task for any central team. Do not customize software and configuration, which are likely to

change frequently, in an AMI; instead configure them by using Amazon Elastic Compute Cloud (Amazon EC2) user data scripts or automation tools, such as Chef, Puppet, or AWS OpsWorks.

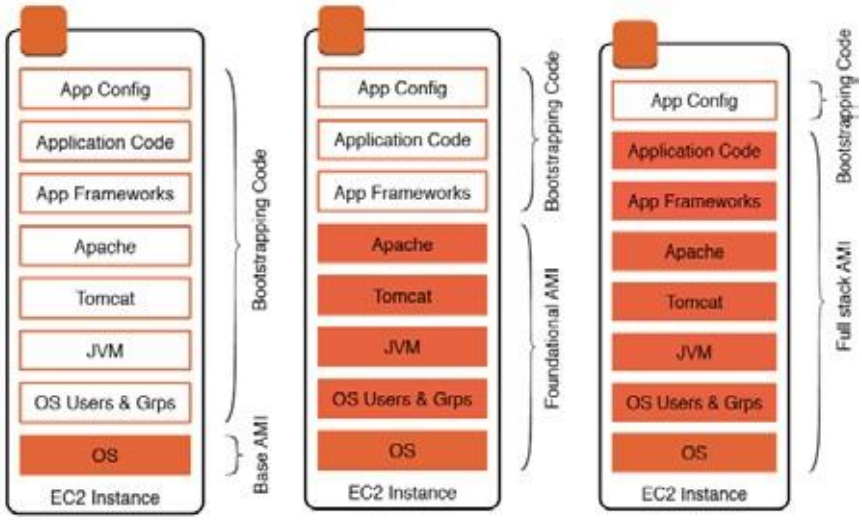


Figure 3: Differences Between Fully-Cconfigured and Base AMIs

Figure 3 shows how preconfigured AMIs can be used through automation and policy as the standard to control which new EC2 instances are deployed by workload owners. Building AMIs can be partially automated by using tools such as Aminator and Packer.[11]

Continuous Monitoring

Continuous monitoring is the proactive approach of identifying risk and compliance issues by accurately tracking and monitoring system activity. Certain compliance standards, such as NIST SP 800-53, require continuous monitoring to meet specific security controls. AWS includes several services and native capabilities that can facilitate a continuous monitoring solution in the cloud.

AWS CloudTrail

AWS CloudTrail is a service that logs API activity within an AWS account and delivers these logs to an Amazon Simple Storage Service (Amazon S3) bucket. This data can be analyzed with third-party tools, such as Splunk, Alert Logic, or CloudCheckr.[12] As a security standard, CloudTrail should be enabled on all accounts and should log to a bucket that is accessible by security tools and applications.

Amazon CloudWatch Alarms

Amazon CloudWatch alarms notify users and applications when events related to AWS resources occur. For example, the failure of an instance can trigger an alarm to send an Amazon Simple Notification Service (Amazon SNS) notification by email to a group of users. You can create common alarms for metrics and events within an account that must be monitored.

Centralized Logging

In AWS, application logs can be centralized for analysis by security tools. This can be simplified by using Amazon CloudWatch Logs. CloudWatch Logs provides an agent, which can be configured to send application log data directly to CloudWatch. Metric filters can then be used to track certain events and activity at the OS and application levels.

Notifications

Amazon SNS can be used to send email or SMS-based notifications to administrative and security staff. Within an AWS account, you can create Amazon SNS topics to which applications and AWS CloudFormation deployments can publish. These push notifications can automatically be sent to individuals or groups within the organization who need to be notified of Amazon CloudWatch alarms, resource deployments, or other activity published by applications to Amazon SNS.

AWS Config

AWS Config is a service that provides you with an AWS resource inventory, a configuration history, and configuration change notifications, all of which enable security and governance.[13] AWS Config allows detailed tracking and notification whenever a resource in an AWS account is created, modified, or deleted.

The Shared Services VPC

Our enterprise customers have found that establishing a single Amazon VPC that contains security applications required for monitoring their applications simplifies centralized control of infrastructure and provides easier access to common features, such as Network Time Protocol (NTP) servers, directory services, and certificate management repositories.

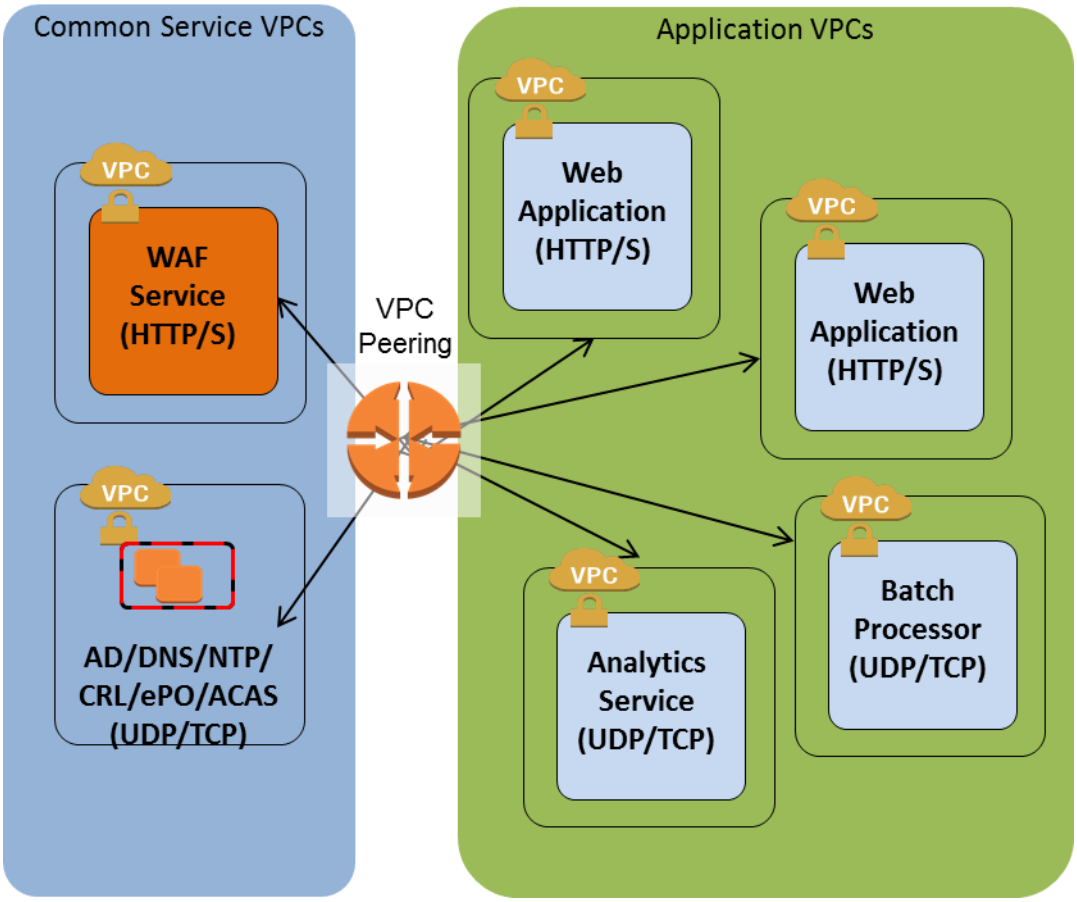


Figure 4: A Sample Shared-Service Amazon VPC Approach for DoD Customers

Figure 4 provides an example of a shared service VPC approach used by a DoD MSO that establishes two VPCs for use by all of their applications. In the first VPC, the MSO established a VPC dedicated to providing a web application firewall that screens all traffic for known attack patterns, creates a single point for monitoring web traffic, and yet does not create a single-point of failure due to its ability to scale with traffic. In the second VPC, the MSO hosts a variety of common services, including Active Directory servers, DNS servers, NTP servers, Host-Based Security System (HBSS) ePolicy Orchestrator (ePO) rollup servers, and a master Assured Compliance Assessment Solution (ACAS) Security Center server.

Each organization must determine the common services that they must host in their AWS environment to support the needs of workload owners.

Automating for Compliance

Any customer can create pre-built and customizable reference architectures with the tools AWS provides, although it does require a level of effort and expertise.

Automation Methods

AWS CloudFormation is the core of AWS infrastructure automation. The service allows you to automatically deploy complete architectures by using pre-built JSON-formatted template files. The set of resources created by an AWS CloudFormation template is referred to as a "stack."

Modular Design for Compliance Automation

When building enterprise-wide AWS CloudFormation templates to automate compliance, we recommend that you use a modular design. Use separate stacks based on the commonality of configuration among applications. This can automate and enforce the baseline standards for security and compliance described in the previous sections.

Figure 5 shows how a customer can develop and maintain AWS CloudFormation templates using a modular design. A single workload would use one template from each of these stacks nested in a single template to deploy and configure an entire application.

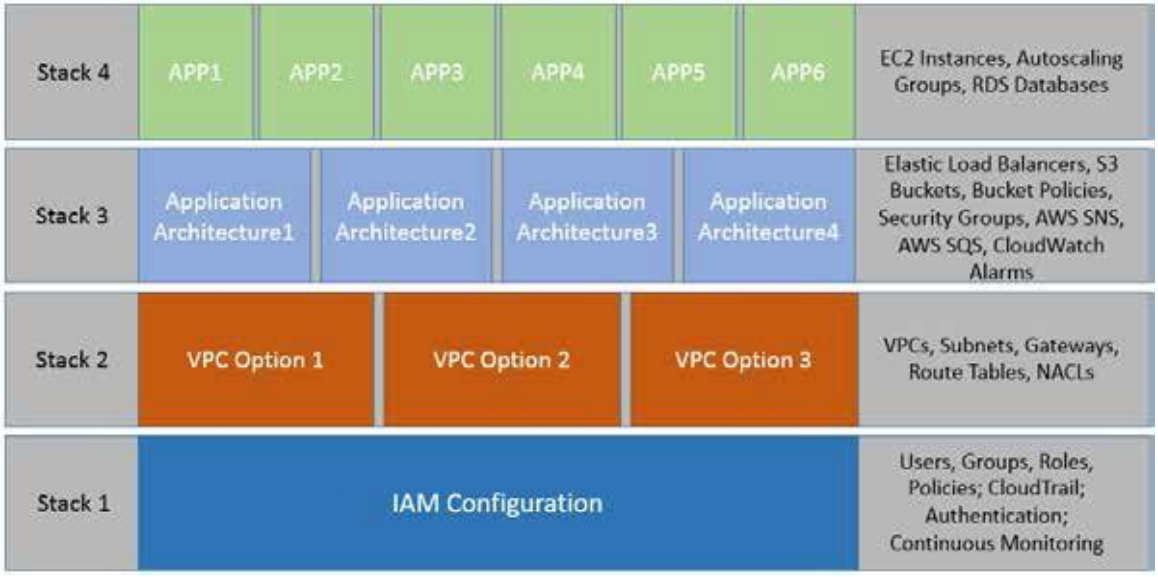


Figure 5: AWS CloudFormation Stacks

Stack 1 – Stack 1 is the primary security template applied to each account; it deploys common IAM users, roles, groups, and associated policies.

Stack 2 – Generally, there will be a template for each common use case to deploy the associated VPC architecture; this can take into account connectivity options such as VPC peering, NAT instances, internet, and virtual private gateways.

Stack 3 – There is a template for each common configuration of an application architecture. They contain application-related components that are common among multiple applications, but distinct among use cases, such as elastic load balancers, Elastic Load Balancing SSL configuration, common security groups, and common S3 buckets.

Stack 4 – There is a template for each specific application that deploys the associated EC2 instances, autoscaling groups, and other instance-level resources. In this stack, instances can be bootstrapped with required user data and other resources, such as application-specific security groups, can be created.

Use Case Packages

Building templates in this manner allows you to reuse configurations. For specific use cases and application types, you can use "packages" that consist of

multiple templates nested within a single main template to deploy an entire architecture, as shown in Figure 6.

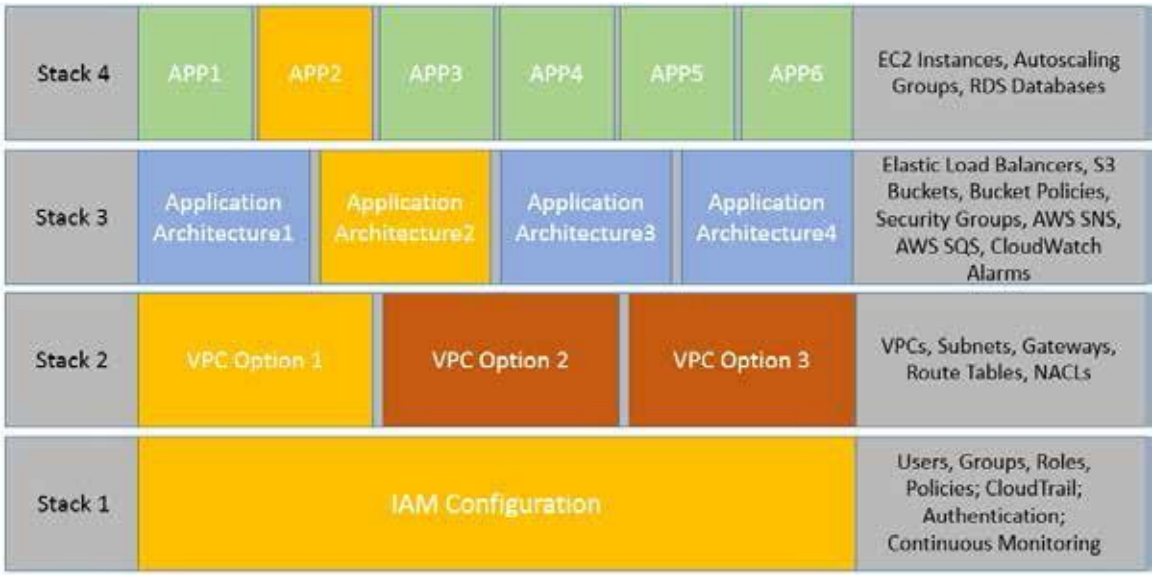


Figure 6: Example Package That Includes IAM Base Configuration, VPC Architecture 1, Application Architecture 2, and APP2 Template

An organization with a decentralized cloud governance model can use this automation structure to establish "blueprint" architectures and allow workload owners full control of deployment at all levels. In contrast, an organization with a centralized cloud team that is responsible for provisioning might allow workload owners to provision only the application-level components of the architecture while retaining responsibility for initial account provisioning, IAM controls, and Amazon VPC configuration.

To successfully build templates to automate compliance:

- Keep templates modular; use nested stacks when possible
- Use parameters as much as necessary to ensure flexibility
- Use the DependsOn attribute and wait conditions to prevent dependency issues when resources are deployed
- Develop a version control process to maintain template packages

- Allow for command line interface (CLI)-based or AWS Service Catalog-based deployment
- Use a parameters file
- Use IAM policies to restrict the ability of users to delete AWS CloudFormation stacks

Automating Compliance for EC2 Instances

There are four tools for automating the configuration of EC2 instances at the operating system and application levels to meet compliance requirements.

Custom AMIs

AWS allows you to create customized AMIs that can be built and hardened for use by workload owners to further install software and applications.

Building a compliant AMI may requires you to take into account the following:

- Software packages and updates
- Password policies
- SSH keys
- File system permissions/ownership
- File system encryption
- User/group configuration
- Access control settings
- Continuous monitoring tools
- Firewall rules
- Running services

User Data Scripts

You can employ user data to bootstrap EC2 instances to install packages and perform configuration on launch. Utilize user data to directly manipulate instance configuration with any of the following tools:

- **Cloud-Init directives** – Specify configuration parameters in user data which cloud-init can use to directly modify configuration. An example of a directive is "Packages," which can install a list of specific packages on the instance.
- **Shell scripts** – Include Bash or PowerShell scripts directly in user data to run on instance launch. There is a 16 KB raw data limit on user data, which limits this option.
- **External scripts** – A user data script can pull down a larger shell script from an S3 bucket URL or any other location and run this script to further configure the instance.

Configuration Management Software

Configuration management solutions allow continuous management of instance configuration. This can automate consistency among instances and make managing changes easier. Examples of such solutions include:

- Chef
- Puppet
- Ansible
- SaltStack
- AWS OpsWorks

By using these configuration management solutions, you can build scripts and packages to secure an operating system. These hardening operations can include modifying user, access, or file system permissions; disabling services; making firewall changes; and many other operations used to secure a system and reduce its attack surface.

The following example of a Chef script implements a password age policy:

```
template '/etc/login.defs' do
              source 'login.defs.erb')
     mode 0444
     owner 'root'
     group 'root'
```

```
    variables
       (password_max_age: node['auth']['pw_max_age'],
        password_min_age: node['auth']['pw_min_age'] )
end
```

You can design packages of configuration scripts, for example, Puppet modules or Chef cookbooks, based on specific compliance requirements and apply them to instances that must meet those requirements.

Containers

Containerization with applications such as Docker[14] or Amazon EC2 Container Service (Amazon ECS)[15] allows one or more applications to run independently on a single instance within an isolated user space.

Figure 7: Containerization

From a compliance perspective, containers can be pre-built with a standardized and hardened configuration based on the operating system and application.

Development & Management

Using a modular approach and a common structure for templates simplifies updates and enforces uniform development by those responsible for creating new use case packages. We recommend using the following elements when developing and managing AWS CloudFormation template packages that are architected for compliance.

Outputs

The Output section of a template can include custom information and can be used to retrieve the ID of generated resources when nested stacks are used. It

can also be used to provide general information that can be viewed from the AWS CloudFormation console or from the CLI/API describe-stacks call.

The Output sections of template files should include, at minimum, the following reference information:

- Use case/application type
- Compliance type
- Date created
- Maintained by

Parameters

AWS CloudFormation parameters[16] are fields that allow users to specify data to the template upon launch. Use parameters whenever possible. You can design an entire set of AWS CloudFormation templates for a common use case by using highly customized parameters.

For example, most tiered web applications share a similar architecture. For this type of use case, you can develop a complete four-stack template package so that multiple web-based applications can easily be deployed with the same template files by the user specifying parameters for AMIs and other application-specific resources.

Conditions

AWS CloudFormation allows the use of Conditions[17], which must be true for resources to be created. When used in combination with parameters, conditions enable you to design templates that make reference architectures flexible and based on application requirements For example, a condition can be used to launch an EC2-based database instead of an Amazon Relational Database Service (Amazon RDS) instance based on input parameters specified by the user, as shown in the following snippet:

```
"CreateDBInstance": {
```

```
      "Fn::Not": [
           {
             "Fn::Equals": [ { "Ref": "DatabaseAmi"   },
"none" ]
           }
          ]
                }
```

Custom Resources

AWS CloudFormation allows you to create custom resources[18], which can be used to integrate with external processes or third-party providers. Custom resources can also be designed to invoke AWS Lambda functions, which can provide levels of automation not available with AWS CloudFormation alone.

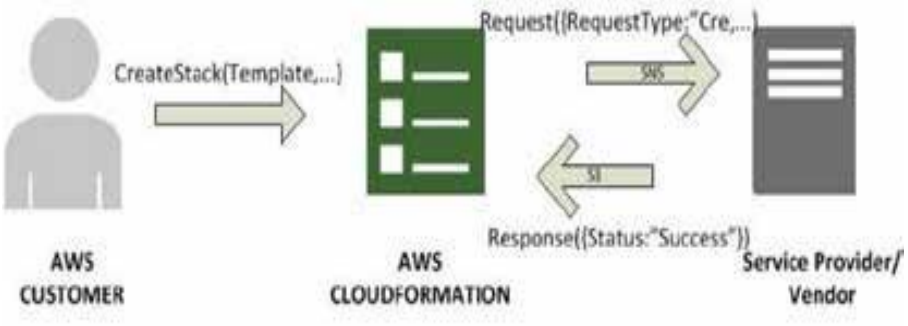
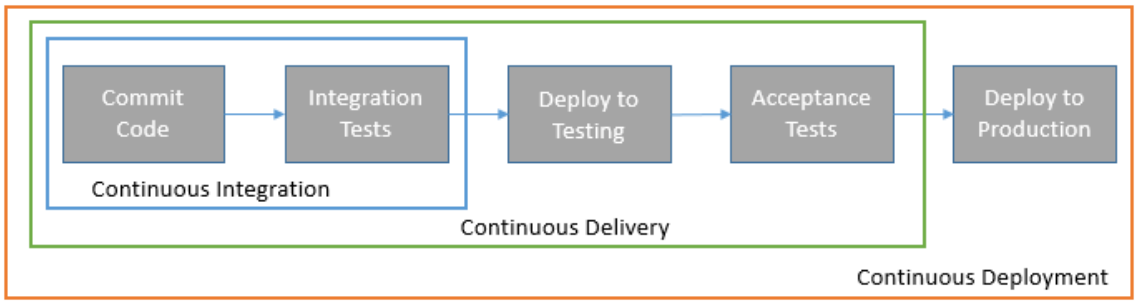


Figure 8: Custom Resources

Infrastructure as Code

AWS CloudFormation templates and associated scripts, documents, and parameter files can be managed just as any application code would be. We recommend that you use version control repositories such as Git or Subversion (SVN) to track changes and allow multiple users to efficiently push updates.

Capabilities such as version control, testing, and rapid deployment are possible with AWS CloudFormation templates just as with any source code. A full Continuous Integration/Continuous Deployment (CI/CD) solution can be implemented using additional tools, such as Jenkins.[19]



CI/CD Stage	Action
Code Commit	AWS CloudFormation JSON Templates committed to code repository
Integration Tests	Compliance tests against template files
Deploy to Testing	Architecture deployed in test AWS account
Acceptance Tests	Compliance tests against deployed resources in test AWS account
Deploy to Production	Architecture deployed in Production AWS account

Figure 9: Example of CI/CD in AWS Using AWS CloudFormation

You can store pre-built use case packages in either a source code repository or in an S3 bucket. This allows provisioning teams and workload owners to easily pull down the latest versions of these files.

Deployment

To ensure a secure, reliable, and efficient deployment of pre-build template packages, you should consider implementing several operational practices, as described in the following sections.

AWS CLI

Although you can use the AWS CloudFormation console to deploy templates from a web-based interface, there are clear advantages to using the AWS CLI and other automated methods – especially if the templates require input to many parameters. The AWS CLI is automatically installed on the Amazon Linux AMI.

You can use the AWS CLI to deploy automated architectures with a single command from an EC2 Linux instance. Including a parameters file simplifies inputting template parameters by eliminating the need to manually input data for each field.

```
aws cloudformation create-stack --stack-name myStack --template-body
file:///template.json    --parameters file:///parameters_file.json --
capabilities
CAPABILITY_IAM
```

You can use an additional script as a wrapper to simplify the CLI command, or, alternatively, to directly call the AWS CloudFormation API to create the stack.

Launch EC2 instances into a pre-defined IAM role that allows access only to the AWS CloudFormation API. To provide "least privilege" within the AWS CloudFormation service, use additional restrictions.

To launch a template from the AWS CLI:

1. Create an IAM role that allows an EC2 instance to access the AWS CloudFormation API.
2. Launch an EC2 instance into the IAM role in a VPC (preferably a shared services VPC).
3. Copy or download the template package to the EC2 instance.
4. Run the AWS CLI `aws cloudformation create-stack` command to launch the template stack.

Security

The security of AWS CloudFormation template packages should always be considered, especially by customers who must adhere to strict compliance requirements. Source code repositories should be secured to allow write access only to those responsible for updating packages. In addition, user names, passwords, and access keys should *never* be included in user data when automating deployment of EC2 instances because they are unencrypted plain text.

It is critical to understand that deleting an AWS CloudFormation stack actually deletes all underlying resources, effectively destroying all data stored in EC2.

	Stack Name	Created Time	Status	Status Reason
	teststack1	2015-06-22 19:49:19 UTC-0400	DELETE_IN_PROGRESS	User Initiated

To mitigate the risk of accidental resource deletion, use the following safeguards.

IAM permissions.[20] Restrict the ability to delete AWS CloudFormation stacks to only users, groups, and roles that require that ability. You can write IAM policies that deny users and groups to which those policies are applied the ability to delete any stack.

The following is an example of an IAM policy that denies the DeleteStack and UpdateStack API calls:

```
{
    "Version":"2012-10-17",
    "Statement":[{
        "Effect":"Deny",
        "Action":[
            "cloudformation:DeleteStack",
            "cloudformation:UpdateStack"
        ],
        "Resource":"*"
        }]
  }
```

Deletion Policy.[21] Resources such as S3 buckets and EC2 and RDS instances support the AWS CloudFormation DeletionPolicy attribute. Use this attribute to require that resources be retained upon stack deletion, or that a snapshot be created (if snapshots are supported).

The following is an example of a deletion policy with an S3 bucket AWS CloudFormation resource:

```
    "myS3Bucket" : {
      "Type" : "AWS::S3::Bucket",
```

```
    "DeletionPolicy" : "Retain"
  }
```

Auditing

Automating architecture deployment in AWS can help simplify the process of auditing and accrediting deployed applications. Having a base configuration for components such as IAM and VPC controls ensures that workload owners are deploying architectures based on compliance standards.

Security personnel at the customer's MSO can "sign off" on reusable template packages that are based on customer security standards and compliance requirements as compliant.

The security accreditation and auditing process can make use of automation with the following AWS capabilities:

- **Tagging**–AWS resources can be queried for common tags. Tags can be applied at the stack level to all resources that support tagging.
- **Template validation**–A scripted validation of the configuration can be tested against the AWS CloudFormation template files prior to deployment.
- **SNS notification**–A nested stack in a template can be configured to send notifications about stack events to an Amazon SNS topic. These Amazon SNS topics can be used to alert individuals, groups, or applications that a specific template has been deployed in the account.
- **Testing deployed resources**–Through the AWS API, scripted tests can be conducted to validate that deployed architectures meet security requirements. For example, tests can be run to detect if any security group has open access to certain ports or if there is an internet gateway in a VPC that should not have one.
- **ISV solutions**–Third-party solutions for analyzing deployed architectures are available from AWS Partners. Security control validation can also be implemented through solutions such as Telos' Xacta risk management solution.

AWS Service Catalog Integration

AWS Service Catalog allows IT administrators to create and manage approved catalogs of resources, which are called products. IT administrators create portfolios of one or more products which they can then distribute to AWS end users and workload owners. End users can access products through a personalized portal.[22]

Product – Products can be created to provide specific types of applications or to address specific use cases, or, alternatively, they can be used to deploy base resources, such as IAM and VPC configuration, which other resources, such as EC2 instances, can utilize. Template package deployment can be further automated and simplified by making the template package an AWS Service Catalog product.

Portfolios – A portfolio consists of one or more products. Portfolios can include products for different types of use cases and can be organized by compliance type.

Permissions – End users and workload owners who are IAM users or members of IAM groups or roles can be given permission to use specific portfolios based on the level of access they need and what they need to deploy.

Constraints – Constraints are a granular control applied at a portfolio or product level that restrict the ways that AWS resources can be deployed. Constraints can be used to allow templates to deploy all resources at a higher level of access than a workload owner has through IAM policies.

Tags – Tags can be used to control access to resources or for cost allocation. Tags are enforced at the portfolio or product level.

AWS Service Catalog allows sharing of portfolios that are created in a common shared services AWS account. This allows central management of and access to deployable reference architectures.

Central Management of AWS Service Catalog

Customers with centralized governance models can fully control and manage the AWS Service Catalog products that workload owners have access to.

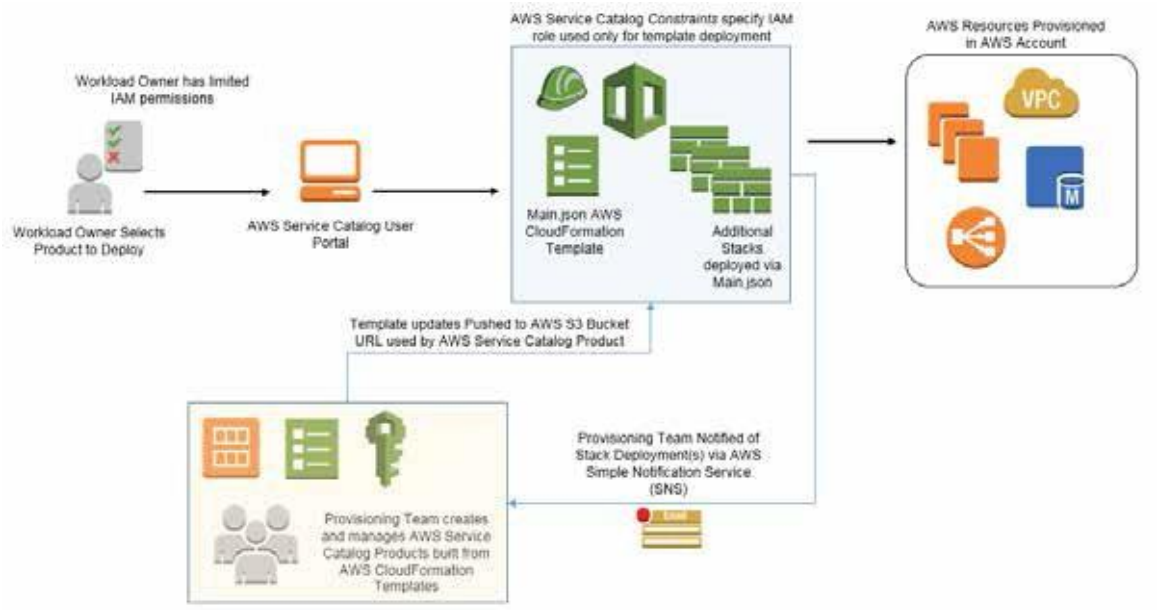


Figure 10: Using AWS Service Catalog Constraints

Automating for Governance: High-Level Steps

Automating a compliant, secure, and reliable architecture that adheres to an organization's governance model involves several basic steps. This section presents a high-level overview.

Prerequisites

Before beginning to develop automated reference architectures based on compliance requirements, your organization must define the following:

- Cloud strategy and roadmap
- Governance model
- Cloud tasks, roles, and responsibilities
- VPC and account creation strategy
- Security standards and compliance requirements

Automating for compliance will often be part of a larger IT transformation initiative. Many architectural requirements relate directly to existing governance and security-related decisions.

Step 1: Define Common Use Cases

Customers must first determine the standard use cases of their workloads. Many applications deployed on AWS support a common use case. These use cases share identical or similar base architectures for VPC design, IAM configuration, and other architectural components.

The following are examples of a few common use cases:

- **Web applications** – Web applications normally consist of multiple tiers (proxy/web, application, and database) for hosting web-based applications accessed by end users. These applications can be designed for scalability and elasticity when properly architected in AWS. Different VPC configurations are required depending on whether the application is intended to be internal facing or accessible from users on the public Internet.
- **Enterprise applications** – Enterprise applications are almost always commercial off-the-shelf (COTS) products that are used widely within an organization in critical-to-business functions. Examples include Microsoft SharePoint, Active Directory, PeopleSoft, and Oracle E-Business Suite. Often, each enterprise application addresses a specific use case with an architecture that is standardized.
- **Data analytics** – Applications that analyze large data sets have architectures that require the deployment of common data analytics applications and use AWS big data services, such as Amazon Redshift, Amazon Elastic MapReduce (Amazon EMR), Amazon Kinesis, and Amazon DynamoDB (DynamoDB).

Step 2: Create and Document Reference Architectures

A well-designed reference architecture provides clear documentation on how resources will be used within AWS. Reference architectures should be created in Visio, PowerPoint, or another platform from which they can be distributed.

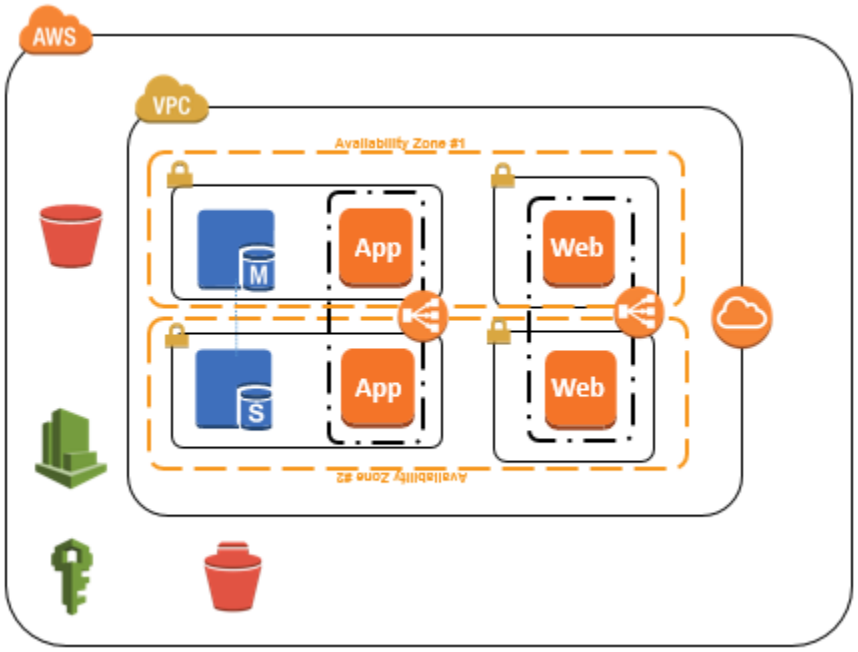


Figure 11: Example Reference Architecture in PowerPoint

Step 3: Validate and Document Architecture Compliance

Accurately documenting how the reference architecture satisfies compliance requirements can reduce the amount of effort required for a workload owner to ensure that the architecture being deployed meets compliance requirements.

Compliance documentation may include:

- A security controls implementation matrix (SCTM)

- A system security plan (SSP)
- A concept of operations (ConOps)

Organizations that must follow specific compliance controls should determine which resources, components, and configurations meet the requirements of each control. Including this documentation in a packaged deployment reduces the need to repeat the same compliance analysis for a proposed architecture.

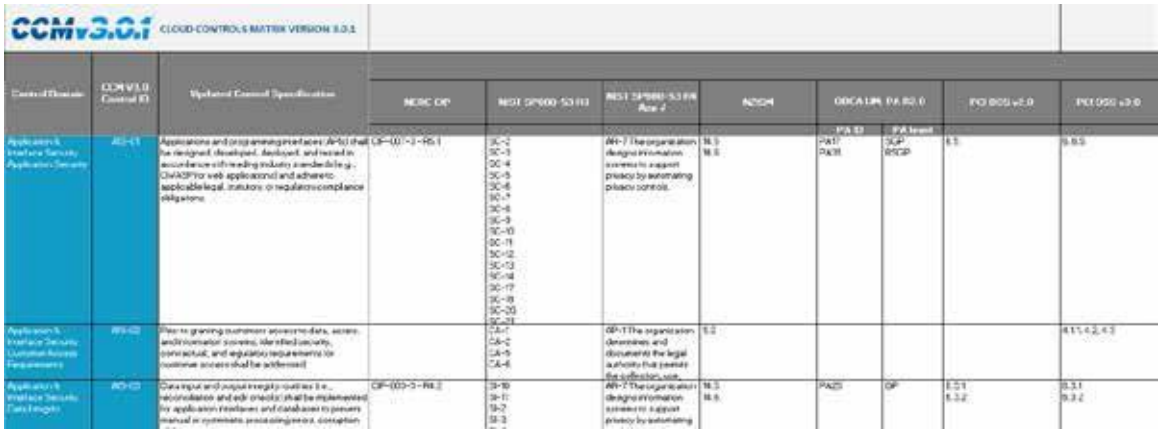


Figure 12: Example of a Security Controls Implementation Matrix Provided by the Cloud Security Alliance

Step 4: Build Automated Solutions Based on Architecture

There are many ways to automate infrastructure creation with AWS services and features. Most commonly, AWS CloudFormation templates are used to automate deployment and configuration of AWS resources-. Create template packages using the design guidelines provided in "Automating for Compliance," earlier in this whitepaper.

When building templates, determine which configurations are common among various types of applications and use cases. Properly maintain and update templates when necessary.

Step 5: Develop an Accreditation and Approval Process

Existing processes and methods for evaluating systems against compliance requirements may not apply or may need to be changed for applications in the cloud. When automating compliance for an entire enterprise, involve security teams early on so they can provide input and gain a deeper understanding of how applications will be deployed in AWS.

The accreditation and approval plan for automated deployments should consider of all of the following:

- The compliance standards that the organization must follow
- The current approval process for applications and infrastructure
- The existing security requirements related to networking, continuous monitoring, access control, and auditing
- The current (and proposed) tools for security analysis, scanning, and monitoring
- The hardening requirements for deployed operating systems, if there are any, and the need for pre-hardened custom images
- The processes and methods used to validate both architecture templates and deployed configurations

Conclusion

Developing an automated solution for governance and compliance can reduce the cost, time, and effort to deploy applications in AWS, while minimizing risk and simplifying architecture design. When this approach is packaged into a reusable solution, it can decrease the level of effort to produce compliance-related documentation and allow time normally spent evaluating compliant architectures to be used to drive the organization's goals and mission.

Contributors

The following individuals and organizations contributed to this document:

- Mike Dixon, Consultant, AWS Public Sector Sales
- Lou Vecchioni, Senior Consultant, AWS ProServ
- Brett Miller, Senior Consultant, AWS ProServ
- Josh Weatherly, Practice Manager, AWS ProServ
- Andrew McDermott, Senior Compliance Architect, AWS Security

Notes

[1] http://www.gartner.com/it-glossary/it-governance/

[2] http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r4.pdf

[3] http://d0.awsstatic.com/whitepapers/compliance/aws-architecture-and-security-recommendations-for-fedramp-compliance.pdf

[4] http://iase.disa.mil/cloud_security/Documents/u-cloud_computing_srg_v1r1_final.pdf

[5] http://aws.amazon.com/compliance/hipaa-compliance/

[6] http://www.27000.org/iso-27001.htm

[7] http://aws.amazon.com/compliance/pci-dss-level-1-faqs/

[8] http://media.amazonwebservices.com/AWS_Security_at_Scale_Governance_in_AWS.pdf

[9] http://aws.amazon.com/partners/managed-service/

[10] https://media.amazonwebservices.com/AWS_Security_at_Scale_Governance_in_AWS.pdf

[11] https://github.com/Netflix/aminator

https://www.packer.io/intro/index.html

[12] http://aws.amazon.com/cloudtrail/partners/

[13] http://aws.amazon.com/config/

[14] https://www.docker.com/

[15] http://aws.amazon.com/ecs/

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

[17] http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/conditions-section-structure.html

[18] http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-cfn-customresource.html

[19] https://wiki.jenkins-ci.org/display/JENKINS/AWS+Cloudformation+Plugin

[20] http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/using-iam-template.html

[21] http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-attribute-deletionpolicy.html

[22] http://aws.amazon.com/servicecatalog/