# INFRASTRUCTURE CONFIGURATION MANAGEMENT *"How do I implement a configuration management solution on AWS?"*

## Overview

Amazon Web Services (AWS) offers its customers several methods to help configure and manage infrastructure deployed on the AWS cloud. These methods range from AWS-managed configuration management services to third-party AWS Partner Network (APN) products, however it is not always easy to decide between the potential options. This document provides best practices and guidance to consider when choosing configuration management solutions for managing changes to AWS resources, Amazon Elastic Compute Cloud (Amazon EC2) instance operating systems, application stacks, or other infrastructure.

The following sections assume basic knowledge of AWS resource management, Amazon EC2, and operating system (OS) administration, management, and configuration.

## General Best Practices

There are several best practices to consider when managing infrastructure configuration. First, it is important to understand the types of resources to manage, and their different characteristics that must be accounted for in a configuration management system. For example, the configuration and orchestration of AWS resources (such as security groups, Auto Scaling groups, Elastic Load Balancing load balancers, etc.) is very different than the configuration and orchestration of OS and application stack changes. It is also important to recognize that manual configuration of distributed systems is time consuming, error prone, and can lead to inconsistently configured systems. Therefore, the ability to automate, monitor, and track configuration changes is a key component of any configuration management solution.

When implementing an infrastructure configuration management solution, incorporate the following best practices:

- Ensure the solution covers configuration management for all infrastructure components including AWS resources, OS, and application stacks.
- Provide a mechanism to describe resource configurations in a reusable way.
- Provide version-controlled repositories of configuration templates that can be used and customized as needed.
- Leverage scripting languages and operating systems that an organization is already using or is comfortable developing and maintaining expertise in.

## Application on the AWS Platform

The following sections briefly describe AWS-managed services and third-party tools that AWS customers commonly use for configuration management on AWS.[1] While each of these options can be used independently, most AWS customers leverage a combination of these options to ensure configuration management coverage for both their AWS resources as well as their OS and application stack resources.

AWS recommends using the following combination of configuration management tools:

- AWS Config with Config Rules or an AWS Config Partner[2] to provide a detailed, visual, and searchable inventory of AWS resources, configuration history, and resource configuration compliance.
- AWS CloudFormation or a third-party AWS-resource orchestration tool to manage AWS resource provisioning, update, and termination.
- AWS OpsWorks or a third-party server configuration management tool to manage OS and application stack configuration changes.

[1] Additional third-party tools include numerous, but less ubiquitous, open source and commercial configuration management tools.
[2] http://aws.amazon.com/config/partners/

## Inventory and Configuration Tracking of AWS Resources

The following tools help customers identify their AWS resources and how these resources change over time.

### AWS Config and Config Rules

AWS Config provides a detailed inventory of the current configuration of AWS resources and continuously records configuration changes, such as the value of tags on Amazon EC2 instances, ingress/egress rules of security groups, and network ACL rules for VPCs.[3] Customers can use AWS Config to determine how a resource was configured at any point in time, to view resource dependencies, and to send notifications when the resource configuration changes. AWS Config Rules is a new set of capabilities that allow customers to evaluate whether their AWS resources comply with desired configurations. Customers can use either predefined, AWS-managed rules or define their own, and use these rules to evaluate AWS resource compliance.

## Change Orchestration for AWS Resources

The following tools assist with the coordination and management of changes to AWS resources, including resource creation, modification, and termination.

### AWS CloudFormation and AWS Service Catalog

AWS CloudFormation enables customers to model, create, modify, and terminate collections of AWS resources using pre-defined, reusable infrastructure templates. AWS CloudFormation simplifies infrastructure management, allowing customers to provision and manage AWS resources as a single unit, without worrying about resource dependencies. Customers can use change sets to preview how proposed updates to a stack might impact running resources, allowing them to perform updates with more confidence and predictability.

Along with AWS resource creation and management, customers can use AWS CloudFormation to bootstrap Amazon EC2 instances to automate the initial installation and configuration of the OS and application stack, and also of third-party configuration management tools. AWS Service Catalog extends AWS CloudFormation by allowing IT administrators to group related AWS CloudFormation templates into product portfolios to publish to their AWS users, ensuring compliance and organizational consistency.

Note that if AWS CloudFormation is used to instantiate an AWS resource, it must be used for making ongoing changes to that resource. Otherwise, the resource can become out of sync with AWS CloudFormation and cause undesired results when using AWS CloudFormation to make changes in the future.

### AWS Elastic Beanstalk and AWS OpsWorks

AWS Elastic Beanstalk (Elastic Beanstalk) is an AWS managed service for deploying and scaling web applications. Elastic Beanstalk automatically handles the provisioning and configuration of Amazon EC2 instances, Elastic Load Balancing load balancers, Auto Scaling, application deployment, and application health monitoring. It also provides the ability to integrate with additional AWS services such as Amazon Relational Database Service (Amazon RDS) or Amazon ElastiCache instances and to retain full control over automatically provisioned AWS resources.

AWS OpsWorks is a fully managed application-stack management service that includes provisioning and configuration of Amazon EC2 instances, Auto Scaling, application deployment, and application health monitoring. It also provides the ability to integrate with additional AWS services such as Elastic Load Balancing and Amazon RDS instances.

[3] Please see http://aws.amazon.com/config/details/ for a list of supported AWS resources.

## OS and Application Stack Management

AWS offers managed services to assist with the installation and ongoing management of OS and application stacks.

### AWS Elastic Beanstalk

AWS Elastic Beanstalk (Elastic Beanstalk) environment configuration[4] extensions enable the automatic installation and configuration of advanced OS and application stack settings during application deployment. This allows customers to push infrastructure configuration changes to running instances in conjunction with application deployments. Since infrastructure changes are bundled with application deployments, AWS does not recommend making infrastructure changes independently from application deployments.

### AWS OpsWorks

AWS OpsWorks allows administrators to implement Amazon EC2 instance configuration changes on demand or automatically based on lifecycle events. Specific events will automatically trigger the appropriate community- or customer-defined recipes to configure the required OS, application stack, or application changes. AWS OpsWorks supports the implementation of infrastructure changes either in conjunction with or independently from application deployments.

## Partners and Third-Party Tools

AWS customers can select from different AWS Partner Network (APN) configuration management tools to complement and extend AWS managed services. This section discusses common third-party options as well as specific guidance for selecting a stack-level configuration management tool.

### AWS Config Partners

AWS Config Partners integrate their configuration and change management products with AWS Config to provide additional visualization, search, and configuration management capabilities for AWS customers. See the AWS Config Partner website for a list and description of these product offerings.

### HashiCorp Terraform

Similar to AWS CloudFormation, AWS Technology Partner HashiCorp's Terraform uses configuration files to describe and deploy AWS infrastructure. Terraform separates infrastructure deployment planning from the execution phase. It refreshes and reviews current resource configuration state before generating an action plan. The plan includes all actions to create, modify, or terminate resources. Once the plan is captured, customers can limit the execution phase to include select actions. HashiCorp also offers Terraform graph, which allows organizations to visualize AWS resource dependencies to better understand the scope of a configuration change before deciding to implement it.

### Stack-Level Tools

An effective stack-level tool provides a well-defined method to represent and reuse configurations, mechanisms to manage and communicate required changes to EC2 instances, and extensibility to allow for new features and functionality. Third-party tools are often appropriate for customers in the following situations:

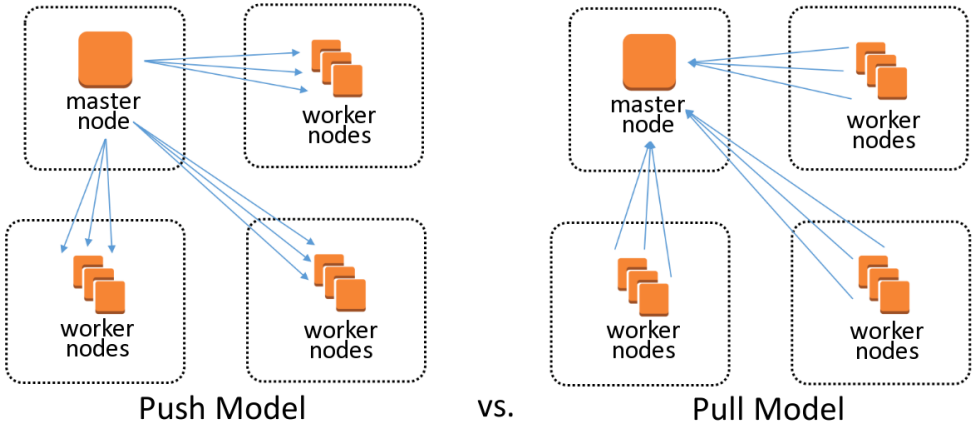
- They require customizations or code changes to the tool that an AWS service does not offer natively.
- They require a specific features or functionalities that are not otherwise available.
- They already have existing expertise and investment in a specific tool that they would like to leverage on the AWS platform.

[4] http://docs.aws.amazon.com/elasticbeanstalk/latest/dg/customize-containers.html

Selecting the right tool involves a number of variables, including programming language and configuration file preference, operating system requirements, and what system the tool uses to manage configurations (see the next section on push and pull models). Additionally, pre-defined templates can significantly reduce a tool's learning curve and jumpstart the amount of time it takes an organization to get started. Therefore ensure to evaluate a tool's community and pre-defined template repositories for templates that are applicable to your organization's OS and application stack configuration requirements.

**Topologies**

Generally speaking, these tools operate in a push or pull model. In the *push model*, a centralized master server pushes configuration changes to all worker nodes. In the *pull model*, worker nodes are configured to check-in and retrieve configuration details from a central master server.




Deciding on whether to use a push or a pull model depends on the types of systems the tool will manage. If there is a known set of systems that a centralized master server can easily identify, the push model can be effective. To implement this model, all systems must have network connectivity to the master server, and OS firewalls and security groups must allow communication between their instances and the master server.

AWS customers with more dynamic systems, such as those using Auto Scaling, often prefer to use the pull model, where OS firewall and security group settings are only required on the master side. This allows for the management of any instance that has connectivity to the master server.

**Common Third-Party Tools**

These are some of the most common tools that AWS customers implement for OS and application stack configuration management:

- **Chef** is written in Ruby and uses Ruby DSL for its configuration files. Chef uses a client pull model by default.[5]
- **Puppet** is also written in Ruby, but utilizes a custom DSL for its configuration files. Puppet uses a client pull model by default.[5]
- **Ansible** is written in Python and uses YAML-based configuration files. Ansible is considered agentless because it utilizes SSH and PowerShell for making configuration changes and most closely resembles a push model.
- **Salt** is also written in Python, but uses a custom format for its configuration files. Salt uses a push model.

[5] AWS offers automated, customizable Quick Start Reference Deployments for Chef Server and Puppet. For more information on available AWS Quick Starts, see https://aws.amazon.com/quickstart/

# Resources

| | |
|---|---|
| Managing Your AWS Infrastructure at Scale | https://d0.awsstatic.com/whitepapers/managing-your-aws-infrastructure-at-scale.pdf<br>AWS Whitepaper on deploying and managing infrastructure on AWS in a scalable and predictable way |
| AWS Config | https://aws.amazon.com/config/ |
| AWS CloudFormation | https://aws.amazon.com/cloudformation/ |
| AWS Service Catalog | https://aws.amazon.com/servicecatalog/ |
| AWS OpsWorks | https://aws.amazon.com/opsworks/ |
| AWS Elastic Beanstalk (Elastic Beanstalk) | https://aws.amazon.com/elasticbeanstalk/ |
| HashiCorp Terraform | https://www.terraform.io/ |
| Ansible | http://www.ansible.com/aws |
| Chef | https://www.chef.io/ |
| Puppet | https://puppetlabs.com/ |
| Salt | http://saltstack.com/ |
| AWS Quick Starts | https://aws.amazon.com/quickstart/<br>AWS CloudFormation templates and supporting materials to rapidly deploy fully functional software on the AWS cloud. |