AWS Whitepaper

Migrating Magento Open Source or Adobe Commerce on Cloud Infrastructure Self-Service to AWS

Migrating Magento Open Source or Adobe Commerce on Cloud Infrastructure Self-Service to AWS: AWS Whitepaper

Table of Contents

Migrating Magento Open Source or Adobe Commerce on Cloud Infrastructure Self-Service to AWS

Publication date: **July 19, 2023** (*Document history*)

Adopting AWS for the Magento Open Source or Adobe Commerce on Cloud Infrastructure Self-Service installation presents many benefits such as increased business agility, flexibility, and reduced costs. This whitepaper outlines the benefits of cloud hosting and considerations for migrating Magento Open Source or Adobe Commerce on Cloud Infrastructure Self-Service installation to AWS. It also provides guidance for an organization that plans to AWS their cloud footprint. The content targets technical leaders and business leaders responsible for deploying and managing on-premises Magento Open Source or Adobe Commerce on Cloud Infrastructure Self-Service or enterprise edition on AWS.

Introduction

Creating a new or migrating an existing on-premises Magento Open Source or Adobe Commerce on Cloud Infrastructure Self-Service setup to Amazon Web Services (AWS) cloud presents an opportunity to transform your organization by lowering costs, increasing agility, and delivering it reliably and globally. This whitepaper presents a cloud migration strategy and considerations when migrating Magento to AWS.

This whitepaper provides general guidance for cloud migration with specific steps related to migrating Magento open-source installation to cloud and use of the AWS Terraform Quick Start to deploy Magento open-source edition on the AWS Cloud. In addition, the paper provides AWS reference architecture guidance to enable an organization that wants to install Magento enterprise edition on AWS. The first section of the whitepaper describes the reasons to migrate to the cloud and the common challenges that organizations face when migrating to the cloud. Next, the migration process and the migration strategies that organizations can choose from as well as deployment options are discussed. Lastly, the whitepaper concludes by discussing steps necessary to deploy the Quick Start along with security and compliance, architectural components, connectivity and a strategy you can adopt for migration.

The TerraForm guide is for IT infrastructure architects, administrators, and DevOps professionals who are planning to implement or extend their Magento Open Source community editions workloads on the AWS Cloud using the Terraform Quick Start on Amazon Web Services (AWS).

Are you Well-Architected?

The AWS Well-Architected Framework helps you understand the pros and cons of the decisions you make when building systems in the cloud. The six pillars of the Framework allow you to learn architectural best practices for designing and operating reliable, secure, efficient, cost-effective, and sustainable systems. Using the AWS Well-Architected Tool, available at no charge in the AWS Management Console, you can review your workloads against these best practices by answering a set of questions for each pillar.

For more expert guidance and best practices for your cloud architecture—reference architecture deployments, diagrams, and whitepapers—refer to the AWS Architecture Center.

Drivers for migrating to the cloud

The drivers behind starting a new Magento Open Source or Adobe Commerce on Cloud Infrastructure Self-Service setup or moving an existing on-premises setup to the cloud are numerous but the most common strategic drivers include: reducing capital expenditure, decreasing ongoing cost, improving scalability and elasticity, improving time-to-market, and attaining improvements in security and compliance. In addition, situational and business drivers also influence the move to the cloud.

DevOps

Supporting your organization's DevOps strategy may be a primary driver for migrating to cloud or may be an unanticipated benefit. In either case, migrating to cloud provides a technical foundation to supporting your organization's DevOps strategy by way of the same capabilities expected of cloud and as defined by NIST: On demand and self-service, broad network access, pooled resources, rapid elasticity—all delivered as a metered service providing you the ability to control how your organization consumes it.

Each of these capabilities directly maps to demands placed on a technology organization, regardless of your organization's adoption of DevOps, Site Reliable Engineering (SRE), and so on, but most importantly, it is the ability to programmatically define infrastructure and configuration, that is, infrastructure as code (IaC), and using that ability to dynamically create/tear down environments as part of a well implemented software development life cycle (SDLC) process.

In addition to providing a supporting technology platform for the enablement of DevOps processes on AWS around Magento Open Source or Adobe Commerce on Cloud Infrastructure Self-Service to AWS provides a collection of services that can offer (in the absence of) or augment your existing software configuration management (SCM) solutions, including AWS CodeCommit, AWS CodeBuild, AWS CodePipeline, and AWS CodeDeploy, which provides for a managed source control, build, continuous integration/continuous deployment (CI/CD) and deployment services.

Data center consolidation

Data center consolidation is a key requirement driver that may warrant the need to move to the cloud. For example, an organization's current data center can no longer support the business need for growth in terms of current space, power, and cooling. Also, an organization's current data center may have too many single point of failures and carry inherent risks of outages.

Mergers and acquisitions

Mergers and acquisitions are a situational driver, and might require an organization to separate and consolidate application setup. Also, an organization may face the sale of a building, rental fees, or increases in co-location costs that may result in similar needs to consolidate or separate application setup, leading to the need to move to the cloud.

Digital transformation

Digital transformation is more than simply digitizing data, and it involves the transformation of business and organizational activities, processes, competencies to accelerate deliverables that differentiate an organizations core business. The need for digital transformation in organizations have resulted in the evolution of organization's IT department to become more agile and innovative to adapt to the changing needs of an organization.

AWS Cloud infrastructure setup frees an organization's IT department from the heavy lifting of racking, stacking, and powering servers to focus on the organization's own customers. Concentrating on the projects that differentiate an organization's core business, rather than the infrastructure, substantially improves products, services, delivery, and ultimately the ability to compete.

Benefits of the cloud

Organizations considering a transition to the cloud are often driven by their need to become more agile and innovative. The traditional capital expenditure (CAPEX) funding model makes it difficult to quickly test new ideas. The AWS Cloud model gives you the agility to quickly spin up new instances on AWS, and the ability to try out new services without investing in large upfront, sunk costs (costs that have already been incurred and can't be recovered). AWS helps lower customer costs through its pay-for-what-you-use pricing model.

Operational improvements

The value proposition of migration of on-premises Magento Open Source or Adobe Commerce on Cloud Infrastructure Self-Service or enterprise edition to AWS is further enhanced by the availability of several services that provide for operational insight and agility.

Operational insight into the platform from not only a technical perspective (for example, requests per hour) but also from a business operational perspective (for example, orders per

hour), particularly when the two sets of data can be married, providing a near real-time look into campaign performance, platform operations costs, and a variety of other indicators.

This data provides a basis for, and support of, change as does the agility afforded by AWS, allowing for a content and functional deployment pipeline, A/B testing and feedback loops, allowing for continuous improvement, measurement and pivoting when it comes to the user experience of Magento Open Source or Adobe Commerce on Cloud Infrastructure Self-Service on AWS.

Technology improvements

The momentum of a migration presents an opportune time to look at technical improvements. Enabled with AWS, these technical improvements can be implemented in a commitment-free manner, evaluated and codified.

A typical Magento on-premises installation might leverage as little as one server or numerous servers, based on scale and architecture, but in addition to servers, third-party services might be leveraged as well, such as content delivery networks, and indexing services.

The breadth and depth of the AWS service offerings, as well as the billing constructs available on AWS serve to eliminate the baseline quantity of servers to be managed as well as provide a means to consolidate vendors, achieve larger quantities of scale, and have predictable baseline and burst cost models as it relates to operating the infrastructure. This is exclusive of the potential gain in efficiency around management of the platform, furthering lowering operating costs.

Cloud adoption challenges

Cloud has become a key pillar of most enterprises' digital transformation strategies. Organizations are both migrating new processes to the cloud and augmenting their existing cloud operations to take advantage of new and evolving services. However, these are complex initiatives that many organizations stumble through because of the following constraints and concerns.

Cost and budget constraints

A core reason why organizations adopt a cloud IT infrastructure is to save money. However, there are budget and cost constraints about the moving and running operations in the cloud that often pose challenges for organizations. AWS offers cost estimation and budgeting tools, such as AWS Cost Explorer, AWS Cost and Usage Reports, and AWS Budgets, that can guide organizations and alleviate these concerns.

Security concerns

Organizations are accustomed to having full ownership from the physical building that house the servers to the software on the servers. This ownership made people comfortable that the data was secure. Ownership and the geographic placement of data have become major topics for cybersecurity and cloud policy initiatives around the globe.

As technology has evolved, however, most threats are exploited remotely. The physical location of data has little to no impact on threats propagated over the internet. Organizations often have the misconception that cloud environment is less secure. In addition, organizations IT departments are well versed with security of on-premises infrastructure but often struggle with implementing security in the cloud due to lack of knowledge or skills.

Responding to business requirements

Organizations struggle to identify the different end customers, such as shoppers, content creators, customer service agents, and merchandisers, in a typical e-commerce application setup. This becomes a challenge in a cloud migration initiative to respond and work backwards from the requirements for each customer type.

Managing legacy infrastructure

Organizations may not realize the cost savings immediately from moving Magento Open Source or Adobe Commerce on Cloud Infrastructure Self-Service on-premises installation to the cloud because in some cases the on-premises infrastructure cannot be decommissioned while other services and applications are still running on that infrastructure. This results in expanding the scope for organizations to manage existing legacy infrastructure along with the cloud infrastructure.

Competing projects, staffing concerns, and skills shortages

Organizations struggle with cloud-related skills shortages in existing staff and staffing concerns to hire cloud trained staff. In addition, competing priorities and projects between multiple cloud initiatives at an organization lead to the IT staff split between projects. This ironically leads to slow and costly processes to implement and optimize systems meant to deliver speed and agility. AWS Cloud and managed service providers, such as CloudHesive, offer support services, which have become much more proactive and strategically focused in response to market demand for increased levels of responsiveness, access to tools, and strategic guidance.

Vendor management

Organizations often lack resources to manage vendors and see cloud service providers an addition to that list. Also, getting the right documentation, contract management and compliance reports are other challenges that add to the complexity. AWS, however, simplifies these challenges with its self-service approach and availability of compliance reports online.

Five-phase migration process

When considering the migration of a Magento Open Source or Adobe Commerce on Cloud Infrastructure Self-Service to the cloud, it is typically just one part of an overall migration plan. For example, a company embarking on a modernization journey to move from a data center or co-location facility to AWS must develop a detailed migration plan that considers the sequence and strategy for moving applications and platforms with the least impact on ongoing business operations.

This section covers the five-phase migration process in the context of migrating Magento Open Source or Adobe Commerce on Cloud Infrastructure Self-Service on-premises installation workloads to the cloud.

Phase 1: Migration preparation and business planning

In this phase, you gain a complete understanding of the benefits of migrating to the cloud as well as establishing your business objectives. A business case for the migration is developed and the constraints of existing architectures and systems are considered. This is also where AWS Partners who specialize in cloud migrations and Magento Open Source or Adobe Commerce on Cloud Infrastructure Self-Service deployments can be engaged to help develop a migration plan.

Phase 2: Portfolio discovery and planning

Next, you inspect your entire IT portfolio, understanding any dependencies that exist between workloads, and consider the migration strategies to meet your business objectives.

For Adobe, this phase details how Magento Open Source or Adobe Commerce on Cloud Infrastructure Self-Service integrates with other workloads, both internal and external to the business.

Phases 3 and 4: Designing, migrating, and validating applications

This is where the migration strategy for each application is designed, performed, and validated. There are six common application migration strategies which are discussed in more detail in the next section.

Phase 5: Modern operating model

Finally, this phase is where you iterate on your new foundation, retiring old systems, and continuing to improve and move toward a modern operating environment.

Migration strategies

This section provides the six common migration strategies for moving applications and systems to the cloud and then describes which of those strategies apply to on-premises Magento Open Source or Adobe Commerce on Cloud Infrastructure Self-Service or enterprise edition on AWS. workloads. It's useful to understand these broader strategies since Magento Open Source or Adobe Commerce on Cloud Infrastructure Self-Service is typically just one component of a portfolio of applications and therefore part of an overall migration plan.

Six common strategies: "Six Rs"

The six approaches described below are common migration strategies and build upon "The 5 Rs" outlined by Gartner in 2011. Choosing the right on-premises Magento Open Source or Adobe Commerce on Cloud Infrastructure Self-Service or enterprise edition on AWS migration strategy depends upon the business drivers for cloud adoption, as well as time considerations, business and financial constraints, and resource requirements.

Re-host

Rehosting, or *lift and shift*, is typically used when an organization is looking to quickly migrate applications to the cloud to meet a business case. Applications are moved as-is to the cloud without making any changes to the application or its dependencies. Although this strategy does not immediately bring the full benefits of the cloud, it allows for a swift migration and cost savings from hosting in the cloud.

Re-platform

Also referred to as *lift, tinker, and shift*, re-platforming involves taking an existing application, migrating it to the cloud, and replacing specific application dependencies with fully managed alternatives available in the cloud. For example, rather than directly hosting a relational database on EC2 instances, the database for many applications can be easily replaced by Amazon Relational Database Service (Amazon RDS). The benefit to this strategy is that the operating responsibility of undifferentiated components can be offloaded to AWS without requiring significant changes to the core application.

Re-purchase

Re-purchase strategy involves moving from perpetual licenses to a software-as-a-service (Saas) model. In context of Adobe, a re-purchase strategy would involve moving from a traditional on-premises Magento Open Source or Adobe Commerce on Cloud Infrastructure Self-Service or enterprise edition on AWS (often referred to as M1) to Magento Commerce cloud by Adobe.

Re-factor / Re-architect

Re-factor or Re-architect strategy gives the most opportunity to optimize and re-skin or re-imagine the application architecture from ground up. This presents an opportunity to deliver features using cloud native technologies. This strategy is often driven by strong business need to add features, scale, or performance that would otherwise be difficult to achieve in the application's existing environment.

Retire

This strategy involves completing a discovery of the existing environment and removing applications and features that are no longer needed. For example, hardware monitoring may not be required if you are planning to move to a managed Magento Open Source or Adobe Commerce on Cloud Infrastructure Self-Service solution.

Retain

This is also referred to as a *re-visit* strategy or do nothing. This strategy involves revisiting the existing Magento Open Source or Adobe Commerce on Cloud Infrastructure Self-Service application at a later point in time because migrating to cloud may not align with current business needs.

Magento versions

Magento is available in two versions:

- Magento Open Source (formerly known as Community Edition)
- Magento Commerce

For version differences, see the feature comparison on the Adobe Commerce website.

Adobe Commerce Open Source is available only for self-hosting. Magento Commerce is available for self-hosting or as part of Magento Commerce Cloud, now known as Adobe Commerce.

Reference architecture

The following figure shows the reference architecture for deploying either version of Magento on AWS. For reference architecture details, see Hosting Magento eCommerce Software on AWS.

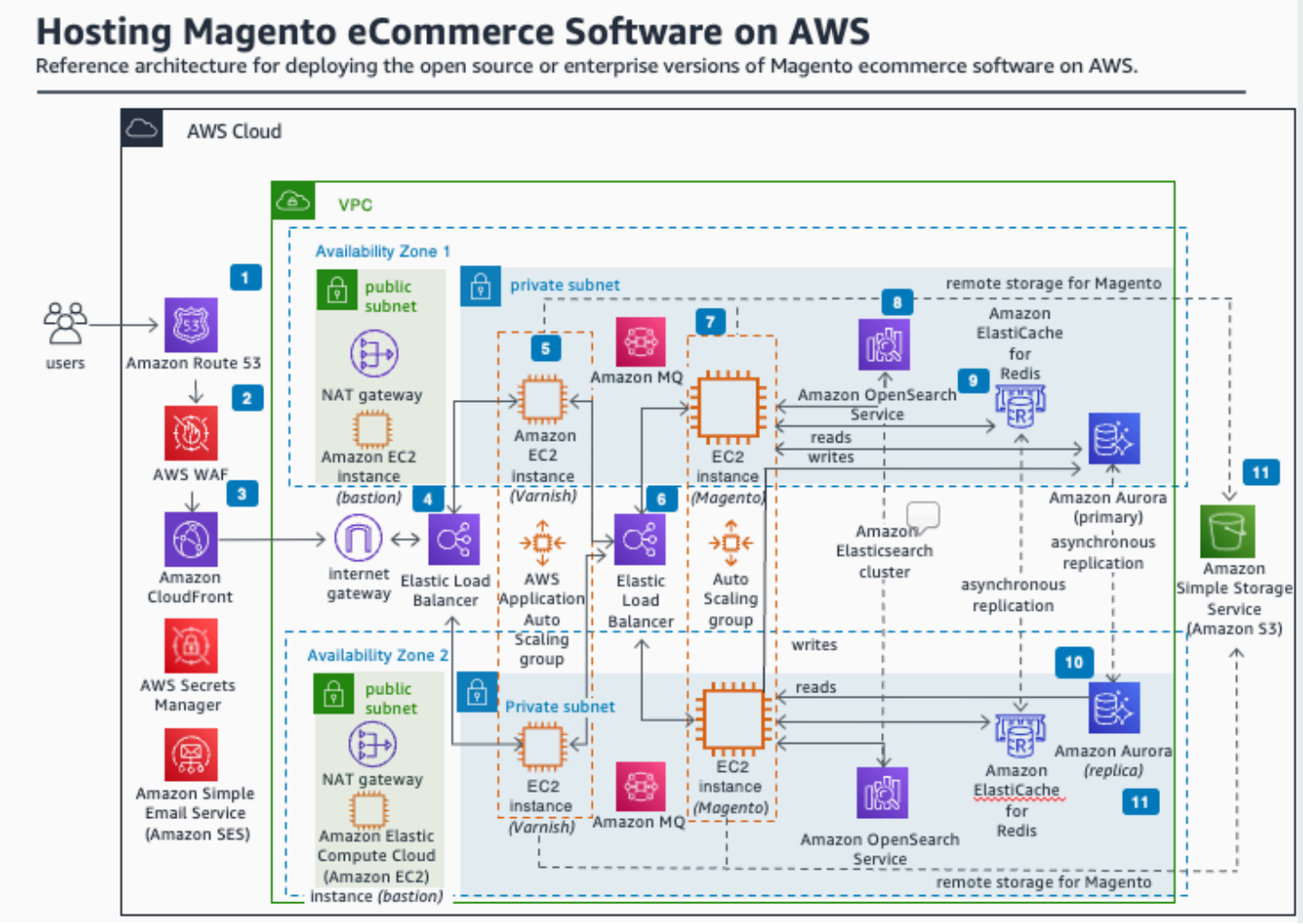

Figure 1 – Reference architecture for Magento Open Source or Adobe Commerce on Cloud Infrastructure Self-Service

Deployment options

There are several deployment options available for running Magento (both Magento Open Source or Adobe Commerce on Cloud Infrastructure Self-Service) on AWS. The most appropriate choice depends on your requirements for cost, scale, availability, and flexibility as well as the AWS and Magento skills of your organization. This section outlines the full spectrum of deployment options along with some key characteristics to keep in mind when evaluating each option.

Table 1 – Summary of Adobe Commerce Deployment Options on AWS

Deployment Style	Option	Description
Self-managed	Amazon Lightsail	Easy, cost effective start. Best for small business.
Self-managed	AWS Marketplace	AMI based solutions from variety of providers. Integrate other AWS services. Customer responsible for Adobe Commerce patching/updates and configuring network connections.
Self-managed	AWS Quick Start for Magento	Deploys Magento open-source reference architecture per AWS best practices in minutes. Highly configurable. Customer responsible for server and Magento maintenance.
Managed hosting	AWS Consulting partners	Easy deployments, including Magento open-source or Adobe Commerce on cloud infrastructure self-service store front customizations. Partner responsib

Deployment Style	Option	Description
		le for infrastructure and Magento maintenance. Best for organizations with less evolved IT departments or lacking skills/people to build/customize Magento deployments
Managed hosting	Magento Commerce Cloud by Adobe	Most popular, highly scalable hosting option with Magento Commerce application and infrastructure managed by Adobe and store front customizations managed by merchant.

Self-managed

AWS provides three options to quickly get started with a self-managed deployment of Magento. By self-managed we mean that AWS or an AWS partner provides the scripting or Amazon Machine Image (AMI) to deploy Magento and the necessary infrastructure dependencies, such as EC2 instances, in your AWS account. Once deployed, you are responsible for managing the deployment going forward including monitoring, patching, and upgrading.

Amazon Lightsail

Amazon Lightsail provides virtual servers, storage, databases, and networking that are easy-to-use and designed to allow you to quickly get started with the cloud. Popular application stacks, or blueprints, are available that can be deployed on Lightsail virtual servers. These stacks come preconfigured with all of the necessary components to get started with an application in minutes.

The Magento application stack is provided by Bitnami and includes Apache, Varnish, Memcached, MySQL, and Magento Open Source bundled in an AMI.

From the Lightsail console in your AWS account, you simply select the AWS Region and availability zone where you want to launch Magento, choose an instance plan, and launch your instance. After

a few minutes, your instance is deployed and ready to access. Although Magento and all of its dependencies come preconfigured, you can still securely access the instance using a browser-based SSH interface or your favorite SSH client to make lower level changes. The web-based Magento Administration user interface can also be used to create and customize your stores.

With the simplicity and cost effectiveness of deploying Magento Open Source through Lightsail also comes some important tradeoffs. These tradeoffs are important to keep in mind when it comes to scalability, availability, and maintenance of your e-commerce site. First, the Magento opensource AMI provided by Bitnami that is used by Lightsail installs Adobe Commerce and all dependencies on a single instance. Although this keeps the deployment simple, it also limits the ability to scale your e-commerce site, creates multiple single points of failure (SPOF), and leaves you with the responsibility to patch and update dependencies such as Memcached and MySQL. Therefore, selecting Lightsail as a deployment option should only be considered for smaller e-commerce sites where you expect a consistently low level of traffic from visitors.

AWS Marketplace

The AWS Marketplace is an e-commerce site where AWS customers can discover, procure, and deploy solutions provided by AWS partners. Thousands of solutions are available across more than 1,000 categories including infrastructure, business applications, machine learning, and many others. Deployment options supported on the Marketplace include applications deployed directly into customer accounts via AMIs or Docker containers, Amazon SageMaker, or SaaS solutions. All software purchased through the Marketplace appears on the customer's AWS invoice along with any other AWS resources consumed.

Magento deployment options currently available in the AWS Marketplace are AMI-based. These offerings include Magento opensource and all the necessary dependencies such as MySQL, a web-server, and caching components. Therefore, AWS customers can deploy Magento open-source or Adobe Commerce on cloud infrastructure self-service directly into their AWS account and get it up and running within minutes.

Similar to the Lightsail option, the customer is responsible for patching and upgrading Magento and its dependencies. In addition, the customer is responsible for configuring the networking environment, or Amazon Virtual Private Cloud (Amazon VPC), within which Magento Open Source or Adobe Commerce on Cloud Infrastructure Self-Service is deployed. Lastly, customers should closely investigate and understand the scalability and high availability characteristics of each option, the Magento Open Source version bundled with each option, and any included customizations such as enhanced caching or multi-store setups. For example, some are all-in-one

bundles that are intended to be deployed on a single Amazon EC2 instance. Although this provides a simpler configuration and lower cost, it introduces several single-points-of-failure and lacks the ability to take advantage of the cloud's elasticity and high availability capabilities.

AWS Quick Start for Magento Open Source

Magento is an open-source content management system for e-commerce websites. AWS enables you to set up the infrastructure to support deployment in a flexible, scalable, and cost-effective manner in the AWS Cloud. This reference deployment will help you rapidly build a Magento Open Source cluster by automating configuration and deployment tasks.

The automated deployment builds a cluster that runs Magento version 2.4.3 or higher along with optional sample data.

This guide covers the deployment of Magento Open Source in the AWS Cloud. It doesn't provide Magento product usage information. For general guidance and best practices for using Magento, see the Magento User Guide on the Adobe website.

Cost and licenses

This deployment launches Magento Open Source automatically into a configuration of your choice. You are responsible for the cost of the AWS services used while running this Quick Start reference deployment. There is no additional cost for using the Quick Start. The cost will vary depending on the storage and compute configuration of the cluster you deploy. See the pricing pages for each AWS service you will be using for full details.

This Quick Start uses Magento Open Source (formerly Community Edition), which is open-source software distributed under the Open Software License (OSL 3.0).

AWS components for Magento Open Source

Running this Quick Start with default parameters for a new VPC deploys and configures a VPC that spans two Availability Zones. Each Availability Zone is configured with a private and a public subnet. This Quick Start deploys the following AWS components in the AWS Cloud:

- In a public subnet, a bastion host provides Secure Shell (SSH) access to the Magento web servers. The bastion host is maintained by an Auto Scaling group that spans multiple Availability Zones, and is configured to ensure there is always one bastion host available.

- AWS-managed network address translation (NAT) gateways deployed into the public subnets and configured with an Elastic IP address for outbound internet connectivity. NAT gateways are used for internet access for all EC2 instances launched within the private network.
- Auto Scaling is enabled to automatically increase capacity if there is a demand spike, and to reduce capacity during low traffic times. The default installation sets up low and high CPU-based thresholds for scaling the instance capacity up or down. You can modify these thresholds during launch and after deployment.
- An IAM instance role with fine-grained permissions for access to AWS services necessary for the deployment process.
- Appropriate security groups for each instance or function to restrict access to only necessary protocols and ports. For example, access to HTTP server ports on Amazon EC2 web servers is limited to Elastic Load Balancing. The security groups also restrict access to Amazon RDS DB instances by web server instances.

The core AWS components used by this Quick Start include the following AWS services.

- Amazon EC2 – The Amazon Elastic Compute Cloud (Amazon EC2) service enables you to launch virtual machine instances with a variety of operating systems. You can choose from existing Amazon Machine Images (AMIs) or import your own virtual machine images.
- Amazon VPC – The Amazon Virtual Private Cloud (Amazon VPC) service lets you provision a private, isolated section of the AWS Cloud where you can launch AWS services and other resources in a virtual network that you define. You have complete control over your virtual networking environment, including selection of your own IP address range, subnet creation, and configuration of route tables and network gateways.
- Terraform modules on AWS – You can now use Terraform modules on Amazon Web Services (AWS) to deploy native Terraform resources on the AWS Cloud. Terraform modules on AWS are published under an open-source license.

Terraform modules on AWS are available in the Terraform registry on the AWS Integration and Automation namespace page. Use the links provided to access modules in the Terraform registry and source code on GitHub. For module deployment instructions, refer to the `README.md` file in the GitHub repository.

- Auto Scaling – Auto Scaling helps maintain high availability and manage capacity by automatically increasing or decreasing the EC2 instance fleet. You can use Auto Scaling to

run your fleet at optimal utilization by increasing instance capacity during demand spikes and decreasing capacity during down times.

- Elastic Load Balancing – Elastic Load Balancing automatically distributes incoming application traffic across multiple EC2 instances.
- Amazon CloudFront - Amazon CloudFront is a web service that speeds up distribution of your static and dynamic web content, such as `.html`, `.css`, `.js`, and image files, to your users. CloudFront delivers your content through a worldwide network of data centers called edge locations.
- The Amazon Simple Storage Service (Amazon S3) – Amazon S3 is an object storage service that offers industry-leading scalability, data availability, security, and performance. This Magento deployment uses Amazon S3 as Remote Storage Module for Magento to provide the option to store media files and schedule imports/exports in a persistent, remote storage container.

> **Note**
>
> Magento highly discourages the use of public buckets due to high security risks.

- Amazon OpenSearch Service — OpenSearch has quickly become the most popular search engine and is commonly used for log analytics, full-text search, security intelligence, business analytics, and operational intelligence use cases. As of version 2.4, Magento requires OpenSearch to be the catalog search engine. Magento supports using OpenSearch provided by Amazon Web Services (AWS).
- Amazon ElastiCache – Amazon ElastiCache service makes it easy to deploy, operate, and scale an in-memory data store or cache in the cloud. The service improves the performance of web applications by allowing you to retrieve information from fast, managed, in-memory data stores, instead of relying entirely on slower disk-based databases.
- Amazon MQ — Amazon MQ is a managed message broker service that makes it easy to set up and operate message brokers in the cloud. The MQF uses RabbitMQ as the messaging broker, which provides a scalable platform for sending and receiving messages. It also includes a mechanism for storing undelivered messages. RabbitMQ are primarily needed for B2B and async operations like import, export, or Bulk operations.
- Amazon RDS – Amazon Relational Database Service (Amazon RDS) makes it easy to set up, operate, and scale a growing set of relational databases, including MySQL and Amazon Aurora,

both of which are supported by the Magento Quick Start. With Amazon RDS, you can deploy scalable relational databases in minutes with cost-efficient and resizable hardware capacity.

> **Note**
>
> The split database feature was deprecated in version 2.4.2 of Magento. See Revert from a split database to a single database.

- IAM – AWS Identity and Access Management (IAM) enables you to securely control access to AWS services and resources for your users. With IAM, you can manage users, security credentials such as access keys, and permissions that control which AWS resources users can access, from a central location.

Architecture

This Quick Start provides two deployment options. Depending upon which option you choose, it creates and configures the necessary AWS components in the AWS Cloud.

Deploying this Quick Start **with default parameters for end-to-end deployment** (which creates a new VPC) builds the following Magento environment in the AWS Cloud.

Architecture flow of AWS components

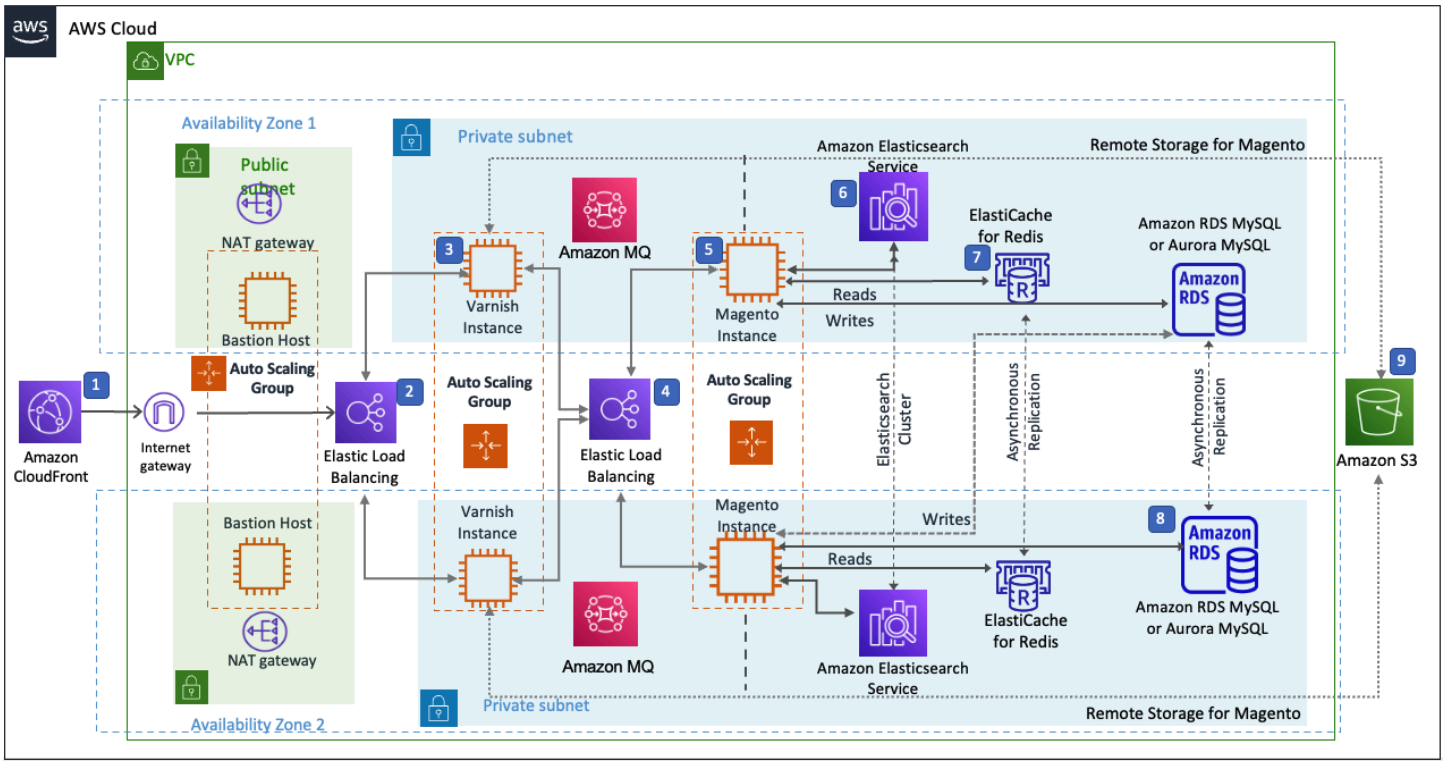

Figure 2 – Reference architecture deployed by AWS Quick Start for Magento Open Source

1. Amazon CloudFront is deployed as a content delivery network (CDN). CloudFront speeds up distribution of static and dynamic web content.
2. First Elastic Load Balancing (Application Load Balancer) distributes traffic across Varnish instances in an AWS Auto Scaling group in multiple Availability Zones.
3. Varnish deployed on Amazon EC2, Varnish Cache is a web application accelerator caching HTTP reverse proxy. Balancer distributes traffic from Varnish Cache across the AWS Auto Scaling group of Magento instances in multiple Availability Zones.
4. Second Elastic Load Balancing (Application Load Balancer) distributes traffic from Varnish Cache across the AWS Auto Scaling group of Magento instances in multiple Availability Zones.
5. Magento web server on Amazon EC2 instances launched in the private subnets.
6. Amazon OpenSearch Service for Magento catalog search.
7. An Amazon ElastiCache cluster with the Redis cache engine launched in the private subnets.
8. Either an Amazon RDS for MySQL or an Amazon Aurora database engine deployed via Amazon RDS in the first private subnet. If you choose Multi-AZ deployment, a synchronously replicated

secondary database is deployed in the second private subnet. This provides high availability and built-in automated failover from the primary database.

9. Amazon S3 created as remote storage for web server instances to store shared media files.
10. Amazon MQ (optional) is a message broker that offers a reliable, highly available, scalable, and portable messaging system. The Message Queue Framework (MQF) is a system that allows a module to publish messages to queues for Magento flow. It also defines the consumers that will receive the messages asynchronously. Bulk operations are actions that are performed on a large scale. Example bulk operations tasks include importing or exporting items, changing prices on a mass scale, and assigning products to a warehouse. For each individual task of a bulk operation, the system creates a message that is published in a message queue and processed by background consumer runs.

Magento Open Source components

This Quick Start deploys Magento Open Source (2.4.3) with the following prerequisite software:

- Operating system: Amazon Linux x86-64 or Debian
- Web server: NGINX
- Database: Amazon RDS for MySQL 5.6 or Amazon Aurora 5.7
- Programming language: PHP 7.4, including the required extensions
- Message broker: Amazon 3.8.11
- Database Cache: Amazon ElastiCache Redis 6.x
- Page Cache: Varnish 6.5
- Content Catalog Search: Amazon OpenSearch Service 7.10

Deployment steps

The procedure for deploying a Magento cluster on AWS consists of the following steps. For detailed instructions, follow the links for each step.

Step 1. Prepare an AWS account

1. If you don't already have an AWS account, create one at https://aws.amazon.com by following the on-screen instructions. Part of the sign-up process involves receiving a phone call and entering a PIN using the phone keypad.

2. Use the Region selector in the navigation bar to choose the AWS Region where you want to deploy the Magento cluster on AWS. For more information, see Regions and Availability Zones. Regions are dispersed and located in separate geographic areas. Each Region includes at least two Availability Zones that are isolated from one another but connected through low-latency links.

> **Important**
>
> This Quick Start uses Amazon Aurora, which might not be available in all AWS Regions. Before you launch this Quick Start, check the Region table for availability.

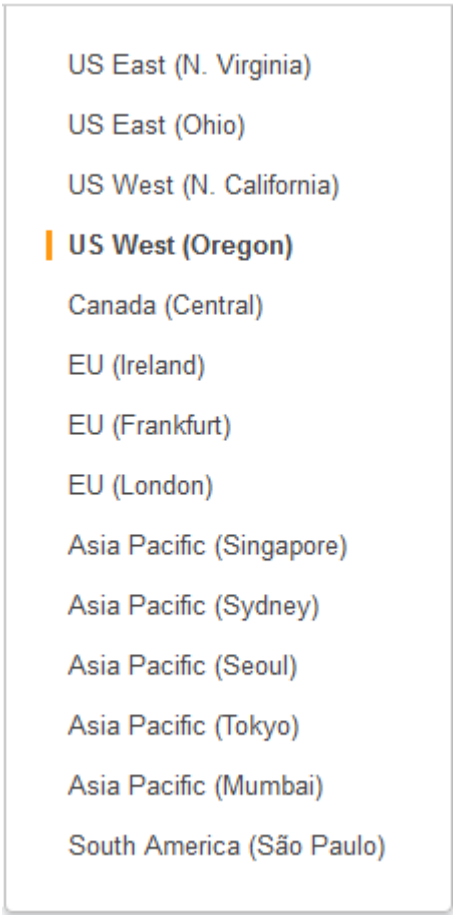

Figure 3 – Choosing an AWS Region

> **Tip**
>
> Consider choosing a Region closest to your data center or corporate network to reduce network latency between systems running on AWS and the systems and users on your corporate network.

3. Create a key pair in your preferred Region. In the navigation pane of the Amazon EC2 console, choose **Key Pairs**, **Create Key Pair**, type a name, and then choose **Create**.

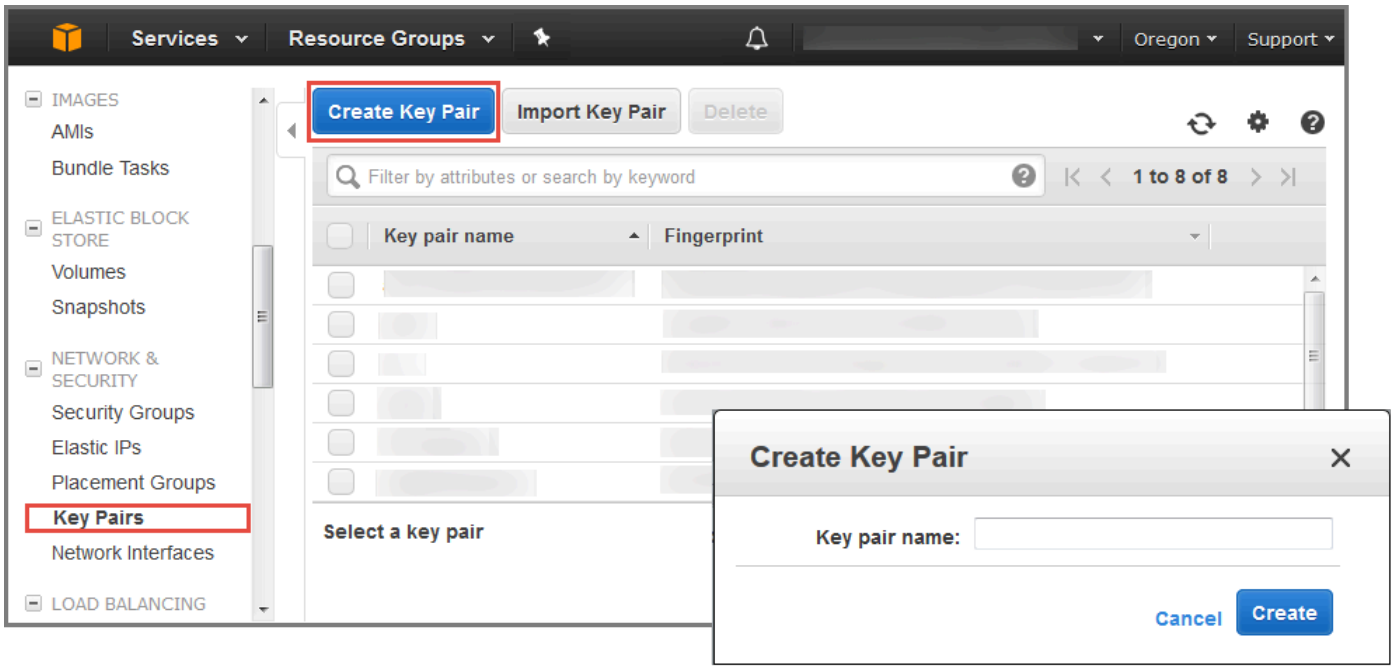

Figure 4 – Creating a key pair

Amazon EC2 uses public-key cryptography to encrypt and decrypt login information. To be able to log into your instances, you must create a key pair. On Linux, we use the key pair to authenticate SSH login.

For this deployment, store the private key you created in the previous step in Secrets Manager using the AWS Management Console as plaintext.

a. In the AWS Management Console, navigate to AWS Secrets Manager, choose your AWS Region, and choose **Store a new secret**.

b. Select **Other type of secrets** and choose **Plaintext**.

c. Clear the `{"":""}` JSON format from the **Plaintext** section.

d. Copy and paste your private key.

Figure 5 – Copying key

e. Keep the `DefaultEncryptionKey` to encrypt your SSH Key secret. Click **Next**.

f. Set the secret name as "**ssh-key-admin**"

g. Click **Next**. Leave the automatic rotation to disabled. Select **Next**.

h. Review and select **Store**.

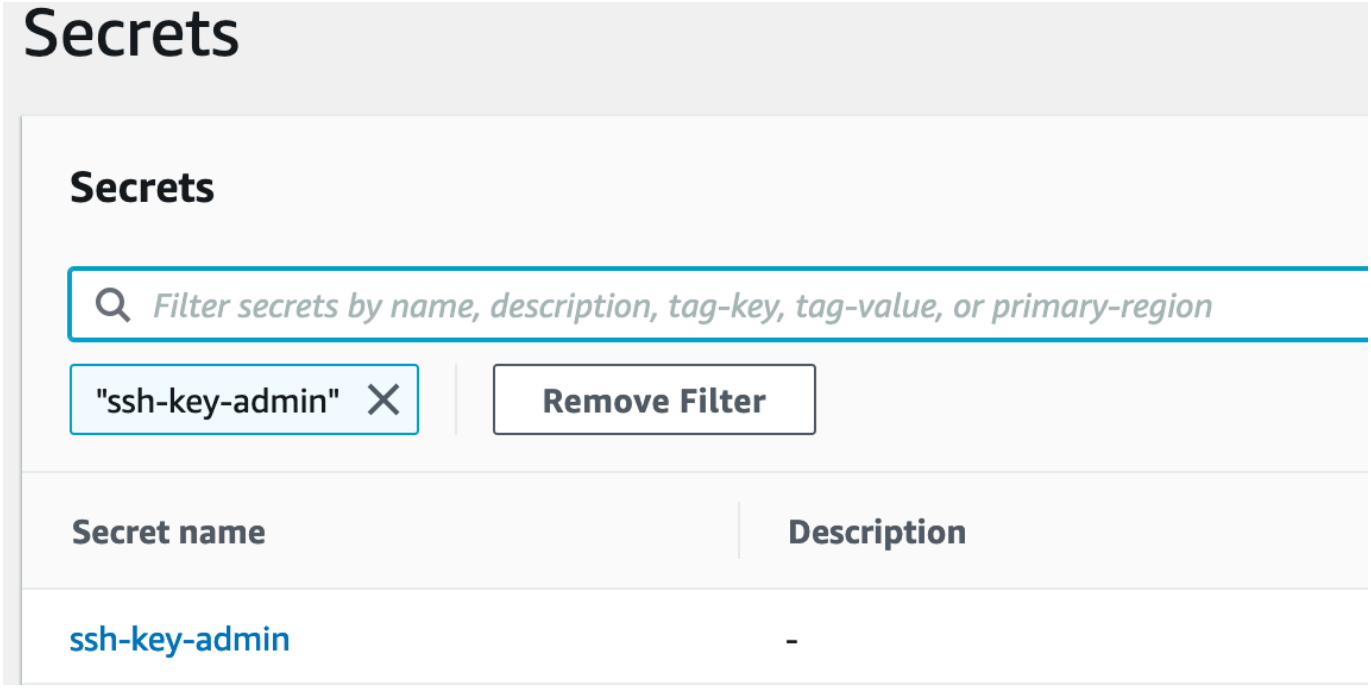

Figure 6 – Reviewing secrets

4. If necessary, request a service quota increase for the instance types used for the deployment. You might need to request an increase if you need additional Elastic IP addresses or if you already have an existing deployment that uses the same instance types as this architecture. On the Service Quotas console, for each instance type that you want a service quota increase, choose the instance type, choose **Request quota increase**, and then complete the fields in the quota increase form. It can take a few days for the new service quota to become effective.

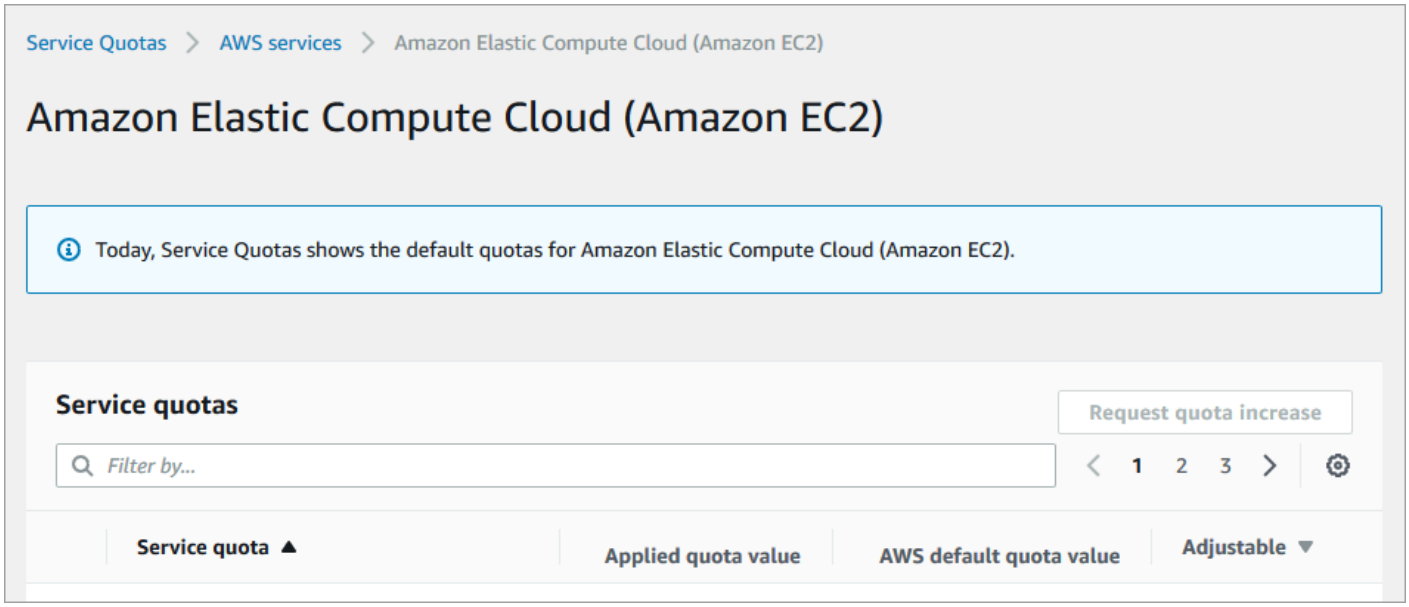

Figure 7 – Requesting a service quota increase

Step 2. Create Magento keys for deployment

This deployment use Magento Composer to manage Magento components and their dependencies. To learn more about Magneto Composer, see the Adobe documentation.

1. Create a Magento public authentication key for Composer Username.
2. Create a Magento private authentication key for Composer Password.

For detailed instructions on creating keys, see the Adobe documentation.

Figure 8 – Creating a Magento access key

Step 3. Set up Terraform and a Terraform Cloud account

1. Install Terraform. For installation steps, see the Terraform documentation.
2. Set up a Terraform Cloud account. For setup instructions, see the Terraform Cloud documentation (There is a free tier available.)
3. Create a workspace in Terraform to organize infrastructure. For setup instructions, see the TerraForm Cloud workspace documentation.

Step 4. Prepare local environment with Terraform setup

Generate a Terraform Cloud token:

```
terraform login
```

Export the `TERRAFORM_CONFIG` variable:

```
export TERRAFORM_CONFIG="$HOME/.terraform.d/credentials.tfrc.json"
```

Configure the `tfvars` file

Create `terraform.tfvars` in the following path:

```
$HOME/.aws/terraform.tfvars
```

An example of the `tfvars` file contents:

```
AWS_ACCESS_KEY_ID = "{insert access key ID}"
```

```
AWS_SECRET_ACCESS_KEY = "{insert secret access key}"
AWS_SESSION_TOKEN = "{insert session token}"
```

Note

We recommend using AWS Security Token Service (AWS STS)–based credentials.

Warning

Follow best practices for managing secrets, and ensure that your credentials are not stored in a public repository.

Note

Before deployment, you must create both an AWS key pair and a Magento deployment key.

Deploy the module (Linux and iOS)

1. Clone the repository from GitHub.
2. Navigate to the repository's root directory.
3. Navigate to the `setup_workspace` directory:

```
cd setup_workspace
```

4. Run the following commands in order:

```
terraform init
```

Alternatively, for the previous command, specify the file:

```
terraform apply -var-file="$HOME/.aws/terraform.tfvars"
```

5. You are asked for the following:
   a. The AWS Region where you want to deploy this module. This must match the Region where you generated the key pair.

b. The organization under which Terraform Cloud runs. This can be found in the Terraform Cloud console.

c. Setup confirmation.

> **Note**
>
> Terraform Cloud creates the workspace, which contains the Terraform Cloud organization name.

6. Navigate to the directory, and deploy Magento (the previous `terraform init` command generates `backend.hcl`):

```
cd ../deploy
```

- Open, edit, and review all of the variables in the `variables.tf` file.
- Update the `default=` value for your deployment.
- The `description=` value provides additional context for each variable.

The following items must be edited before deployment:

- Project-specific: `domain_name`
- Magento information: `mage_composer_username`
- Magento information: `mage_composer_password`
- Magento information: `magento_admin_password`
- Magento information: `magento_admin_email`
- Database: `magento_database_password`
- Variable `base_ami_os`: Use `amazon_linux_2` or `Debian_10`.
- Variable `use_aurora`: If you are using Amazon RDS for MySQL instead of Amazon Aurora, change to `false`.

> **Important**
>
> Don't store secret information in a public repository.

7. After you review, update and save the `./deploy/variables.tf` file, see the Deployment section.

Run the following commands from an IDE or terminal with Terraform installed.

1. Initialize the environment:

```
terraform init
```

1. Verify that the deployed architecture is correct:

```
terraform plan
```

2. Validate the code:

```
terraform validate
```

3. Deploy the infrastructure run one of the following commands:

```
terraform apply
```

or

```
terraform apply -var-file="$HOME/.aws/terraform.tfvars"
```

Post-deployment steps

`terraform apply` runs remotely in Terraform Cloud and takes about 30–60 minutes to deploy.

During the deployment, you should receive an AWS email to allow Amazon SES to send you emails. Verify this before you log into Magento.

After the Terraform deployment completes, an output shows the relevant information for accessing Magento.

> **Important**
>
> After Terraform completes, Magento bootstraps the environment, which takes about 15–20 minutes. Various Magento install and configuration commands run during this time, and

the site enters maintenance mode. After it exits maintenance mode, images sync with your Amazon Simple Storage Service (Amazon S3) bucket.

Test the deployment

After Terraform completes, it outputs the frontend and backend URLs. Use the credentials specified in the `variables.tf` file to log in as an administrator. Run the following command to connect to the web node:

```
ssh -i PATH_TO_GENERATED_KEY -J admin@BASTION_PUBLIC_IP admin@WEB_NODE_PRIVATE_IP
```

Note

Ensure that you have SSH key forwarding enabled.

Clean up the infrastructure

Note

If you want to retain the Magento files stored in your S3 bucket, copy and save the bucket's objects before completing this step.

When you no longer need the infrastructure, run one of the following commands to remove it:

```
terraform destroy
```

or

```
terraform destroy -var-file="$HOME/.aws/terraform.tfvars"
```

After you remove the infrastructure, the database is stored as an artifact.

Troubleshooting

Terraform can sometimes timeout when interacting with the AWS API. It is usually best to do a terraform destroy and then do terraform apply when encountering these errors.

For troubleshooting common Quick Start issues visit the AWS Quick Start General Content Guide or the Troubleshooting CloudFormation page in the AWS documentation.

After you successfully deploy a Quick Start, confirm that your resources and services are updated and configured — including any required patches — to meet your security and other needs. For more information, see the AWS Shared Responsibility Model.

Going beyond infrastructure with Shero Commerce

Shero Commerce is an e-commerce agency based in NY with offices around the world. Their team of skilled specialists and certified Magento developers create highly integrated e-commerce stores designed for increased conversions, faster site speed, enhanced customer experience, better SEO, and long-term growth.

Once you have deployed Magento open-source or Adobe Commerce on cloud infrastructure self-service and have your ecommerce site up and running, the challenge of getting customers to visit your site, converting those customers to buyers, and optimizing the cost of your system are all things Shero can help with.

Shero helps with SEO to drive traffic to your site, and helps with the user experience (UX) and user interface (UI) of your site to help turn more customers into buyers. One example is the outdoor knife and cutlery retailer Benchmade, who approached Shero with a goal of maximizing U.S. revenue and scaling their Magento store into the international market, all while maintaining brand integrity.

Shero assisted Benchmade with a redesigned home page, upgraded product listings and search capability, a product customizer that allows customers to view multiple versions of the same product, and more. The site relaunched in time to see holiday weekend sales jump more than 100% over the previous year.

For customers interested in additional modernization of their ecommerce platform, Shero has worked with online sellers to integrate their Magento open-source or Adobe Commerce on cloud infrastructure self-service store with Amazon Alexa, called Voice Commerce. This integration allows customers to use their voice to interact with the seller's storefront to make purchases or get recommendations. Learn more about voice-enabled purchasing for Magento open-source or Adobe Commerce on cloud infrastructure self-service environments in this APN TV video.

Adapting to the ever-changing world of e-commerce, Shero continues to expand their offerings with innovative opportunities like this Magento Quick Start. To learn more or to contact their team, visit SheroCommerce.com.

Managed hosting

For customers who are not comfortable or do not have the resources to manage their own deployment of Magento Open Source or Adobe Commerce on Cloud Infrastructure Self-Service on AWS, there are several companies that specialize in providing managed hosting deployments of Magento Open Source or Adobe Commerce on Cloud Infrastructure Self-Service on AWS. These companies, such as CloudHesive, take care of the aspects of deploying, securing, patching, and maintaining Magento. Some also provide design services and custom development for Magento storefronts. You can use AWS Partner Finder to find and compare providers that specialize Magento Open Source or Adobe Commerce on Cloud Infrastructure Self-Service hosting.

Magento Commerce Cloud by Adobe

One of the most popular managed hosting options for Magento is offered by Adobe itself called Adobe Commerce (also known as Magento Commerce Cloud). It's a fully managed automated hosting platform that comes with a variety of additional deployment and development features in addition to the self-hosted Magento on cloud and Magento Open Source platforms.

Adobe Commerce pre-provisioned infrastructure includes PHP, MySQL, Redis, RabbitMQ, and OpenSearch technologies. In addition, it provides a Git-based workflow with automatic build and deploy for efficient rapid development and continuous deployment every time you push code changes in a platform as a service (PaaS) environment. Magento Commerce Cloud has AWS hosting that offers a scalable and secure environment for online sales and retailing.

Adobe Commerce Cloud extensibility strategy

Adobe Commerce on cloud infrastructure self-service or Adobe Commerce Cloud provides native support for Adobe Developer App Builder, Adobe's cloud-native serverless extensibility platform. With Adobe Developer App Builder, business can build apps that extend the Storefront and admin UI, build middleware app integrations using Rest and GraphQL API, all in a no/low code environment. App Builder manages provisioning of storage, CDN, compute, and security along with a Developer Console and command line interface for centralized developer experience across Adobe solutions. App Builder also includes an API Mesh that enables customers to easily configure and integrate third-party APIs and run GraphQL queries across multiple sources of data through a single mesh. Every Magento Commerce customer is entitled to a base level of App Builder capacity that covers most common extensibility use cases, with the ability to purchase additional capacity if needed. To learn more about the integrations capabilities, see Magento Commerce documentation. To learn more about the trial sign up process, check out the trial sign-up page

Security and compliance

Despite the common misconception that a cloud environment is less secure than on-premises infrastructure, strategic goals in relation to security and compliance are often key drivers for organizations to migrate to the cloud. Leading hyperscale cloud service providers such as AWS invest heavily in security and compliance and deliver a better security profile than what the biggest and most conservative organizations can deliver internally.

Security is a top priority at AWS. As an AWS customer, regardless of your size or investment, you inherit all the benefits of AWS experience, tested against the strictest of third-party assurance frameworks.

AWS Shared Responsibility Model

Under the AWS Shared Responsibility Model, AWS provides a global secure infrastructure and foundation for compute, storage, networking and database services, as well as higher level services. AWS provides a range of security services and features that AWS customers can use to secure their assets. AWS customers are responsible for protecting the confidentiality, integrity, and availability of their data in the cloud, and for meeting specific business requirements for information protection. In a simple way, AWS is responsible for security *of* the cloud and the customer is responsible for security *in* the cloud.

When leveraging the AWS Cloud, customers can choose a security solution that is designed to protect their organization's content, platform, applications, systems and networks, while also meeting their business needs. AWS offers a wide range of tools and features that help organizations increase privacy and control network access so they can more easily meet their needs within the AWS Shared Responsibility Model. Amazon Virtual Private Cloud (Amazon VPC) enables you to create a logically isolated portion of the AWS Cloud, from which you can launch Amazon EC2 instances in a virtual network that you define. Security groups allow you to define a virtual firewall around your EC2 instances, which contains rules that control the inbound and outbound traffic to your instances. Network access control lists (ACLs) provide an optional layer that allows you to control traffic in and out of one or more subnets in your VPC.

Best practices

AWS and Magento open-source or Adobe Commerce on cloud infrastructure self-service offer a range of tools to help secure your cloud resources and to help you meet your compliance needs under organizational and open standards. Best practices include the following:

Network security

Amazon VPC allows you to create private networks within AWS and control network access to your instances and subnets. Use private or dedicated connectivity options such as AWS Direct Connect to connect your on-premises office or datacenter to AWS. If you are already using AWS in your organization then use AWS Private Link to connect Magento Open Source or Adobe Commerce on Cloud Infrastructure Self-Service hosted cloud to your existing VPC. Lastly, always include a web application firewall (AWS WAF) and distributed denial of service (DDoS) mitigation technologies (AWS Shield) as part of your automatic scaling or content delivery strategy.

Data encryption

Always encrypt your data both at rest and in transit. You can use TLS to encrypt the data in transit. Encryption at rest is achieved via data encryption capabilities available in AWS storage services such as Amazon Elastic Block Store (Amazon EBS), Amazon Simple Storage Service (Amazon S3), and Amazon Relational Database Service (Amazon RDS). Also, there are dedicated hardware-based cryptographic key storage options available for customers to help satisfy their compliance requirements.

Access control

Protect your AWS and Magento Open Source or Adobe Commerce on Cloud Infrastructure Self-Service user credentials. AWS credentials are used to access AWS services where Magento Open Source or Adobe Commerce on Cloud Infrastructure Self-Service credentials are used to manage your storefront within Magento. Use appropriate permissions across user accounts that access AWS resources and user accounts that access Magento store front customizing capabilities. AWS Identity and Access Management (IAM) enables you to create multiple users and manage the permissions. AWS supplies two types of security credentials: AWS access keys and X.509 certificates. Access keys and certificates for authentication to AWS services. As a good practice, it is recommended that you incorporate a key rotation mechanism into your application architecture. Lastly for extra security, AWS recommends that you use multifactor authentication (MFA) for all user accounts, including options for hardware-based authenticators, and integrate with federated identity providers such

as on-premised corporate directories to reduce administrative overhead and improve end- user experience.

If users already have identities (user credentials) outside of AWS, such as in a corporate directory, then you can use identity federation along with AWS IAM Identity Center to simplify the user management process. You can use the identity information from the external corporate directory system and use appropriate roles inside IAM for managing permissions.

AWS Secrets Manager can be used to rotate, manage, and retrieve database credentials, API keys, as well as Magento open-source or Adobe Commerce on cloud infrastructure self-service secrets (user names and passwords). You can configure Secrets Manager to rotate secrets automatically, which can help you meet your security and compliance needs. Secrets Manager offers built-in integrations for Amazon Aurora Amazon RDS and can rotate credentials for these databases natively. To retrieve secrets for Magento open-source or Adobe Commerce on cloud infrastructure self-service application, you can replace plaintext secrets with a call to Secrets Manager APIs, eliminating the need to hardcode secrets in source code or update configuration files and redeploy code when secrets are rotated.

Monitoring and logging

Always have the right monitoring and logging tools enabled to give you the visibility you need to spot issues before they impact your business. AWS features a variety of services that give you deep visibility (who, what, when, and from where) such as 1) AWS CloudTrail for API calls (access requests), 2) AWS Config (configuration history), 3) Amazon CloudWatch to gain system-wide visibility into resource utilization, application performance, and operational health and 4) VPC flow logs that log all network traffic flowing through the VPC. In addition, you can configure CloudWatch Events to automatically alert as well as respond to adverse events. Lastly, Amazon GuardDuty is a threat detection service that automates continuous monitoring for malicious activity and unauthorized behavior using machine learning.

Security guidance

AWS provides customers with guidance and expertise through online tools, resources, support, and professional services provided by AWS and its partners. AWS Trusted Advisor is an online tool that inspects your AWS environment to help close security gaps, and finds opportunities to save money, improve system performance, and increase reliability. AWS Advisories and Security Bulletins provide advisories around current vulnerabilities and threats, and enable customers to work with AWS security experts to address concerns like reporting abuse, vulnerabilities, and penetration

testing. Magento security center provides advisories around latest Magento Open Source or Adobe Commerce on Cloud Infrastructure Self-Service patches and security updates. Magento security scan tool, a free tool from Adobe, can be used to monitor your websites for security risks, update malware patches, and detect unauthorized access.

PCI DSS

AWS supports a variety of security standards and compliance certifications including the Payment Card Industry Data Security Standard (PCI DSS), which is a crucial requirement for organizations who process credit cards and store cardholder information. Organizations that fail to comply with PCI requirements can expect large fines, which can also result in canceling their ability to process payments. PCI compliance requires organizations to safeguard their customers' payment card information following security requirements that include policies and procedures, software design, and network architecture.

AWS is certified as a PCI DSS 3.2 Level 1 Service Provider, the highest level of assessment available. Magento Commerce is PCI certified as a Level 1 Solution Provider. Organizations can use AWS and Magento's PCI Attestation of Compliance to aid their own PCI certification process.

The PCI DSS certification for an organization involves attestation of the following 12 requirements, broken into 6 groups: 1) Build and Maintain a Secure Network, 2) Protect Cardholder Data 3) Maintain a Vulnerability Management Program, 4) Implement Strong Access Control Measures, 5) Implement Strong Access Control Measures and 6) Regularly Monitor and Test Networks. For more information on PCI Compliance, refer to PCI DSS resources on AWS website and PCI Security Standards Council website.

AWS architecture components

As you plan your migration from an on-premises environment to AWS, you might find yourself introduced to new concepts. The first of those is the concept of managed services, which, simply defined, is a specific capability delivered as a service. For example, Amazon Elastic File System (Amazon EFS) provides network file system (NFS) based storage, eliminating the need to otherwise manage a fleet of EC2 instances to accomplish the same.

Another might be the scaling of shared services – so whereas before your on-premises environment may have consisted of a cluster of load balancing appliances, AWS offers (in a similar fashion as Amazon EFS described above) Elastic Load Balancing as a service, allowing you to manage the components used to deliver Magento Open Source or Adobe Commerce on Cloud Infrastructure Self-Service in an end-to-end fashion.

Yet another concept might be the physical architecture of AWS – where a singular AWS Region comprises numerous groups of physically isolated datacenters (each group of data centers is referred to as an Availability Zone), providing native support for not just N+1 or active/passive or active/active resiliency, but doing so in physically distributed manner without implementing complex solutions.

In line with these previously described services, where an on-premises implementation of Magento Open Source or Adobe Commerce on Cloud Infrastructure Self-Service might look like shared domain, networking, and load balancing services with an external content delivery network, all running on one or more servers (physical or virtual), in AWS, you may find yourself using any number of services to accomplish the same outcome from a service delivery perspective, while reducing cost and providing increased automation capabilities and increased scalability and resiliency capabilities.

Types of services

Continuing the concepts previously described, a typical collection of AWS services supporting your Magento Open Source or Adobe Commerce on Cloud Infrastructure Self-Service environment, assuming no external or on-premises services are leveraged, might look like this:

- Amazon Route 53 for DNS
- Amazon CloudFront for content delivery network
- Amazon S3 as remote storage for web server instances to store shared media files.

- Elastic Loading Balancing through an Application Load Balancer for load balancing
- AWS Auto Scaling Groups of Amazon EC2 (virtual machine) instances for the Magento open-source or Adobe Commerce on cloud infrastructure self-service execution environment
- Amazon Relational Database Service (Amazon RDS) for the Magento open-source or Adobe Commerce on cloud infrastructure self-service database environment
- Amazon ElastiCache (Redis OSS) for session and configuration caching
- Amazon Simple Email Service (Amazon SES) as your mail gateway
- Amazon MQ (optional) is a message broker that offers a reliable, highly available, scalable, and portable messaging system.

The above list, while detailed with regard to the Magento Open Source or Adobe Commerce on Cloud Infrastructure Self-Service environment itself, excludes numerous services providing supporting infrastructure of your platform (such as Amazon Virtual Private Cloud [Amazon VPC]) and as such this list should not be treated as a definitive list of services or requirements. Its intention is to illustrate the baseline AWS services you can use to host your Magento Open Source or Adobe Commerce on Cloud Infrastructure Self-Service and eliminate unnecessary care and feeding of servers.

Monitoring

An additional benefit to leveraging managed services is that each service provides observability via metrics (CloudWatch Metrics), logs (CloudWatch Logs) and events (CloudWatch Events) providing the often-sought single pane of glass around each of your Magento Open Source or Adobe Commerce on Cloud Infrastructure Self-Service environments. This data can include performance, exceptions (both technical measures) as well as business measures such as cost per transaction (via the CloudWatch API). This robust set of data, in addition to being available historically for reporting and near real time for dashboarding is also available for alerting – of both individuals in your organization and automations for self-healing processes.

From an organizational operations perspective, each of the preceding services also support the application of arbitrary metadata (referred to as *tags*), which can be used to allocate costs (cost per environment, as an example), in conjunction with AWS configuration management service, AWS Config, as well as an AWS robust audit trail solution, AWS CloudTrail, which provides insight into user or service access and change to AWS services.

Finally, services such as Application Load Balancer integrate into AWS X-Ray, which is the AWS distributed application tracing solution which provides for additional instrumentation into the

runtime environment of Magento Open Source or Adobe Commerce on Cloud Infrastructure Self-Service and associated code.

DevOps

Access to on-demand compute and managed services, along with a common set of operational capabilities (authentication, authorization and auditing via AWS Identity and Access Management and CloudTrail APIs, and SDKs available on a variety of platforms and common metric, log, and event ingestion services) support's an organization's goal towards automation.

AWS provides numerous services that abstract procedural level automation of resource lifecycle management including AWS CloudFormation, AWS Elastic Beanstalk, AWS OpsWorks, Amazon Elastic Container Service (Amazon ECS), and Amazon Elastic Kubernetes Service (Amazon EKS), in addition to a rich ecosystem of third-party services such as Terraform, Ansible and others.

There are multiple approaches with regards to managing a code base running in a compute environment. One such approach is using Amazon Machine Images (AMIs) where a machine image is programmatically configured at launch and cycled through with each code revision. Another approach is to statically maintain and configure managed machines leveraging agent or agent-based code deployment solutions. Lastly, container-based code deployments where a container is moved through different environments instead of code.

Each of these have their own strengths and weaknesses, however, container-based code deployments are becoming more common and support additional compute services such as AWS Fargate, a serverless container solution.

Regardless of the approach, the primary objective should be to maintain the compute environments in a hands-off approach with best security practices and monitoring solutions that provide good operational support.

For Magento, the management of the application lifecycle involves many of the same concepts, considered best practices and leveraged by other development platforms such as the use of a code repository and versioning (Git), separate build environments (for code compilation) and separate runtime environments (for example, development, and production). In addition, Magento Open Source or Adobe Commerce on Cloud Infrastructure Self-Service supports the management of stores, cron entries, and so on, via the command line. This management logic can be built into the deployment pipeline. Also the configuration files can be versioned similar to code. Magento recommends management of assets via the same code repository as the application and configuration files, but this may create additional complexity, depending on the size of your

application's assets. As an alternative approach, you can use one-way synchronization between environment specific Amazon Elastic File System (Amazon EFS) volumes.

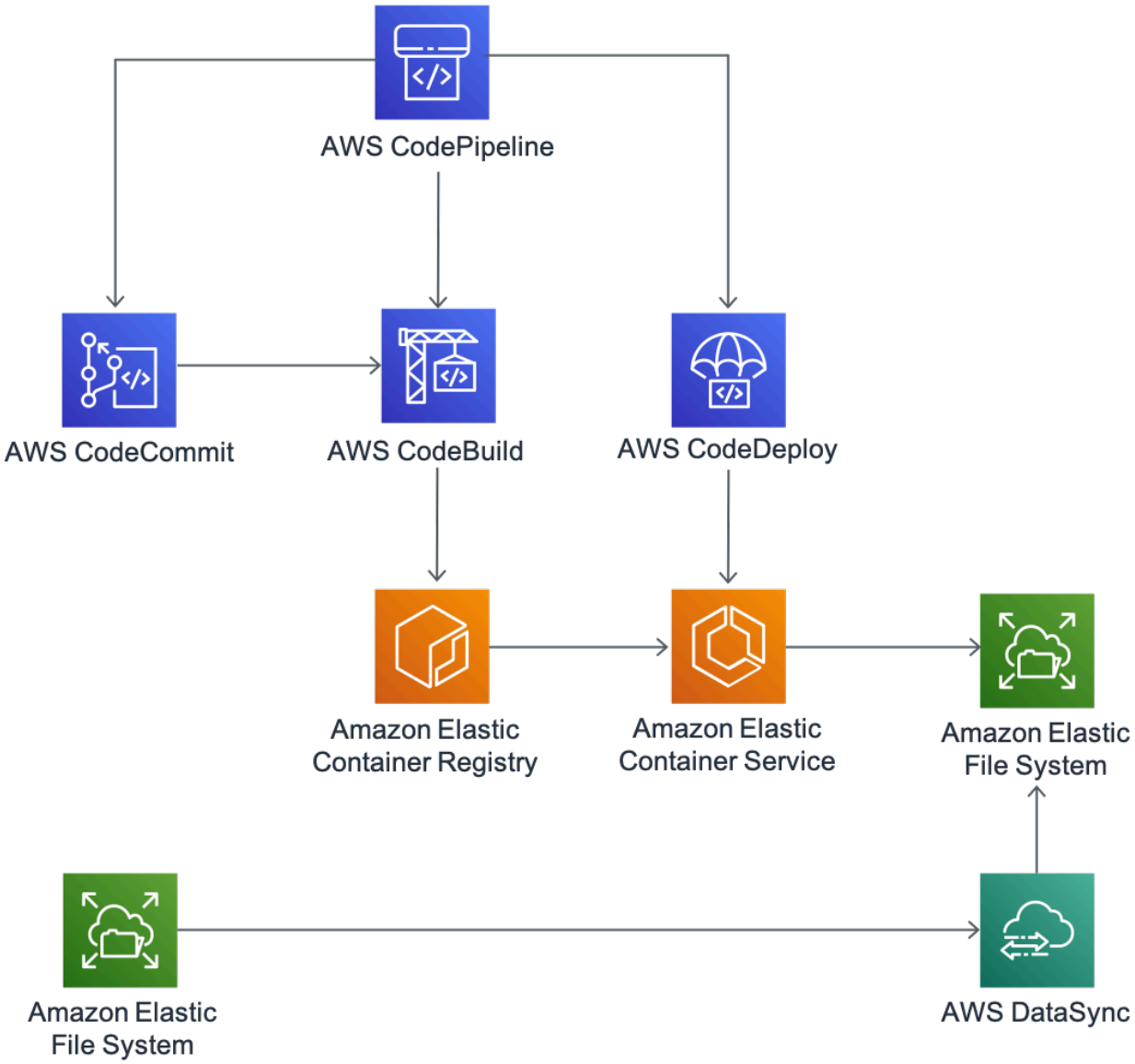

Figure 9 – Reference architecture for DevOps using containers

Connectivity

The migration of Magento Open Source or Adobe Commerce on Cloud Infrastructure Self-Service might be part of an enterprise cloud adoption strategy, driven by a compelling event or a combination of the two. Regardless of the reasons, make sure to consider systems dependent on Magento Open Source or Adobe Commerce on Cloud Infrastructure Self-Service and vice versa. Connectivity between the Magento Open Source or Adobe Commerce on Cloud Infrastructure Self-Service systems running in AWS and these dependent systems (whether in AWS, a data center, or a SaaS/PaaS provider) is key to planning and successfully running a migration. Make sure to also consider connectivity for the Magento administrator, system administrator, database administrator, and infrastructure administrator.

Types of connectivity

Five connectivity channels are available in AWS:

- API (serverless resources such as Amazon S3, Amazon DynamoDB, etc.)
- Internet (publicly routed, typically brokered via SFTP based automation)
- VPN (IPSEC/B2B)
- AWS Direct Connect (Dedicated connectivity, via either a physical or logical connection)
- Intra VPC (typically environment:tier <-> environment:tier),
- Inter VPC (via Peering or Transit Gateway) and AWS PrivateLink (typically private connectivity, AWS based SaaS offerings).

Whether your connectivity is between tiers, environments, other services within your enterprise on and off AWS or with an AWS Partner, each of these approaches has advantages and disadvantages associated with them.

Network dependencies

Of most importance, however, is fully understanding the network dependency map that exists within your Magento Open Source or Adobe Commerce on Cloud Infrastructure Self-Service environment ahead of migration. Having a dependency map can help better plan the steps required in the migration and avoid accelerated post migration changes required to account for missed dependencies, both building to support a successful migration.

Investigation around this might begin with reviewing load balancer and firewall configurations as they relate to hostnames and IP addresses of the servers Magento Open Source or Adobe Commerce on Cloud Infrastructure Self-Service is running on, its related service configurations and any associated logs – particularly load balancer, firewall, and application performance management (APM) logs which may provide hints around third-party services not otherwise discoverable.

Service congruency

For example, a typical Magento Open Source or Adobe Commerce on Cloud Infrastructure Self-Service environment is likely accessed from the internet via a load balancing appliance or a firewall providing NAT public IP address space, and may have integrations to other internet-based services (for example, a web service offered by a payment processor or shipping company) as well as legacy integrations via SFTP or a similar internet-based file sharing service.

These legacy integrations described earlier would translate into AWS services in the way of an Application Load Balancer on one set of subnets protected via a security group and web application firewall, to a fleet of EC2 instances running Magento Open Source or Adobe Commerce on Cloud Infrastructure Self-Service dependent services. A NAT gateway providing outbound access to internet-based services. Same or separate fleet of EC2 instances for providing integration with other internet services such as payment processors, or leverage serverless solutions such AWS lambda and AWS Transfer for SFTP for Amazon S3 based on the use-case.

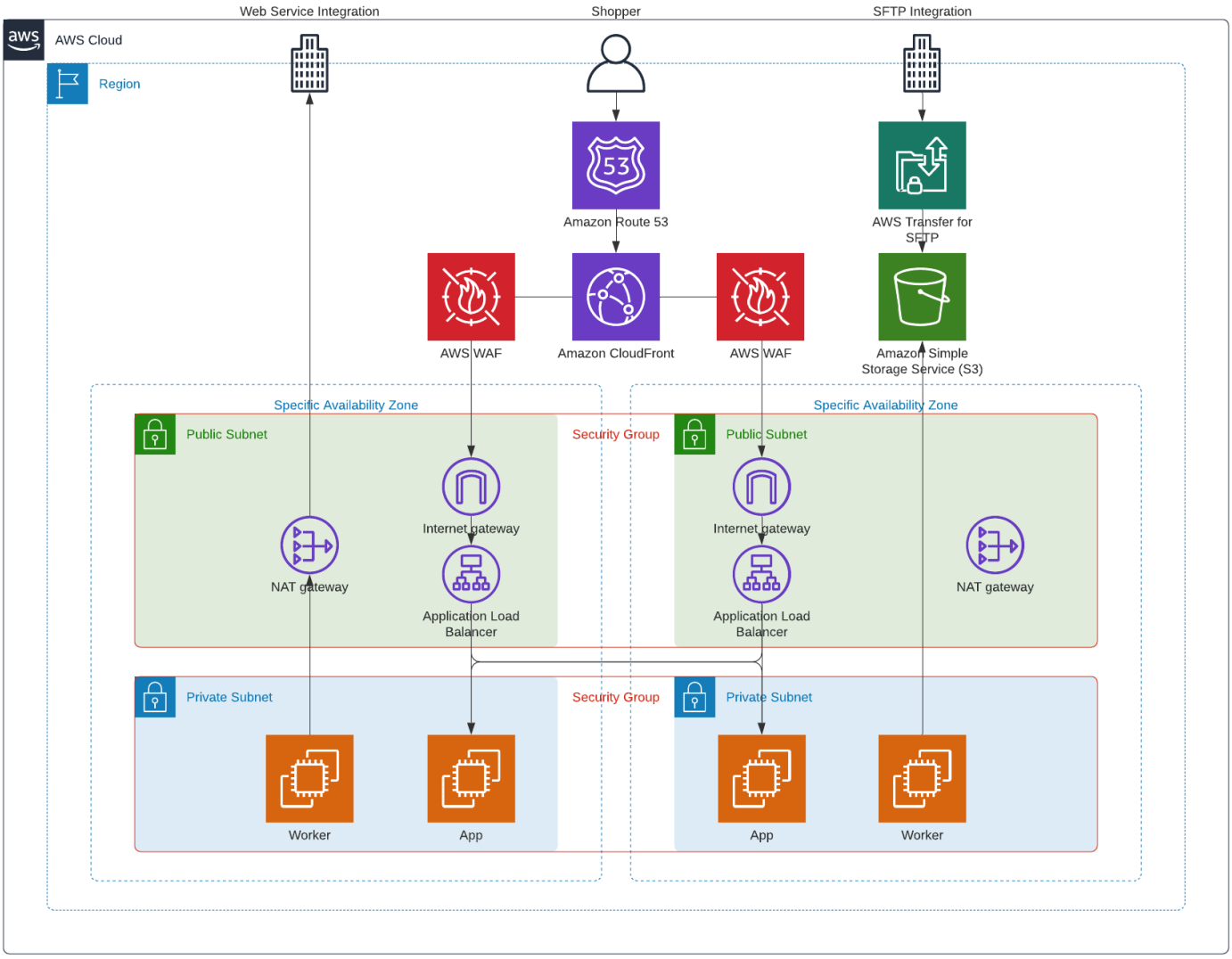

Figure 10 – Reference architecture showing service congruencies for containers

Migration plan

This section steps through an example migration, broken into four phases.

Planning

Planning in migration involves identifying goals, scope and business requirements. Often overlooked, and related to these items, are having quantifiable goals as measures of success, specifically with regard to performance (for example, performance shall remain the same or be improved by *x* margin) as well as a full understanding of the components that comprise the to-be-migrated environments.

To solve for the first, you can use the existing monitoring used in your enterprise or use AWS monitoring capabilities. Rather than focusing on machine-level statistics (such as CPU, memory, storage consumption, storage IO, and network consumption), which are important with regard to machine configuration, focus on the end-user experience, for example, response times for key activities, such as adding to cart, checking out, and payment.

To solve for the second, you can use existing financial records, inventories, or monitoring to identify the components that comprise each environment, including those that might not be evident, for example, virtualized solutions, or those delivered as a third-party solution, such as DNS or CDN).

Staging

After you have identified measures of success and completed the discovery, you should have a sense of the baseline services required to support your application. Compared to your existing AWS footprint (if there is one) you may opt to create an entirely new AWS account or Amazon VPC, use an existing AWS account or Amazon VPC, or go through the collective supporting characteristics for first time AWS adoption found in the AWS Landing Zone Solution.

Once this work is complete, you can begin prototyping your environment on AWS – familiarizing yourself with some of the services mentioned above and identifying potential replacements to share service, appliance or server based services. This is the re-platforming approach.

Once this is setup, you can begin to stage an environment for your first test migration. This environment would essentially replicate your current production environment from a software,

configuration, code, and management perspective and aim to test both basic functionality of the platform as well as your data backup/restore and/or data replication processes that will be leveraged during the actual migration.

This would be your second decision point for rehosting versus re-platforming: there are multiple approaches you can take to migrating your servers to AWS – from as basic as exporting a virtual machine and importing it into AWS, to using CloudEndure Migration for real time replication of the machine. While opinions vary on the subject, the reader is strongly encouraged to take an approach biased to re-platforming as it presents an opportunity to significantly improve operations of the platform with minimal (but not nonexistent) up front work.

This first attempt at the migration provides you with a harness to execute, record, measure and modify the previously defined plans and tests, understand baseline performance as well as capture step-timings and improve the overall cutover plan. This step can be repeated as often or as frequently as needed until a comfort level is obtained to move into the next step.

Cutover

The final step of the migration process is the live migration from the current on-premises environment to the AWS based environment. Essentially the process is similar as the previous step, though you will likely place the store in maintenance mode and/or use a static maintenance site to prevent live transactions during the most critical parts of the cutover.

An important part of this step is to have agreement with all participating parties around the maximum time to be spent on each step as well as critical milestone steps having pass/fail criteria along with rollback criteria in the event of a failure. Having these steps defined ahead of time avoid last minute decisions based on arbitrary criteria from being made in the event an unanticipated challenge is met (game days, tabletop exercises or simulations attempt to drive a similar thought process).

At this point you've effectively migrated your Magento Open Source or Adobe Commerce on Cloud Infrastructure Self-Service platform to AWS, and you can begin the decommissioning process – first through soft methods (e.g. stopping services) on to harder methods (decommissioning of virtual machines and servers).

The cutover process varies based on the specific Magento Open Source or Adobe Commerce on Cloud Infrastructure Self-Service installations architecture, but in general, you want to stop existing Magento processes as well as disable any scheduled `cron` jobs.

You can redirect traffic by changing the current content delivery network (CDN) configuration, the current DNS configuration (ensuring as part of premigration you've lowered the TTL) or, optionally, using the EC2 instances on AWS as new upstream targets for your current load balancer (this being the least preferred option).

Some customers might use this as an opportunity to change DNS or CDN providers and these migration activities can be performed as part of a pre-migration or post-migration, following their own migration process and steps.

Optimization

Previously made architecture decisions can also be revisited post-migration. For example, if you used a rehost approach, you can evaluate and re-platform post-migration services, or right-size services based on current and forecasted load. This period of time is an excellent period to evaluate potential opportunities for reserved instance purchases, which offer a discount to service cost in trade for commitment to use a service for a period of time (ideal for Amazon RDS, Amazon ElastiCache and permanent Amazon EC2 instances in production).

Other areas of consideration

Magento Open Source or Adobe Commerce on Cloud Infrastructure Self-Service M1 to M2 Migration Magento 1.x versions (M1) to Magento 2.x versions (M2) migration consists of four main components:

- Data migration
- Extensions migration
- Custom code migration
- Theme migrations

A deep dive into these four areas specific to how the organization has setup M1 outlines the challenges and the migration strategy. Magento Open Source or Adobe Commerce on Cloud Infrastructure Self-Service has developed the Data Migration Tool for data migration and Code Migration Toolkit for code migration. These resources are described in detail in the Magento Migration Guide.

Magento hardware sizing guidelines on AWS

As per Magento hardware recommendations, Adobe recommends that one CPU core can effectively serve around two (up to four) Magento requests along with one `cron` process simultaneously. Determine your organization's stable expected request rate to find the appropriate number of cores needed to support your application and use automatic scaling to dynamically extend web tier nodes as needed.

```
N[Cores] = (N[Expected Requests] / 2) + N [Expected Cron Processes]
```

In addition, Adobe recommends at least 2 GB memory on build servers and 1 GB on web nodes along with sufficient network bandwidth to prevent bottlenecks on read-write operations.

Keeping these principles in mind, choose the appropriate instance from the Amazon EC2 instance types that balances your organization's cost and business needs. Instance types from the Amazon EC2 general purpose family (especially M types) provide a good balance between compute, memory, and network resources for Magento Open Source or Adobe Commerce on Cloud Infrastructure Self-Service applications.

Backups and disaster recovery (DR)

As a best practice, explore backup and disaster recovery (DR) options based on your business needs. Backup and disaster recovery options depend on the deployment option and database choices (Amazon RDS vs Amazon Aurora) you choose. Your organization's business needs dictate the recovery point objective (RPO) (that is, the maximum time to last backup) and the recovery time objective (RTO). RTO often varies depending upon the size of your organization's storage.

At a minimum, we recommend that you take regular database backups and have a working copy of the AWS CloudFormation template to provision the infrastructure when needed in case a recovery situation arises. Alternately, you can choose to have a working Amazon Machine Image (AMI) copy that has Magento Open Source or Adobe Commerce on Cloud Infrastructure Self-Service installed along with your organization's latest changes or customizations. You can use this AMI to provision infrastructure using an AWS CloudFormation template to reduce the overall RTO.

Conclusion

This paper presented the business drivers for migrating Magento Open Source or Adobe Commerce on Cloud Infrastructure Self-Service to the AWS Cloud along with the strategies and considerations. Migrating Magento Open Source or Adobe Commerce on Cloud Infrastructure Self-Service eCommerce software on AWS provides a secure and scalable foundation for delivering great digital experiences for customers. As you prepare for your migration to AWS, we recommend that you consider the guidance outlined in this document and consult the additional references provided in Further reading.

Contributors

Contributors to this document include:

- Kenney Rajan Sr. Solutions Architect, Amazon Web Services
- Vikram Mehto Sr. Solutions Architect, Amazon Web Services
- Rachael Barnett Sr. PDM Amazon Web Services
- Anuj Ratra, Sr. Solutions Architect, Amazon Web Services
- James Jory, Sr. Solutions Architect, Amazon Web Services
- Patrick Hannah, Chief Technology Officer, CloudHesive

Further reading

For additional information, see:

- AWS Partner Finder
- Magento Developer Documentation
- Magento AWS Quick Start
- AWS Digital Customer Experience Competency & Partners
- AWS Webinar: How to Manage Organizational Change and Cultural Impact During a Cloud Transformation
- Magento Hardware Sizing
- Magento Commerce Configuration Best Practices
- Migrating to AWS: Best Practices and Strategies

Document history

To be notified about updates to this whitepaper, subscribe to the RSS feed.

Change	Description	Date
Whitepaper updated	Updated to reflect Adobe Commerce and latest best practices. Title of whitepaper changed to reflect new content.	July 19, 2023
Initial publication	Whitepaper first published.	April 21, 2020

Note

To subscribe to RSS updates, you must have an RSS plug-in enabled for the browser that you are using.

Notices

Customers are responsible for making their own independent assessment of the information in this document. This document: (a) is for informational purposes only, (b) represents current AWS product offerings and practices, which are subject to change without notice, and (c) does not create any commitments or assurances from AWS and its affiliates, suppliers or licensors. AWS products or services are provided "as is" without warranties, representations, or conditions of any kind, whether express or implied. 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.

AWS Glossary

For the latest AWS terminology, see the AWS glossary in the *AWS Glossary Reference*.