

HCL Commerce

on Amazon Web Services

Deployment Guide

Version 1.2

Table of Contents

1.	<i>Introduction</i>	5
1.1.	HCL Commerce Use Cases / Overview (INT-001)	5
1.2.	Typical Deployment Overview (INT-002)	5
1.3.	All Deployment Options Discussed in Guide (INT-003)	8
1.4.	Expected Time to Complete Deployment (INT-004)	8
1.5.	Regions Supported (INT-005)	8
2.	<i>Prerequisites and Requirements</i>	9
2.1.	AWS Specific Requirements (PRQ-001)	9
2.2.	HCL Commerce Software Requirements (PRQ-001)	10
2.3.	Required Skills or Specialized Knowledge (PRQ-002)	11
2.4.	Environment Configuration for Deployment (PRQ-003)	11
3.	<i>Architecture Diagrams</i>	12
3.1.	Standard HCL Commerce on AWS Deployment (ARCH-001)	12
3.2.	Customer Data Diagram(s) (ARCH-003 / DSEC-008)	13
3.3.	Network Diagram(s) (ARCH-005 / DSEC-010)	14
3.4.	Integration Point Diagram(s) (ARCH-006)	14
4.	<i>Security</i>	14
4.1.	IAM Best Practices (DSEC-001 / DSEC-002 / DSEC-003)	14
4.2.	Public Resources (DSEC-004)	15
4.3.	IAM Roles and Policies (DSEC-005)	15
4.4.	User Created Keys for Deployment (DSEC-006)	17
4.5.	Secrets Storage (DSEC-007)	18
4.6.	Customer Sensitive Data Storage (DSEC-008)	19
4.7.	Data Encryption Configuration(s) (DSEC-009)	19
5.	<i>Costs</i>	20
5.1.	Billable Services (Mandatory/Optional) (CST-001)	20
5.2.	Cost Model / Licensing Costs (CST-002)	20
6.	<i>Sizing</i>	21
6.1.	EC2 Instance Type and Size (SIZ-001)	21
6.2.	EBS Volume Type and Size (SIZ-002)	22
6.3.	Instance Size Selection for Managed AWS Services (SIZ-003)	23
6.4.	Amazon Dynamo DB Read/Write Capacity (SIZ-004)	23
7.	<i>Deployment Assets</i>	23
7.1.	Deploying workload on AWS as per typical deployment architecture (DAS-001)	23
7.2.	Maximizing Uptime & Availability (DAS-002)	49
7.3.	Different Deployment Configurations (DAS-003)	49
7.4.	Testing / Troubleshooting (DAS-004)	49
8.	<i>Health Check</i>	50
8.1.	Assess & Monitor Application Health (HLCH-001)	50
9.	<i>Backup and Recovery</i>	51
9.1.	Automated Backup of Necessary Components (BAR-001)	51
9.2.	Restoring Data from a Backup (BAR-002)	51
9.3.	Recovery from Failure (BAR-003)	51
9.4.	Recovery from Availability Zone Failure (BAR-004)	51

9.5.	Manage Service Limits for Disaster Recovery (BAR-005).....	52
9.6.	Recovery Time Objective (RTO) and Recovery Point Objective (RPO) (BAR-006).....	52
10.	<i>Routine Maintenance</i>	52
10.1.	Rotating Programmatic Credentials and Crypto Keys (RM-001)	52
10.2.	Software Patches and Upgrades (RM-002)	53
10.3.	Managing Licenses (RM-003)	53
10.4.	Managing AWS Service Limits	53
11.	<i>Emergency Maintenance</i>	54
11.1.	Handling Fault Conditions (EMER-001)	54
11.2.	Recovering the Software (EMER-002).....	54
12.	<i>Support</i>	54
12.1.	How to Receive Support (SUP-001)	54
12.2.	Technical Support Tiers (SUP-002).....	54
12.3.	Support Tiers & SLAs (SUP-003)	55

Revision History

Version	Author	Date	Comments
1.0	Commerce Team	13JUN2022	Reorganize Sections/Headers For Commerce Team Effort
1.1	Commerce Team	13JUL2022	Most current edits/updates included
1.2	Commerce Team	30AUG2022	Most current edits/update included

1. Introduction

1.1. HCL Commerce Use Cases / Overview (INT-001)

HCL Commerce provides a powerful customer interaction platform for omni-channel commerce. It can be used by companies of all sizes, from small businesses, to large enterprises, and for many different industries. It provides easy-to-use tools for business users to centrally manage a cross-channel strategy. Business users can create and manage precision marketing campaigns, promotions, catalog, and merchandising across all sales channels, or use integrated AI enabled content management capabilities.

HCL Commerce is a single, unified platform that offers the ability to do business directly with consumers (B2C), directly with businesses (B2B). HCL Commerce is a customizable, scalable, and high availability solution that is built to use open standards. It uses cloud friendly technology to make deployment and operation easy and efficient.

1.2. Typical Deployment Overview (INT-002)

For the intended use of the software, see [1.1. HCL Commerce Use Cases](#).

Commerce applications

The specific HCL Commerce application deployment topology depends on the Search solution and Store solution combination that is used. Deployments that are based on previous versions of HCL Commerce can be migrated and remain fully supported by HCL Commerce 9.1. These migrations from older implementations also have an impact on the server topology.

A new implementation of HCL Commerce 9.1 deployment is intended to utilize a separate Store Server to host the storefront, and the newer, more capable search solution based on Elasticsearch.

Some containers that are listed contain applications that are used sparingly, or not at all, and are only operated temporarily to aid in starting, modifying, diagnosing, or monitoring the HCL Commerce deployment. These include the Utility server, the Support container, the Cache Manager, and the Must-Gather application.

Resource	Description	Docker Name
Transaction Server	The HCL Commerce server running on WebSphere Application Server. Provides the transactional business logic and APIs which serve other servers of the HCL Commerce application.	ts-app
Transaction Web Server	The web server that provides access to the HCL Commerce tooling and other services.	ts-web

Search Server	If deploying with SOLR as search engine, this is the SOLR Search server.	search-server
Store Server	Serves store assets including JSP files, e-Marketing Spots and images.	crs-app
Tooling Web	An IBM HTTP Server which hosts and serves angular based tooling, a single page application (SPA) and the associated front-end assets that are displayed within Management Center for HCL Commerce.	tooling-web
Store Web	An IBM HTTP Server which hosts and serves the React-based headless stores (Emerald for B2C and Sapphire for B2B).	store-web
Elasticsearch Query	If deploying with Elasticsearch, this provides the query service application that provides the APIs for Product, Category as well as configuration endpoints for Search profiles, Matchmakers and NLP NER.	search-query-app
Elasticsearch Ingest	If deploying with Elasticsearch, this provides the ingest service that provides the APIs for accessing and managing Connectors deployed on the Apache NiFi application cluster.	search-ingest-app
Elasticsearch NiFi Server	<p>If deploying with Elasticsearch, this provides the third-party application, Apache NiFi.</p> <p>Apache NiFi is designed to automate the flow of data between systems. HCL Commerce Search uses Apache NiFi as the data ingestion pipeline for all Stores and Product Catalog business data.</p>	search-nifi-app
Elasticsearch NiFi Registry	<p>If deploying with Elasticsearch, this provides the third-party application, Apache NiFi Registry.</p> <p>It is a complimentary application that provides a central location for</p>	search-registry-app

	<p>versioning components assembling the NiFi pipeline.</p> <p>NiFi Registry acts as an internal intermediary flow repository between NiFi and the permanent source of control repository and is used for coordinating flow updates in NiFi.</p>	
Customization Server	A server which runs custom code that you can create to extend HCL provided xC extension points.	xc-app
Utility Server	Contains a utility project that holds scripts to service HCL Commerce operations such as loading access control policies, loading store/catalog data, and cleaning the database of obsolete objects.	ts-utils
Support Container	<p>Contains the deployment and operation python scripts for use in the deployment of HCL Commerce within a Kubernetes cluster.</p> <p>Primarily used for service dependency checks to ensure that the various Commerce applications are brought online properly and in the expected order.</p> <p>Also used by some utility jobs such as for TLS certificate generation for secure ingress.</p>	supportcontainer
Cache Manager	Contains the REST interfaces to operate and monitor the Remote HCL Cache stored in Redis.	cache-app
Must-Gather	Contains the Must-Gather application which aids in the collection and aggregation of HCL Commerce troubleshooting information.	

Required third-party applications

Resource	Description	Docker Name
Elastic Search	Used for on-site search, including indexing and ingestion of product and other data and query services on the storefronts.	
Zookeeper	Publish and subscribe to streams of records.	
Redis	Handles cache invalidation between the Transaction server and the other servers in the HCL Commerce platform.	

For more information, see the [HCL Commerce production environment overview](#), in the HCL Commerce documentation.

1.3. All Deployment Options Discussed in Guide (INT-003)

HCL Commerce can be deployed in a multi-availability zone (AZ) deployment model. For the purposes of this document, a single availability zone is used.

1.4. Expected Time to Complete Deployment (INT-004)

The required infrastructure (such as the domain, obtaining SSL certificates, cluster configuration, creation and initialization of the database, etc.) must be completed before deploying the solution.

Depending on the skill level and the required environment, the expected time can vary, from several hours to a whole day for a new customer. An experienced party who understands the manual process can follow the guidance that is provided in this document to prepare several cloud formation scripts to automate the deployment process. With the help of cloud formation templates and automation, the entire deployment process can potentially be completed within one or two hours.

1.5. Regions Supported (INT-005)

HCL Commerce can be deployed in any Amazon Web Services (AWS) region, provided that the region has access to Amazon Elastic Kubernetes Services (EKS) clusters and stand-alone Amazon Elastic Compute Cloud (EC2) instances.

2. Prerequisites and Requirements

2.1. AWS Specific Requirements (PRQ-001)

The following AWS services are required to deploy HCL Commerce on AWS.

Service	Description
Virtual Private Cloud (VPC)	Allows for the creation of a private network in AWS.
Elastic Compute Cloud (EC2)	Virtual Machine (VM) instances that are required to Support HCL Commerce.
Elastic Kubernetes Service	Used to run and scale HCL Commerce on Kubernetes.
Elastic Container Registry	Used to store, share, and deploy HCL Commerce container images.
Elastic Load Balancing	Used to distribute incoming application traffic across multiple targets.
AWS Cloud Storage	Used to provide the various storage requirements for HCL Commerce.
Route 53	Routing DNS requests.

The following AWS services are optional to deploy HCL Commerce on AWS.

Service	Description
AWS Cloud Formation	Infrastructure as a code service that allow you to easily model, provision, and manage AWS and third-party resources.
AWS Cloud Watch	Used for monitoring various services.
Elastic Container Service	Container management service that allows you to easily run applications on a managed cluster of Amazon Elastic Compute Cloud instances.

2.2. HCL Commerce Software Requirements (PRQ-001)

The following applications are required to deploy HCL Commerce on AWS.

Application	Description
HCL Commerce 9.1 Docker Images	The core platform docker images which can be obtained from HCL Software and stored in your Docker registry of choice.
Kubernetes 1.16 or later	Used for automation, scaling, and deployment management.
Helm 3.3.1 or newer	Used for Kubernetes deployment automation.
Docker 19.03.8 or newer	Docker Container runtime.
Vault 1.9.1	Storing of sensitive data for the application.
Supported databases <ul style="list-style-type: none">• Oracle 18c• Oracle 19c (if running Elasticsearch-based search solution)• IBM DB2 11.5	HCL Commerce platform data resides in this database.
Database VM <ul style="list-style-type: none">• CentOS• RHEL	Used to install the chosen database platform.
Open-source components for Elasticsearch deployment	Elasticsearch, Redis, and Zookeeper.

2.3. Required Skills or Specialized Knowledge (PRQ-002)

The following table lists the skills and knowledge that is required to deploy and operate HCL Commerce on AWS.

Skill	Description
AWS Solutions Architect	To create the deployment architecture following AWS best practices for security and networking.
AWS Engineer	To deploy all required resources as per design and administration.
Kubernetes Administrator	Manage all aspects of EKS.
HCL Commerce Administrator	To administer the HCL Commerce solution once deployed.
Business Users	To leverage the business user tooling of HCL Commerce. For example, to manage storefront catalogs, marketing, or other back-of-house operations.
Database Administrator	Responsible for administration of the HCL Commerce database.
AWS Network Engineer	To manage all aspects of the networking requirements for the solution.

2.4. Environment Configuration for Deployment (PRQ-003)

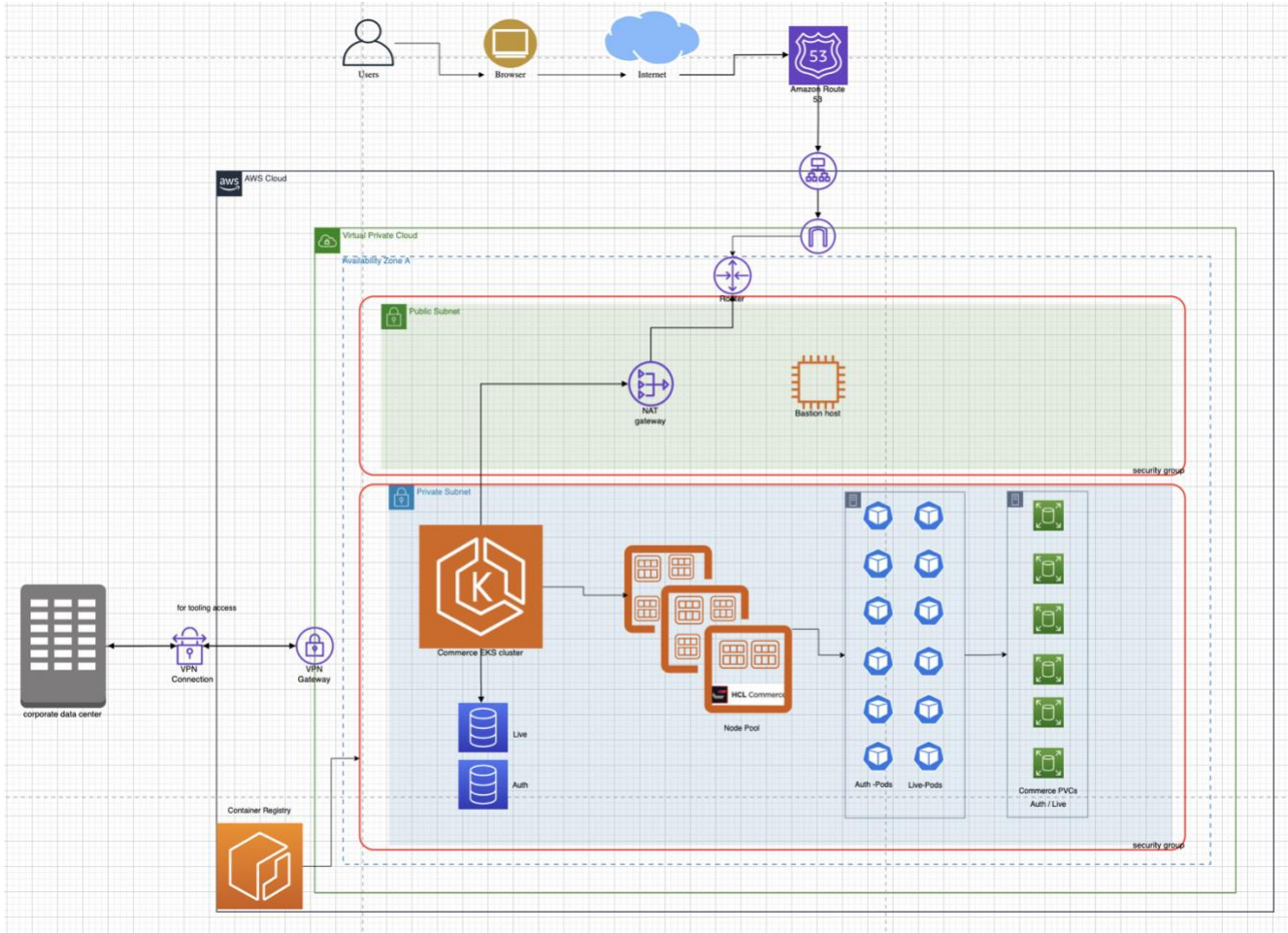
The following table lists the environment configuration that is required for the deployment (for example, an AWS account, a specific operating system, licensing, DNS).

Item	Description
AWS Account	Customer owned AWS account that is used to deploy HCL Commerce on AWS.
HCL Commerce Entitlement	Ensure that you are properly licensed to use HCL Commerce.

3. Architecture Diagrams

3.1. Standard HCL Commerce on AWS Deployment (ARCH-001)

A standard HCL Commerce architecture diagram using [AWS Simple Icons](#).

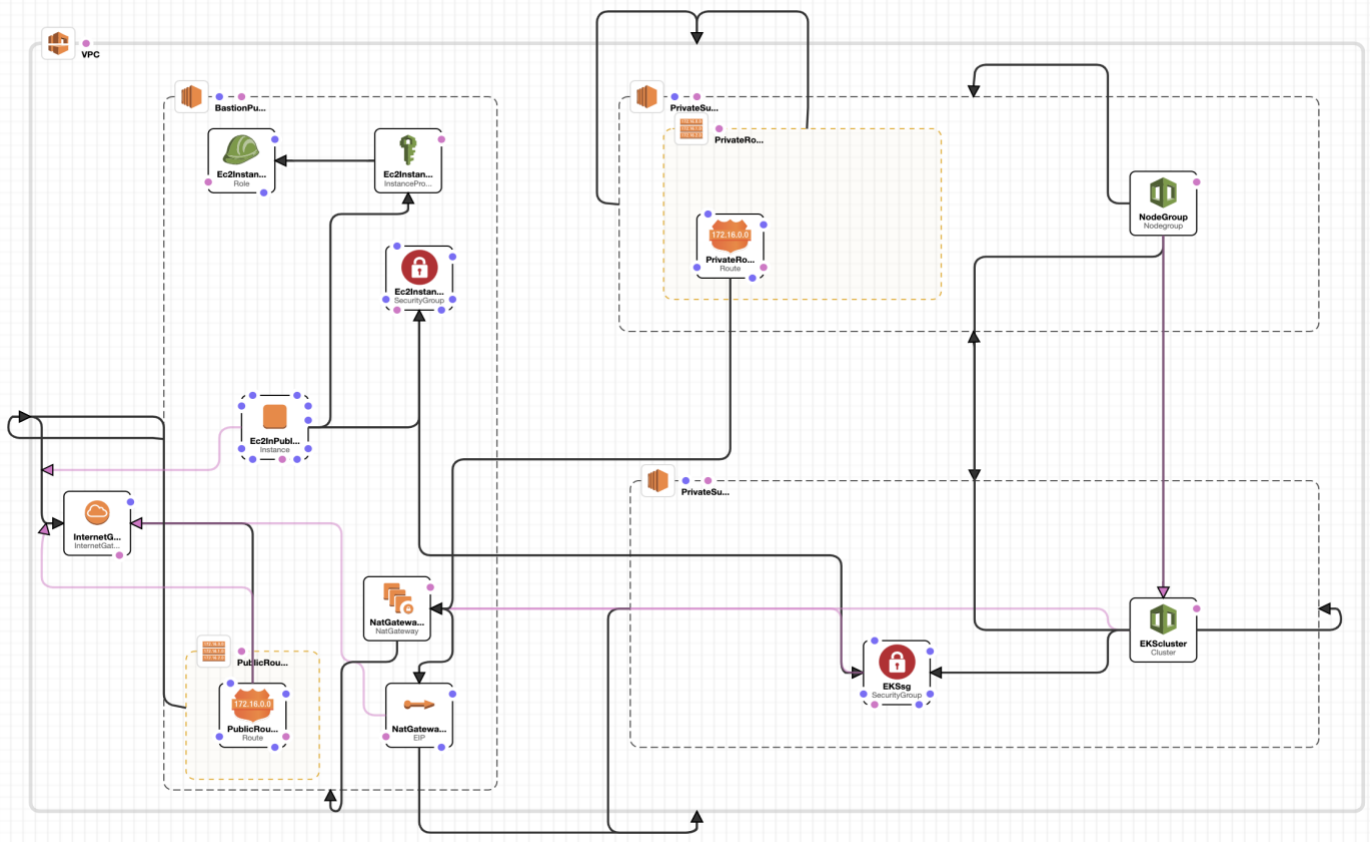


3.2. Customer Data Diagram(s) (ARCH-003 / DSEC-008)

Reference the above diagram. Customer data is stored in the database only.

3.3. Network Diagram(s) (ARCH-005 / DSEC-010)

The following network diagram is provided by the Cloud Formation Design Studio tooling.



3.4. Integration Point Diagram(s) (ARCH-006)

HCL Commerce can be integrated with a wide assortment of third-party solutions and assets. Its integration points depend specifically upon each of those solutions provided and whether there is code developed and deployed by the customer to integrate with those solutions.

A minimum integration requirement for a B2C e-commerce site would be with a payment system.

For more information, see [Integration](#) in the HCL Commerce documentation.

4. Security

4.1. IAM Best Practices (DSEC-001 / DSEC-002 / DSEC-003)

To help keep AWS resources secure, follow these recommended IAM best practices:

- Never use the AWS Root User Access Key.
- Avoid using the AWS Root User account.
- Create a separate AWS admin account to perform administrative tasks.
- Create individual IAM User accounts for each of your team members for them to perform their duties.

- Follow the [least privileges](#) principle when granting access to the individual IAM user accounts.
- Grant access to IAM Groups and add IAM users to these groups rather than granting permissions directly to individual IAM users.

Below are a few examples of IAM Groups that can be created:

- **Admin:** This group would grant full administrator access to the AWS environment. These users can perform all actions, including the delegation of permissions to all services and resources in AWS.
- **Read:** This group would grant read access to the specified AWS services.
- **API:** This group would grant programmatic access to AWS services.
- **Infrastructure:** Setting up infrastructure for HCL Commerce on AWS.
- **Application:** Custom permissions to deploy HCL Commerce within a cluster.

4.2. Public Resources (DSEC-004)

There are no public resources as part of the HCL Commerce deployment.

4.3. IAM Roles and Policies (DSEC-005)

User Groups	Role	Policies	Allow Permissions
Infrastructure	This function handles setting up all of the infrastructure related requirements to deploy HCL Commerce.	AmazonEC2FullAccess AWSCloudFormationFullAccess AmazonS3FullAccess hclsw_commerce_ftr_eks_cluster_role (custom role) _AmazonEKSClusterPolicy AmazonEKSWorkerNodePolicy AmazonEKS_CNI_Policy AmazonEC2ContainerRegistryReadOnly hclsw_commerce_eks_cluster_nodegroup_role (custom policy)	Standard AWS policies and permissions.
Application	This function is responsible for deploying the HCL Commerce application into the provided infrastructure.	EKSDeployAccess EKSIamLimitedAccess	See the JSON samples provided below.

Policy Name	JSON
EKSDeployAccess	<pre>{ "Version": "2012-10-17", "Statement": [{ "Effect": "Allow", "Action": "eks:*", "Resource": "*" }, { "Action": ["ssm:GetParameter", "ssm:GetParameters"], "Resource": ["arn:aws:ssm*:486772507565:parameter/aws/*", "arn:aws:ssm*:::parameter/aws/*"], "Effect": "Allow" }, { "Action": ["kms:CreateGrant", "kms:DescribeKey"], "Resource": "*", "Effect": "Allow" }, { "Action": ["logs:PutRetentionPolicy"], "Resource": "*", "Effect": "Allow" }] }</pre>
EKSiamLimitedAccess	<pre>"Version": "2012-10-17", "Statement": [{ "Effect": "Allow", "Action": ["iam:CreateInstanceProfile", "iam:DeleteInstanceProfile", "iam:GetInstanceProfile", "iam:RemoveRoleFromInstanceProfile", "iam:GetRole", "iam:CreateRole", "iam:DeleteRole", "iam:AttachRolePolicy", "iam:PutRolePolicy", "iam:ListInstanceProfiles", "iam:AddRoleToInstanceProfile", "iam:ListInstanceProfilesForRole", "iam:PassRole",</pre>

	<pre> "iam:DetachRolePolicy", "iam>DeleteRolePolicy", "iam:GetRolePolicy", "iam:GetOpenIDConnectProvider", "iam>CreateOpenIDConnectProvider", "iam>DeleteOpenIDConnectProvider", "iam:TagOpenIDConnectProvider", "iam:ListAttachedRolePolicies", "iam:TagRole"], "Resource": ["arn:aws:iam::YourAccountID:instance-profile/eksctl-*", "arn:aws:iam::YourAccountID:role/eksctl-*", "arn:aws:iam::YourAccountID:oidc-provider/*", "arn:aws:iam::YourAccountID:role/aws-service-role/eks- nodegroup.amazonaws.com/AWSServiceRoleForAmazonEKSNodegroup", "arn:aws:iam::YourAccountID:role/eksctl-managed-*"] }, { "Effect": "Allow", "Action": ["iam:GetRole"], "Resource": ["arn:aws:iam::YourAccountID:role/*"] }, { "Effect": "Allow", "Action": ["iam:CreateServiceLinkedRole"], "Resource": "*", "Condition": { "StringEquals": { "iam:AWSServiceName": ["eks.amazonaws.com", "eks-nodegroup.amazonaws.com", "eks-fargate.amazonaws.com"] } } }] } </pre>
--	---

4.4. User Created Keys for Deployment (DSEC-006)

The following keys are supported by the HCL Commerce Helm Chart and are used for deployment:

Key	Description
Image Pull Secret	<p>This key is used to pull Docker images from a private docker registry.</p> <p>The User must gather the information of the private registry and then use kubecttl command to create the image pull secret. There are several locations where users can apply this key which are the values.yaml files of the HCL Commerce and Vault Helm Charts.</p>
Vault Token Secret	<p>This key is used as the authentication method for the Vault service required when deploying using the HCL Commerce Helm Chart.</p> <p>The User must put the vault token value in both UUID format and Base 64 version in the values.yaml file of the Vault Helm Chart.</p> <p>A Vault token secret will be created within the Vault and Commerce namespaces. The HCL Commerce application can obtain the Vault token from that secret, and then use it to get the key value pairs stored in Vault.</p>
SPI User Authentication	<p>SPI user is used for inter-component server API invocation in Commerce. The username, encrypted password, and Base64 version of the password is specified in the values.yaml of the HCL Commerce Helm Chart.</p>
Commerce Ingress Certificates	<p>One or more secrets containing a TLS key and certificate used by the ingress controllers, or Ambassador, or Emissary in order to be accessed publicly. For production environments, users generate the secrets with a real CA certificate. The user can find the related configurations in the values.yaml file of the HCL Commerce Helm Chart under the ingress section.</p>
Vault CA Certificate	<p>A secret containing a TLS key and certificate marked for CA use (That is, basicConstraints=CA:TRUE) could be created manually or automatically by the support container. Vault would use it as a Certificate Authority to issue certificate to each application to communicate with one another securely. User would be able to find the related configurations in the values.yaml file of the Vault Helm Chart.</p>

4.5. Secrets Storage (DSEC-007)

HCL Commerce supports the use of Vault to store environment configurations and secrets, and currently does not support any other secrets management service.

Users must store database related credentials, JSON Web Key Sets, SPI user credentials, and some other environment configurations in Vault.

More configurations and some sample data can be found in the `values.yaml` file of the Vault Helm Chart.

4.6. Customer Sensitive Data Storage (DSEC-008)

Sensitive HCL Commerce customer (end-user) data is encrypted and stored in the HCL Commerce database. Sensitive deployment-related data, such as the administrator spiuser password and database information, is stored within Vault.

4.7. Data Encryption Configuration(s) (DSEC-009)

Sensitive customer data is encrypted and stored in the HCL Commerce database.

Sensitive data is encrypted using a customer specified, AES-128 bit encryption key, referred to as the merchant key. The merchant key must be periodically changed, using the **MigrateEncryptedInfo** utility. For more information, see [MigrateEncryptedInfo utility](#) in the HCL Commerce documentation.

HCL recommends to also natively encrypt your database at rest as an extra layer of security.

In this deployment guide, a sample DB2 database is used, and does not explicitly configuring this encryption.

To learn more about native DB2 encryption for data at rest, refer to [Db2 native encryption](#) in the IBM Db2 documentation.

For use with the Oracle database, refer to the appropriate documentation from Oracle.

It is recommended to secure the communication from the HCL Commerce application to the database. To learn more about how to configure this, see [Enabling SSL for database connections](#) in the HCL Commerce documentation.

5. Costs

5.1. Billable Services (Mandatory/Optional) (CST-001)

- AWS subscriptions are owned by and billed to the customer.
- HCL Commerce licensing is procured directly from HCL Software.

AWS Service	Description	Billable	Mandatory
Virtual Private Cloud (VPC)	To create a private network.	NO	YES
Elastic Compute Cloud (EC2)	Virtual Instances to host HCL Commerce and associated node pools.	YES	YES
Elastic Container Service	Container management service that allows you to easily run applications on a managed cluster of Amazon Elastic Compute Cloud instances.	YES	YES
Elastic Load Balancing	Used to distribute incoming application traffic across multiple targets.	YES	YES
Identity and Access Management (IAM)	To manage access to AWS services.	NO	YES
S3 / EBS	Used for storage within the HCL Commerce deployment.	YES	YES
Simple Notification Service (SNS)	Works with CloudWatch to provide notifications via email, etc.	YES	NO
CloudWatch	To monitor EC2 resources.	YES	NO
Support	AWS Support for troubleshooting AWS service specific issues.	YES	NO

Refer to <https://aws.amazon.com/> for the latest on pricing and documentation for AWS services.

5.2. Cost Model / Licensing Costs (CST-002)

HCL Commerce licensing costs are based on a variety of performance and resource related metrics. Customers must consult with an HCL Commerce Sales Representative to discuss pricing specifics.

6. Sizing

6.1. EC2 Instance Type and Size (SIZ-001)

The following tables list the recommended EC2 sizing of the HCL Commerce Servers for deployment. Given that HCL Commerce is a highly customizable solution, customers can deploy Commerce based on their own organizational requirements.

The minimum requirements to deploy an HCL Commerce auth and live instance is demonstrated in the following table. This EKS cluster sizing is based on the one replica for each component. In a production setting, EKS auto-scaling should be configured to scale resources to meet changing demands.

Component	Replica	Request CPU	Limit CPU	Request Memory	Limit Memory
ts-app	1	500m	2	5120Mi	5120Mi
ts-db	1	2	2	6144Mi	6144Mi
ts-web	1	500m	2	2048Mi	2048Mi
search-app-master	1	500m	2	4096Mi	4096Mi
search-app-repeater (live)	1	500m	2	4096Mi	4096Mi
search-app-slave (live)	1	500m	2	4096Mi	4096Mi
crs-app	1	500m	2	4096Mi	4096Mi
xc-app	1	500m	2	4096Mi	4096Mi
tooling-web	1	500m	2	2048Mi	2048Mi
store-web	1	500m	2	2048Mi	2048Mi
nifi-app	1	500m	2	10240Mi	10240Mi
registry-app	1	500m	2	2048Mi	2048Mi
ingest-app	1	500m	2	4096Mi	4096Mi
query-app	1	500m	2	4096Mi	4096Mi
cache-app	1	500m	2	2048Mi	2048Mi
graphql-app	1	500m	2	2048Mi	2048Mi
mustgather-app	1	500m	1	4096Mi	4096Mi
ts-utils	1	500m	2	4096Mi	4096Mi

Sizing for HCL Commerce is partially based upon peak order lines per hour, where an order line is a single line item in a transacted shopping cart. This metric gives the basis for the number of Transaction server containers that must be deployed within the Commerce environment. From this starting point, recommendations can be made for the other components that make up the HCL Commerce environment (Storefront, Search, Cache etc.).

HCL Commerce has a minimum set of hardware requirements depending on the planned deployment. For more information, see [Hardware requirements for HCL Commerce](#) in the HCL Commerce documentation.

# EKS Nodes	Type	vCPU	Memory	Storage
6	• m5.xlarge	4	16 GB	200 GB

In the case of rolling upgrades, multiple instances of each application can be active at the same time. As a result, additional vCPUs and memory is required.

The expected resources that are required for any deployment are primarily impacted by the search solution that is used in the deployment.

Below are the resource details for one replica deployment which also includes the overhead requirement that are utilized in a rolling upgrade scenario:

Search solution	vCPUs	Memory
Solr-based search solution	40	68 GB
Elasticsearch-based search solution	48	82 GB

6.2. EBS Volume Type and Size (SIZ-002)

HCL Commerce requires persistent volume storage for the deployment.

By default, the HCL Commerce Helm Chart creates a persistent volume claim with predefined capacities.

For production and critical workloads, you must define your own volume claim template with the desired storage capacity and the required Kubernetes storage class to associate with the persistent volume.

For a demo deployment, PVC for Elasticsearch, Zookeeper, and Nifi are also provisioned during the Helm Chart installation of these components.

```
commerce-nifi-pvc 10Gi
commerce-elasticsearch-pvc 30Gi
commerce-zookeeper-pvc 8Gi
```

Setting up the Asset Tool persistent volume

The Assets Tool was reintroduced in HCL Commerce 9.1.8.0. This tool requires persistent volume storage for your deployment. This storage allows for all assets that are added and managed via the Assets Tool in Management Center for HCL Commerce to be accessed and persisted.

To make files accessible by multiple pods across your deployment, and to allow for the files to be persisted, a ReadWriteMany type of persistent volume is required.

Note: If you are not planning on using the Assets Tool within your deployment, then persistent storage is not required.

For more information, see [Setting up persistent storage volumes for a Kubernetes deployment](#) in the HCL Commerce documentation.

6.3. Instance Size Selection for Managed AWS Services (SIZ-003)

This section is not applicable.

HCL Commerce does not rely on any of the managed AWS services. The customer must decide to use managed AWS services.

6.4. Amazon Dynamo DB Read/Write Capacity (SIZ-004)

This section is not applicable.

HCL Commerce does not support Amazon Dynamo DB.

7. Deployment Assets

7.1. Deploying workload on AWS as per typical deployment architecture (DAS-001)

The deployment of HCL Commerce is broken down into two sections.

1. The first section is deployment of the required infrastructure.
2. The second is then deploying the HCL Commerce solution in to said infrastructure.

All deployment steps are handled using AWS Cloud Formation and AWS CLI commands.

1. Infrastructure deployment

I. Install the utilities that are required:

- Set up **AWS CLI**.

The AWS Command Line Interface (**AWS CLI**) is an open-source tool that enables you to interact with AWS services using commands in your command-line shell.

The following link can be used to install or update the latest release of the **AWS CLI**.

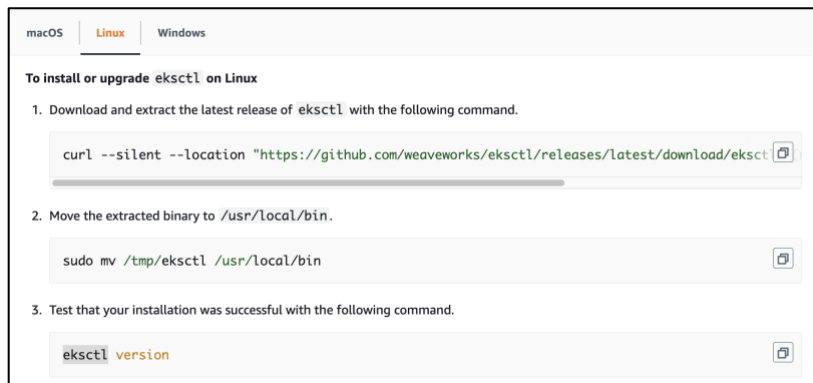
See [Installing or updating the latest version of the AWS CLI](#) in the Amazon documentation.

```
$ curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
unzip awscliv2.zip
sudo ./aws/install
```

- Set up **Eksctl**.

Eksctl is a simple command line utility for creating and managing Kubernetes clusters on Amazon EKS. The following link can be used to install **eksctl**.

See [Installing or updating eksctl](#) in the Amazon documentation.



The screenshot shows the 'Linux' tab of the eksctl installation guide. It lists three steps: 1. Download and extract the latest release of eksctl using a curl command. 2. Move the extracted binary to /usr/local/bin using a mv command. 3. Test the installation with the eksctl version command. Each step includes a code block with the respective command.

```
curl --silent --location "https://github.com/weaveworks/eksctl/releases/latest/download/eksctl"

sudo mv /tmp/eksctl /usr/local/bin

eksctl version
```

- Set up **Kubectl**.

Kubectl is used to run command against Kubernetes cluster. The following link can be used to install **Kubectl**.

See [kubectl](#) in the Kubernetes documentation.

```
[root@ip-10-192-10-21 ~]# curl -o kubectl https://s3.us-west-2.amazonaws.com/amazon-eks/1.22.6/2022-03-09/bin/linux/amd64/kubectl
% Total    % Received % Xferd Average Speed   Time    Time     Time  Current
           Dload  Upload   Total   Spent    Left  Speed
100 44.7M  100 44.7M    0     0 6399k      0  0:00:07  0:00:07 --:--:-- 6423k
[root@ip-10-192-10-21 ~]# openssl sha1 -sha256 kubectl
SHA256(kubectl)= 860c3d37a5979491895767e7332404d28dc0d7797c7673c33df30ca80e215a07
[root@ip-10-192-10-21 ~]# chmod +x ./kubectl
[root@ip-10-192-10-21 ~]# mkdir -p $HOME/bin && cp ./kubectl $HOME/bin/kubectl && export PATH=$PATH:$HOME/bin
[root@ip-10-192-10-21 ~]# kubectl version --short --client
Client Version: v1.22.6-eks-7d68063
[root@ip-10-192-10-21 ~]#
```

- Set up **Helm**.

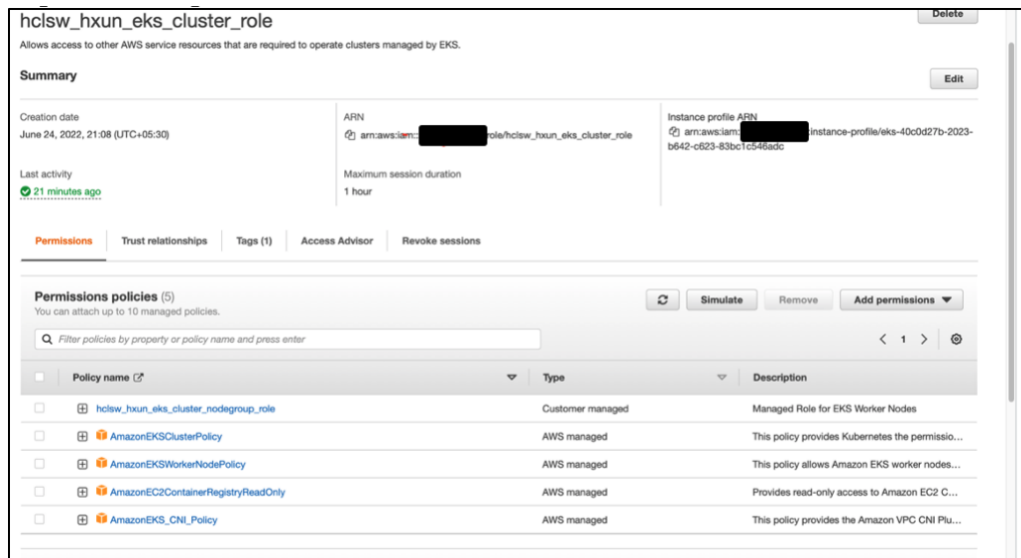
Helm is the package manager for Kubernetes. It is used to deploy Kubernetes applications easily. The following link can be used to install **Helm**.

See [Installing Helm](#) in the Helm documentation.

II. Create an IAM role for EKS Clusters and Node Pools.

Before you can create EKS Clusters and Node Pools, you must create an IAM role with the required IAM policies to do so.

In the example below you can observe the custom role, **hclsw_hxun_eks_cluster_role**, and all of the **AWS Managed** permission policies that make up this role.



III. Create a VPC, EC2 (Bastion), and EKS cluster using AWS cloud formation.

A customized cloud formation template is already available which can be used to create the stack VPC, subnets, routes, gateways, etc, as per the template values.

Navigate to the CloudFormation user interface to create the new stack by uploading the provided template as shown in the following screenshot.

aws

Services

Search for services, features, blogs, docs, and more

[Option+S]

CloudFormation > Stacks > Create stack

Step 1

Specify template

Step 2

Specify stack details

Step 3

Configure stack options

Step 4

Review

Create stack

Prerequisite - Prepare template

Prepare template

Every stack is based on a template. A template is a JSON or YAML file that contains configuration information about the AWS resources you want to include in the stack.

Template is ready

Use a sample template

Create template in Designer

Specify template

A template is a JSON or YAML file that describes your stack's resources and properties.

Template source

Selecting a template generates an Amazon S3 URL where it will be stored.

Amazon S3 URL

Upload a template file

Upload a template file

Choose file

commerce_master_ftr

JSON or YAML formatted file

S3 URL: https://s3-external-1.amazonaws.com/cf-templates-1ohdmfmlzmnvt-us-east-1/20221952up-commerce_master_ftr

View in Designer

Cancel

Next

Once resource creation is completed, its status will be displayed as CREATE_COMPLETE.

Stacks (5)

Filter by stack name

Active

View nested

commerce-ftr-stack

2022-07-14 19:18:28 UTC+0530

CREATE_IN_PROGRESS

Stacks

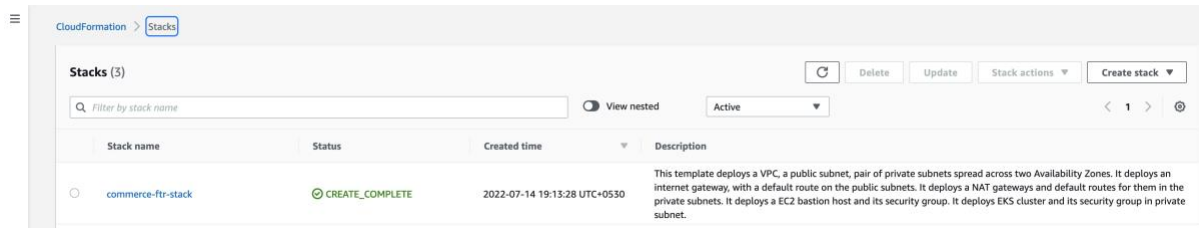
commerce-ftr-stack

Stack info Events Resources Outputs Parameters Template Change sets

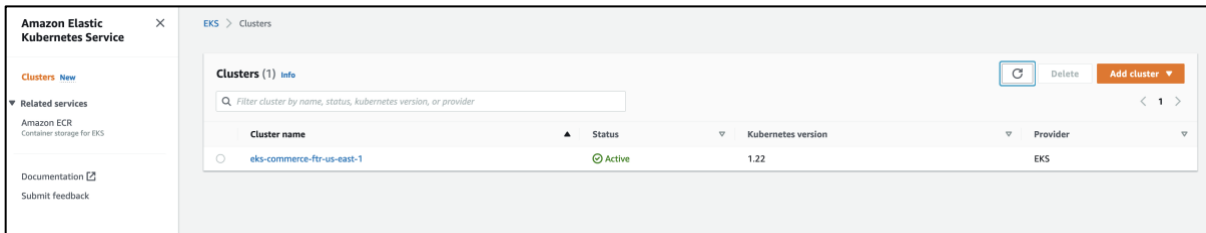
Resources (20)

Logical ID	Physical ID	Type	Status	Status reason	Module
BackendHost	i-04318dd96a3776881	AWS::EC2::Instance	CREATE_COMPLETE	-	-
BackendPublicSubnet	subnet-07a29e1a08a544754	AWS::EC2::Subnet	CREATE_COMPLETE	-	-
CommerceFTRVPC	vpc-02a287a5a6b1a9576	AWS::EC2::VPC	CREATE_COMPLETE	-	-
EC2Cluster	commerce-ftr	AWS::EKS::Cluster	CREATE_COMPLETE	-	-
EC2sg	sg-003469ed02350c19a	AWS::EC2::SecurityGroup	CREATE_COMPLETE	-	-
InternetGateway	igw-0738ae74888a10006	AWS::EC2::InternetGateway	CREATE_COMPLETE	-	-
NatGateway1	nat-082a001f59f36475	AWS::EC2::NatGateway	CREATE_COMPLETE	-	-
NatGateway1EIP	14.88.68.111	AWS::EC2::EIP	CREATE_COMPLETE	-	-
NodeGroup	commerce-ftr-fleet-com-ftr	AWS::EKS::Nodegroup	CREATE_COMPLETE	-	-
PrivateRoute	commer-Private-WQF087J59V1K	AWS::EC2::Route	CREATE_COMPLETE	-	-
PrivateRouteTable	rftb-09a1ea9827c5a6d06	AWS::EC2::RouteTable	CREATE_COMPLETE	-	-
PrivateSubnet1EKS	subnet-06219f1e03a5a4a08	AWS::EC2::Subnet	CREATE_COMPLETE	-	-
PrivateSubnet1RouteTableAssociation	rftbassoc-00376a45a76a4d946	AWS::EC2::SubnetRouteTableAssociation	CREATE_COMPLETE	-	-
PrivateSubnet2EKS	subnet-0a1030c09967732baf	AWS::EC2::Subnet	CREATE_COMPLETE	-	-
PrivateSubnet2RouteTableAssociation	rftbassoc-08f6d5245af35a4	AWS::EC2::SubnetRouteTableAssociation	CREATE_COMPLETE	-	-
PublicRoute	commer-Public-V79F520F1M5U	AWS::EC2::Route	CREATE_COMPLETE	-	-
PublicRouteTable	rftb-022a46544a1f579a7	AWS::EC2::RouteTable	CREATE_COMPLETE	-	-
PublicSubnetRouteTableAssociation	rftbassoc-02b0da46a4658729a	AWS::EC2::SubnetRouteTableAssociation	CREATE_COMPLETE	-	-
SecurityGroupEC2	sg-9a7a3055468276c70	AWS::EC2::SecurityGroup	CREATE_COMPLETE	-	-
VPCGatewayAttachment	commer-VPCGa-175D84XVJJDH1	AWS::EC2::VPCLGatewayAttachment	CREATE_COMPLETE	-	-

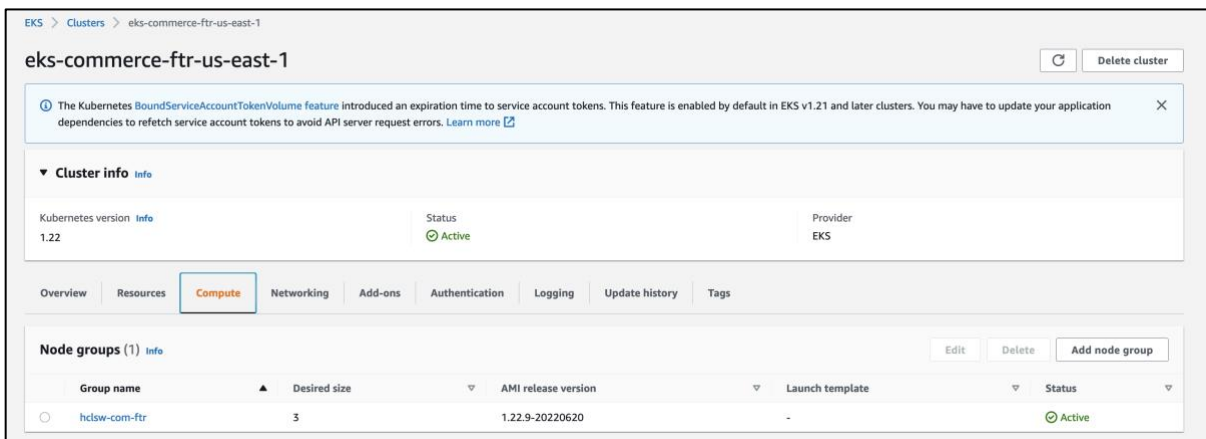
26



Verify the status in the EKS console. It is displayed as **Active** once creation is completed.



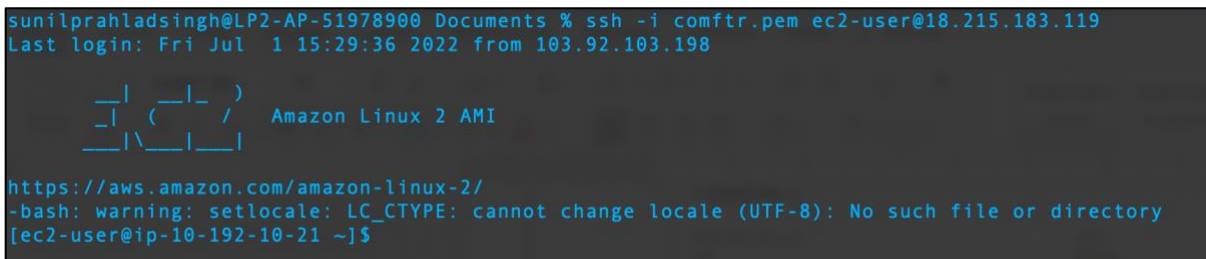
Verify the Node Pool from the EKS console.



IV. Connect to the Bastion host.

Log in to the Bastion host.

```
ssh -i comftr.pem ec2-user@bastion_ipAddress
```



V. Connect to the cluster from the Bastion host.

- For a production environment, it is recommended to configure private control plane access so that connectivity is only permitted from the Bastion host.
- For a development or testing environment, you can configure private/public control plane access so that the cluster can be accessed using local workstations.

To connect and access the cluster using kubectl APIs, use the following commands:

- Authenticate the Bastion host to the EKS cluster.

```
aws eks update-kubeconfig --region us-east-1 --name eks-hxun-n-us-east-1
```

```
[root@ip-10-192-10-21 ~]# aws configure
[AWS Access Key ID [None]: AKIAY3QM43SHRMOLFJRE
[AWS Secret Access Key [None]: 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
[Default region name [None]: us-east-1
[Default output format [None]: json
[root@ip-10-192-10-21 ~]# aws eks update-kubeconfig --region us-east-1 --name eks-commerce-ftr-us-east-1
Added new context arn:aws:eks:us-east-1:608838671503:cluster/eks-commerce-ftr-us-east-1 to /root/.kube/config
[root@ip-10-192-10-21 ~]#
```

- Add a new user to aws-auth configMap to allow EKS access.

```
# Please edit the object below. Lines beginning with a '#' will be ignored,
# and an empty file will abort the edit. If an error occurs while saving this file will be
# reopened with the relevant failures.
#
apiVersion: v1
data:
  mapRoles: |
    - groups:
      - system:bootstrappers
      - system:nodes
      rolearn: arn:aws:iam::608838671503:role/hclsw_commerce_ftr_eks_cluster_role
      username: system:node:{{EC2PrivateDNSName}}
  mapUsers: |
    - groups:
      - system:masters
      userarn: arn:aws:iam::608838671503:user/eksuser
      username: eksuser
kind: ConfigMap
metadata:
  creationTimestamp: "2022-06-30T14:44:47Z"
  name: aws-auth
  namespace: kube-system
  resourceVersion: "929684"
  uid: ac873ab0-2603-4923-9809-92972ffefcf4
~
```

```
[sunilprahladsingh@LP2-AP-51978900 Documents % kubectl get cm -A
NAMESPACE      NAME                               DATA  AGE
default         kube-root-ca.crt                  1      3d21h
kube-node-lease kube-root-ca.crt                  1      3d21h
kube-public     kube-root-ca.crt                  1      3d21h
kube-system     aws-auth                          1      3d21h
kube-system     coredns                          1      3d21h
kube-system     cp-vpc-resource-controller        0      3d21h
kube-system     eks-certificates-controller       0      3d21h
kube-system     extension-apiserver-authentication 6      3d21h
kube-system     kube-proxy                        1      3d21h
kube-system     kube-proxy-config                 1      3d21h
kube-system     kube-root-ca.crt                  1      3d21h
[sunilprahladsingh@LP2-AP-51978900 Documents % kubectl edit cm aws-auth -n kube-system
configmap/aws-auth edited
```

For more information, see [Enabling IAM user and role access to your cluster](#) in the Amazon AWS documentation.

VI. Install the required utilities on the Bastion host to work with EKS clusters.

Refer to [1.i.](#) of Infrastructure deployment to install **kubectrl**, **eksctl**, **helm** and **aws-cli** on the Bastion host.

2. Application deployment

- I. Download the HCL Commerce images from HCL.
 - a. [Log in to Flexnet](#). This requires an account with the appropriate download permissions.
 - b. Click on HCL Commerce under the download section and obtain the associated Docker images and Helm Charts for the version of HCL Commerce that you want to deploy.

HCL SOFTWARE

Home | Activation & Entitlements | License Support | Devices | Downloads | Accounts & Users

License & Download Portal

Recent Entitlements

Activation ID	Product	Product description	Last modified
FLX_20220704055453	HCL BigFix Try and Buy		Jul 4, 2022
f9a1-faec-8b49-4849-a61a-b...	HCL AppScan Standard Edition Authorized User Single Install v9	HCL AppScan Standard Single User product, dynamic analysis desktop tool using the Single User license model (Node Locked).	Jul 4, 2022
a17a-004f-87e2-41b8-bcac-c...	HCL Domino Utility Express		Jul 4, 2022

Recent Releases

Description	Date
HWA Plugins June 24 Pre-Release	Jun 24, 2022
HCL Commerce Enterprise v9.0.1.18 without IBM Cloud Private	Mar 08, 2022
HCL VersionVault v3.0.0	Mar 25, 2022
HCL Exacto version 3.0.1	Mar 29, 2022
HCL ZIE for Windows 1.1.2	Mar 30, 2022

Your Downloads

- HCL BigFix
- HCL Clara
- HCL Commerce**
- HCL Compass
- HCL Connections
- HCL DXPN

Announcements

QuickLinks

- List Entitlements
- List Licenses
- List Devices
- List Accounts
- List Users

</

- II. Create an ECR within an AWS project and load the HCL Commerce Docker images.
 - a. Create an ECR within AWS.

Services

Q ECR

N. Virginia

Containers

Amazon Elastic Container Registry

Share and deploy container software, publicly or privately

Amazon Elastic Container Registry (ECR) is a fully managed container registry that makes it easy to store, manage, share and deploy your container images and artifacts anywhere.

Create a repository

Get started

Pricing (US)

You only pay for the amount of data you store in your public or private repositories and data transferred to the Internet.

Find out more

Getting started

What is Amazon ECR?

Getting started with ECR

Set up a CI/CD pipeline with ECR

Review public patterns

How it works

Benefits

- b. Load the HCL Commerce images locally.

For example,

`docker load --input HCL_Commerce_Enterprise_9.1.10.0_Support_Container_x86-64.tgz`

```
PS C:\Users\hcladmin\Downloads\9110-Flexnet> docker load --input HCL_Commerce_Enterprise_9.1.10.0_Support_Container_x86-64.tgz
e2eb06d8af82: Loading layer [=====] 5.865MB/5.865MB
a2f23be74558: Loading layer [=====] 763.4kB/763.4kB
bc2c1183a08: Loading layer [=====] 27.500kB/27.500kB
dbf4864aaaf7: Loading layer [=====] 4.608kB/4.608kB
f2647165e84e: Loading layer [=====] 8.61MB/8.61MB
8b264723f2a7: Loading layer [=====] 46.350kB/46.350kB
c805f960b117: Loading layer [=====] 110.1kB/110.1kB
848e2183ca0a: Loading layer [=====] 110.1kB/110.1kB
Loaded image: commerce/supportcontainer:9.1.10.0
PS C:\Users\hcladmin\Downloads\9110-Flexnet> docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
commerce/supportcontainer 9.1.10.0           2368869212a8       3 months ago       84.8MB
```

- c. Tag the loaded images with an ECR URL, and push the Docker images to the AWS ECR registry.

For example,

`docker tag imageid 6088386XXXXX.dkr.ecr.us-east-1.amazonaws.com/commerce-9110-ftr:9.1.10.0`

`docker push 6088386XXXXX.dkr.ecr.us-east-1.amazonaws.com/commerce-9110-ftr:9.1.10.0`

```
PS C:\Users\hcladmin> docker images | findstr 9.1.10.0
608838671503.dkr.ecr.us-east-1.amazonaws.com/commerce-9110-ftr 9.1.10.0
2368869212a8 3 months ago 84.8MB
commerce/supportcontainer 9.1.10.0
2368869212a8 3 months ago 84.8MB
PS C:\Users\hcladmin> docker push 608838671503.dkr.ecr.us-east-1.amazonaws.com/commerce-9110-ftr:9.1.10.0
```

- d. Transfer the HCL Commerce Helm Chart bundle file downloaded from Flexnet to the Bastion EC2 instance.

For example,

`scp -i comftr2.pem HCL_Commerce_Helm_Charts_9.1.10.0.bundle ec2-user@18.215.183.119:/tmp/`

```
PLS1-AP-51879415 commerce-helm % scp -i comftr2.pem HCL_Commerce_Helm_Charts_9.1.10.0.bundle ec2-user@18.215.183.119:/tmp/
```



```

saaurabhumathe@LS1-AP-51879415 commerce-helm % ssh -i comftr2.pem ec2-user@18.215.183.119
Last login: Tue Jul  5 13:01:52 2022 from 183.87.112.174

  __|  __|_  )
 _| (  /   Amazon Linux 2 AMI
---|\\___|___|

https://aws.amazon.com/amazon-linux-2/
-bash: warning: setlocale: LC_CTYPE: cannot change locale (UTF-8): No such file or directory
[ec2-user@ip-10-192-10-21 ~]$ cd /tmp
[ec2-user@ip-10-192-10-21 tmp]$ ls -lrt
total 672
drwx----- 3 root      root          17 Jun 30 13:58 systemd-private-2ddba80c8edb4ff187f308f9c0df392a-chronyd.service-7GXkil
-rw-r--r-- 1 ec2-user  ec2-user    684827 Jul  5 13:14 HCL_Commerce_Helm_Charts_9.1.10.0.bundle

```

- e. Extract the git bundle with git clone.

For example,

```
git clone HCL_Commerce_Helm_Charts_9.1.10.0.bundle
```

```

[ec2-user@ip-10-192-10-21 tmp]$
[ec2-user@ip-10-192-10-21 tmp]$ git clone HCL_Commerce_Helm_Charts_9.1.10.0.bundle
Cloning into 'HCL_Commerce_Helm_Charts_9.1.10.0'...
Receiving objects: 100% (778/778), 668.65 KiB | 39.33 MiB/s, done.
Resolving deltas: 100% (590/590), done.
[ec2-user@ip-10-192-10-21 tmp]$ ls -lrt
total 672

```

- f. Configure AWS to connect to the cluster.

```
aws eks update-kubeconfig --region us-east-1 --name eks-commerce-ftr-us-east-1
```

```
aws configure
```

```

[ec2-user@ip-10-192-10-21 HCL_Commerce_Helm_Charts_9.1.10.0]$ aws eks update-kubeconfig --region us-east-1 --name eks-commerce-ftr-us-east-1
Unable to locate credentials. You can configure credentials by running "aws configure".
[ec2-user@ip-10-192-10-21 HCL_Commerce_Helm_Charts_9.1.10.0]$ aws configure

```

III. Configure and deploy the HCL Commerce Helm Chart to deploy the application.

Note: The following steps are derived from the HCL Commerce deployment documentation. For a different format, or further clarification, review [Deploying HCL Commerce Version 9.1 on Kubernetes](#) in the HCL Commerce documentation.

HCL Commerce supports several deployment configurations. The default configuration mode used by the provided Helm Chart is Vault configuration mode. Vault is also the recommended configuration mode for HCL Commerce as it was designed to store configuration data securely. HCL Commerce also uses Vault as a Certificate Authority to issue certificate to each application to communicate with one another securely. Therefore, ensure that you have a Vault service available for HCL Commerce to access. The following steps highlight the minimum requirements before deploying HCL Commerce.

For non-production environments, you can consider the use of hcl-commerce-vaultconsul-

helmchart to deploy and initialize Vault for HCL Commerce as it can initialize the Vault and populate data for HCL Commerce. However, that chart runs Vault in development and non-high availability (HA) mode and does not handle Vault token securely. Therefore, it should not be used for production environments.

a. To deploy a demo instance, the following changes are required.

1. **LICENSE=accept**
2. **imageRepo=ECR_repo_where_the_HCL_Commerce_images_are_pushed**
3. Change the **Enabled** flag to *true* for the services to be deployed with HCL Commerce.
4. Change the Redis, Zookeeper, and Elasticsearch service URL values in Vault to include correct namespace and service name.
5. For Triggering Auto indexing, set the search index creation flag to *true*.
6. If deploying a live environment, set the **pushToLiveEnabled** flag to *true*.

b. Configure the database.

In this document, the sample out-of-the-box containerized DB2 database is used.

Note: It is recommended to set up and use one of the supported database servers outside of the cluster. However, for demo usage HCL Commerce also ships with an out-of-the-box containerized DB2 sample database which HCL Commerce deployment can connect to.

To use a supported database, see [Configuring a database for HCL Commerce](#).

Deploy the HCL Commerce IBM Db2 image on EC2 instance outside the cluster. This instance is set up in a private subnet and has no external IP assigned.

To expedite the deployment, the following docker-compose file can be used. This image cannot be used in a production environment.

version: '2.3'

services:

db:

image: gcr.io/blackjack-209019/services/commerce/release/ts-db:9.1.10.0

hostname: auth_db

privileged: true

environment:

- LICENSE=accept

ports:

- 50000:50000

healthcheck:

test: exit 0

You can check the Docker container status using the following command:

`sudo docker ps -a`

```
[ec2-user@ip-10-192-21-77 tmp]$ sudo docker ps -a
```

CONTAINER ID	IMAGE	NAMES	COMMAND	CREATED	STATUS	PORTS
15915b83f5af	gcr.io/blackjack-209019/services/commerce/release/ts-db:9.1.10.0		"/SETUP/bin/entrypoi..."	26 hours ago	Up 6 hours (healthy)	0.0.0.0:50000-
>50000/tcp, :::50000->50000/tcp, 50001/tcp tmp_db_1						

```
[ec2-user@ip-10-192-21-77 tmp]$
```

By default, the Db2 database listens on port 50000.

c. Deploy NGINX ingress.

- i. Create the ingress-nginx namespace.
- ii. Deploy NGINX ingress to the EKS cluster.

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-v1.2.1/deploy/static/provider/aws/deploy.yaml -n namespace
```

iii. Check the pod health. Wait until everything reports ready and running.

```
kubectl get pods -n namespace
```

Note: If you get the following error, change the ingress API version to /v1 from /v1beta1. Kubernetes version 1.22 only supports /v1 and not /v1beta1 apiVersion: networking.k8s.io/v1

```
Error: unable to build kubernetes objects from release manifest: unable to recognize "": no matches for kind "IngressClass" in version "networking.k8s.io/v1beta1"
```

d. Install Vault using the HCL Commerce hcl-commerce-vault Helm Chart.

- i. Add the appropriate DB hostname and port values for the remote Db2 host VM.
- ii. Create a namespace for Vault.

```
kubectl create ns vault
```

iii. Install Vault.

```
helm install vault ./ -f values.yaml -n vault
```

```
saarabhumathe@LS1-AP-51879415 hcl-commerce-vaultconsul %  
saarabhumathe@LS1-AP-51879415 hcl-commerce-vaultconsul % helm install vault ./ -f values.yaml -n vault  
NAME: vault  
LAST DEPLOYED: Tue Jul 5 13:13:57 2022  
NAMESPACE: vault  
STATUS: deployed  
REVISION: 1  
TEST SUITE: None  
NOTES:  
Chart: hcl-commerce-vaultconsul-2.1.2  
  
This vault deployment is for development only.
```

e. Deploy the required third-party applications, Elasticsearch, Zookeeper

i. Deploy Elasticsearch.

1. Create a namespace for Elasticsearch.

```
kubectl create ns elastic
```

2. Add the Elasticsearch Helm Chart repository.

```
helm repo add elastic https://helm.elastic.co
```

3. Deploy Elasticsearch using a local `elasticsearch-values.yaml` file. A sample version of this file is available in the `sample_values` directory of your cloned HCL Commerce Helm Chart Git project.

```
helm install elasticsearch elastic/elasticsearch -n elastic -f  
elasticsearch-values.yaml
```

4. Monitor the deployment and ensure that all pods are healthy.

```
[ec2-user@ip-10-192-10-21 hcl-commerce]$ kubectl get pods -n el  
NAME                                READY   STATUS    RESTARTS   AGE  
hcl-commerce-elasticsearch-0        1/1     Running   0           8d  
[ec2-user@ip-10-192-10-21 hcl-commerce]$
```

ii. Deploy Zookeeper.

1. Create a namespace for Zookeeper.

```
kubectl create ns zookeeper
```

2. Add the Helm Chart repository.

```
helm repo add bitnami https://charts.bitnami.com/bitnami
```

3. Deploy Zookeeper with the provided `zookeeper-values.yaml` deployment configuration file. A copy of this file can be found in the `sample_values` directory of your cloned HCL Commerce Helm Chart Git project.

```
helm install my-zookeeper bitnami/zookeeper -n zookeeper -f  
zookeeper-values.yaml
```

4. Monitor the deployment and ensure that all pods are healthy.

```
[ec2-user@ip-10-192-10-21 hcl-commerce]$ kubectl get pods -n zookeeper  
NAME                                READY   STATUS    RESTARTS   AGE  
hcl-commerce-zookeeper-0           1/1     Running   0           7d23h  
[ec2-user@ip-10-192-10-21 hcl-commerce]$
```

iii. Deploy Redis.

1. Create a namespace for Redis.

```
kubectl create ns redis
```

2. Add the Helm Chart repository.

```
helm repo add bitnami https://charts.bitnami.com/bitnami
```

3. Deploy Redis using a local `redis-values.yaml` file. A sample version of this file is available in the `sample_values` directory of your cloned HCL Commerce Helm Chart Git project.

```
helm install my-redis bitnami/redis -n redis -f redis-values.yaml --set  
master.disableCommands="" --version="redis-chart-version"
```

Note: Some older versions of the Redis Helm Chart are deprecated by

bitnami, such as the version 15.5.2 sample that was for HCL Commerce 9.1.10. Download the latest released version of the HCL Commerce Helm Chart for the most up-to-date and compatible version of the Redis Helm Chart.

4. Monitor the deployment and ensure that all pods are healthy.

```
[ec2-user@ip-10-192-10-21 hcl-commerce]$ kubectl get pods -n red
NAME                                READY   STATUS    RESTARTS   AGE
hcl-commerce-redis-master-0        1/1     Running   0           7d23h
```

- f. Configure your HCL Commerce deployment Helm Chart.

Use the provided `hcl-commerce-helmchart` to customize your deployment. See the A readme file provided in the helm chart describes the significance of each configurable value in Helm Chart.

Note: It is strongly recommended to not modify the default `values.yaml` configuration file for your deployment. Instead create a copy to use as your customized values file, for example, `my-values.yaml`. This will allow you to maintain your customized values for future deployments and upgrades.

- g. Use **Helm** to control the deployment of HCL Commerce.

Once you have completed the configuration of your deployment in your `my-values.yaml` file and meet the environment prerequisites, you are ready to deploy HCL Commerce by using Helm.

Commerce deployment consists of two environment deployments *auth* and *live*. In addition, a separate deployment for Elasticsearch is called the *shared* environment.

- *Auth* deploys the authoring instance of the HCL Commerce deployment.
 - *Live* deploys the live instance of commerce deployment.
 - *Share* deploys the shared instance of the Elasticsearch deployment.
- These components, including NiFi, Ingest, and Registry, are required by the Elasticsearch-based search solution in both the auth and live HCL Commerce deployments.

Note: For a production environment, it is recommended to use live instances, while for demo or test purposes you can rely solely on the deployment of an auth instance.

You can set the **environmentType** fields in a *values* file to deploy the corresponding instances.

Note: To deploy all the environment types, Share, Auth and Live in single Helm deployment, you must include the environments in the Helm values file with comma separated values.

For example,
environmentType: share,auth,live

This will deploy all the pods search, auth and live commerce into the Kubernetes cluster.

Run the deployment:

```
helm install pathtoHelmChart -f my-values.yaml -n commerce
```

To deploy instances separately:

```
helm install pathtoHelmChart -f my-values.yaml --set common.environmentType=live -n commerce
```

Once the HCL Commerce applications are deployed, if you have further configuration changes or image updates, you can use Helm upgrade command to update the deployment.

To update a deployment run the following Helm command for the release and **environmentType** that you want to update. This can be used with or without the **environmentType** parameter.

```
helm upgrade release-name pathtochart -f my-values.yaml --set  
common.environmentType=environmentType -n commerce
```

Notes:

- *If you use the Elasticsearch-based search solution, it is required to use a completely new persistent volume for NiFi, and clear any existing Zookeeper data before you redeploy. This is required so that the newer version of the connectors can be created automatically during the deployment.*
 - *To clear the NiFi data:*
 1. *Create a new Persistent Volume Claim (PVC), and configure the new PVC name in your deployment values.yaml file.*
 2. *You can then remove the previous attached persistent volume claim.*
kubectrl delete pvc previous_pvc_name -n commerce
 - *To clear Zookeeper data:*
 1. *Delete the existing Zookeeper instance.*
helm delete my-zookeeper -n zookeeper
 2. *Remove the existing persistent volume claims.*
kubectrl delete pvc --all -n zookeeper
 3. *Re-deploy Zookeeper.*
- *HCL Cache caches classes that can be modified in newer versions of HCL Commerce. To avoid errors in de-serializing an old version of the class, it is strongly recommended to clear Redis keys after upgrading HCL Commerce. Redis keys can be cleared with the Redis flushdb or flushall commands.*

Once the deployment is complete, check the pods status of the pods.

`kubectl get pods -n namespace`

```
[ec2-user@ip-10-192-10-21 hcl-commerce]$ kubectl get pods -n commerce
```

NAME	READY	STATUS	RESTARTS	AGE
commerce-commercenfs-0	1/1	Running	0	8h
commerce-hclcom-index-creation-demoqa-1eecxirrx3--1-w6fqd	0/1	Completed	0	119m
demoqaauthcrs-app-5894f4b9bf-vhdzp	1/1	Running	0	7h13m
demoqaauthdb-5bbfd8fcc-zn9tp	1/1	Running	0	8h
demoqaauthquery-app-6b56bbcb8-mqn2t	1/1	Running	0	7h13m
demoqaauthstore-web-6fbd568799-gt5sw	1/1	Running	0	119m
demoqaauthts-app-7db696586b-mn4m8	1/1	Running	0	8h
demoqaauthts-web-5b4dcf6b4c-z2698	1/1	Running	0	7h13m
demoqaingest-app-6bdc7949b8-4xhgb	1/1	Running	0	8h
demoqalivecrs-app-84fc8cff69-pscmt	1/1	Running	0	8h
demoqalivedb-5b758c5b65-kxdsg	1/1	Running	0	8h
demoqalivequery-app-7476b454fb-85z2k	1/1	Running	0	8h
demoqalivestore-web-6b8f4b4c75-b4m87	1/1	Running	0	119m
demoqalivets-app-6b5c7ffdd5-z68gz	1/1	Running	0	8h
demoqalivets-web-69c9f49767-mk6h8	1/1	Running	0	8h
demoqanifi-app-b9cc9df4f-29n8s	1/1	Running	0	8h
demoqaquery-app-76bff94ff-xmg5s	1/1	Running	0	8h
demoqaregistry-app-6d87598b4f-tq8kg	1/1	Running	0	8h
demoqatooling-web-8d7958d86-bbtst	1/1	Running	0	8h

```
[ec2-user@ip-10-192-10-21 hcl-commerce]$
```

h. Build the search index.

An Elasticsearch-based search solution search index can be built with multiple ways.

In recent version of commerce a index-creation job is added with runs after deployment and build the indices of the *storeID* values which are provided within the Helm *values* file.

Once the index is built you will observe the status of pod as completed.

```
commerce-commercenfs-0 1/1 Running 0 8h52m
commerce-hclcom-index-creation-demoqa-y3sqtlyl4k6--1-45dbd 0/1 Completed 0 5h52m
```

A search index can also be built manually using APIs, either from the Transaction server (*ts-app*) or ingest swagger.

Below are the APIs that can be used to trigger indexing manually from the Transaction server for an auth instance and, and then propagated to the live instance using push-to-live.

POST: `https://CommerceServerName`

`:CommerceServerPort/wcs/resources/admin/index/dataImport/build?connectorId=auth.reindex&storeId=1`

POST: `https://CommerceServerName:CommerceServerPort`
`/wcs/resources/admin/index/dataImport/build?connectorId=push-to-live&storeId=1`

Leave the Body empty and use basic authentication with the username *spiuser* and the password *spiuserPassword*.

The screenshot shows a REST client interface with the following details:

- URL:** `https://tsapp.demoqaauth.awsfr.com/wcs/resources/admin/index/dataImport/build?connectorId=p...`
- Method:** POST
- Authorization:** Basic A... (Selected)
- Username:** spiuser
- Password:** (Masked)
- Show Password:** ☐
- Response:** 202 Accepted, 673 ms, 257 B. Status: `"jobStatusId": "1004"`

You can verify the status via the REST interface.

GET: `https://CommerceServerName`
`:CommerceServerPort/wcs/resources/admin/index/dataImport/status?jobStatusId=jobStatusId`

URL: `https://tsapp.demoqaauth.awsfr.com/wcs/resources/admin/index/dataimport/status?jobStatusId=10C...`

Method: GET

Authorization: Basic A...

Username: spiuser

Password: [redacted]

Show Password: ☐

Response Status: 200 OK, 244 ms, 622 B

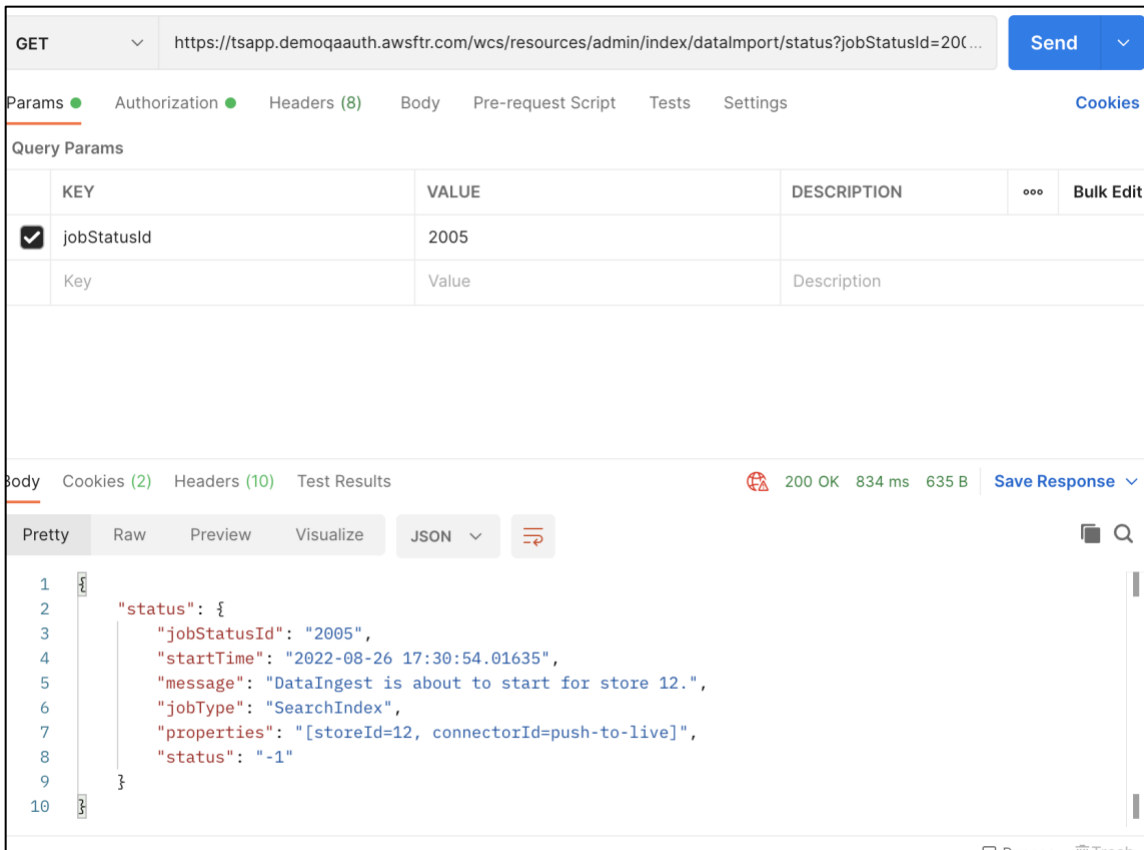
Response Body (JSON):

```

{
  "status": {
    "finishTime": "2022-07-13 11:48:20.404467",
    "lastUpdate": "2022-07-13 11:48:20.404467",
    "progress": "100%",
    "jobStatusId": "1004",
    "startTime": "2022-07-13 11:47:47.317164",
    "runId": "i-1c9474b8-6a77-40af-8afb-70d95e83f35b",
    "message": "Ingest job finished successfully for storeId: 11. ",
    "jobType": "SearchIndex".
  }
}

```

- Check **message** to observe at a high level what problems Ingest/NiFi has:
 - If the index build is stuck in a running state longer than usual, then the **message** lets you know which pipe that the index is stuck on.
 - If the index build fails in the middle, then the **message** lets you know the issues in the pipe causing the index failure.
- The API returns the status information in the form of one of the following status codes:
 - 1 = Failed.
 - 2 (Old version) or 4 = Ingest could not determine the status the Index is in from Elasticsearch.
 - -1 = Index is still running.
 - 0 = Indexing successful.



i. Access the storefront.

Once the deployment is completed successfully, you can get the ingress IP and hostname and add the required host entries to access the application.

Note: If you have the authorized domain and you can add that to the ingress yaml and refer the secret under `tls-secret` which would hold the crt and key file of your domain.

This will ensure the secured https call is made to the HCL Commerce application.

For example, to secure the Emerald store call, create the ingress with your host and add the secret at the bottom of yaml as displayed in the screenshot example below.

```

spec:
  rules:
    - host: demoqaauth.commerceftr.hclcx.com
      http:
        paths:
          - backend:
              service:
                name: demoqaauth.commerceftr.hclcx.com
                port: 80
  tls:
    - hosts:
        - demoqaauth.commerceftr.hclcx.com
      secretName: commerceftr
status:
  loadBalancer:
    ingress:

```

You can get further ingress details using the `kubectl get ing` command and then add the host entries in local hosts file.

```
tls-demoqaauth-cmc-ingress      <none>      cmcdemoqaauth.commerceftr.hclcx.com      a9162b3d89884476dbd0941a0e0b2397-b434097c5ab25fbc.elb.us-east-1.amazonaws.com      80, 443
86m
tls-demoqaauth-reactstore-ingress      <none>      demoqaauth.commerceftr.hclcx.com      a9162b3d89884476dbd0941a0e0b2397-b434097c5ab25fbc.elb.us-east-1.amazonaws.com      80, 443
130m
tls-demoqalive-cmc-ingress      <none>      cmcdemoqalive.commerceftr.hclcx.com      a9162b3d89884476dbd0941a0e0b2397-b434097c5ab25fbc.elb.us-east-1.amazonaws.com      80, 443
91m
tls-demoqalive-reactstore-ingress-live      <none>      demoqalive.commerceftr.hclcx.com      a9162b3d89884476dbd0941a0e0b2397-b434097c5ab25fbc.elb.us-east-1.amazonaws.com      80, 443
121m
```

To get the ingress external IP or alias domain name, use

`kubectl get svc -n ingress-nginx`

```
[ec2-user@ip-10-192-10-247 templates]$ kubectl get svc -n ingress-nginx
NAME                                TYPE                CLUSTER-IP      EXTERNAL-IP      PORT(S)          AGE
ingress-nginx-controller            LoadBalancer       172.20.201.170   a4e30e5bfd27c4b4db92324711e28389-4738be3d03d7bba0.elb.us-east-1.amazonaws.com      80:32504/TCP,443:31405/TCP      84s
ingress-nginx-controller-admission ClusterIP            172.20.199.37   <none>           443/TCP          84s
```

`nslookup a4e30e5bfd27c4b4db92324711e28389-4738be3d03d7bba0.elb.us-east-1.amazonaws.com`

```
[ec2-user@ip-10-192-10-247 templates]$ nslookup a4e30e5bfd27c4b4db92324711e28389-4738be3d03d7bba0.elb.us-east-1.amazonaws.com
Server:      10.192.0.2
Address:      10.192.0.2#53

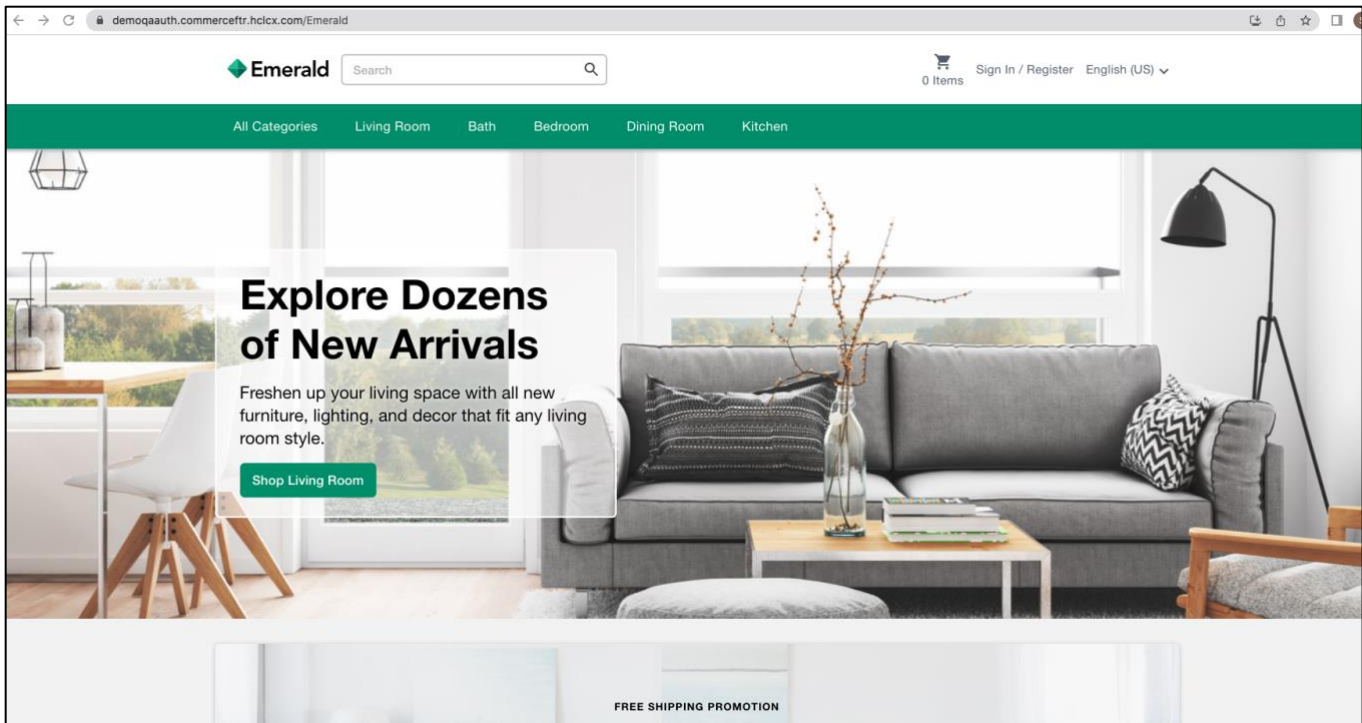
Non-authoritative answer:
Name:   a4e30e5bfd27c4b4db92324711e28389-4738be3d03d7bba0.elb.us-east-1.amazonaws.com
Address: 18.210.47.159
```

Below are some commerce applications (store, cmc) screenshots:

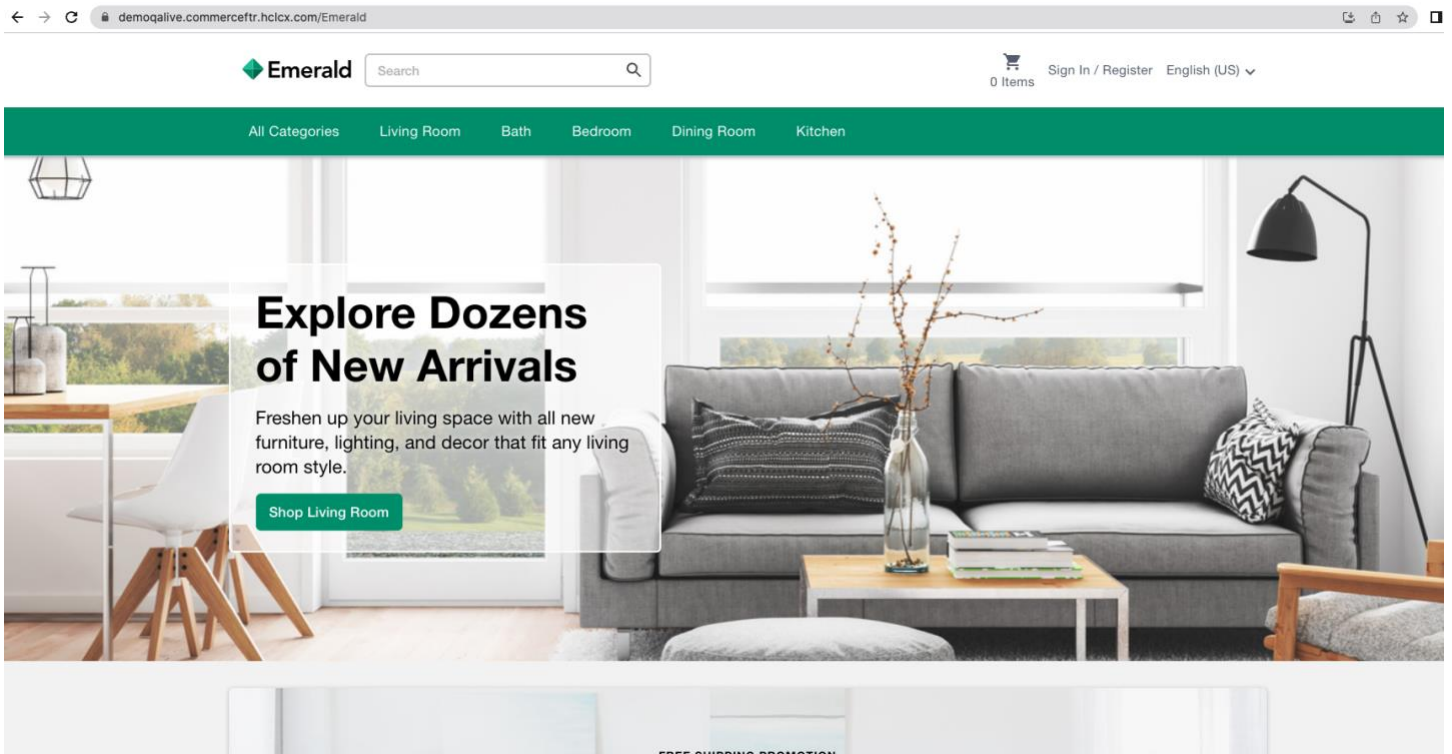
[Accessing the TLS enabled Storefronts in production environment:](#)

Emerald (B2C) auth instance:

<https://demoqalive.commerceftr.hclcx.com/Emerald>

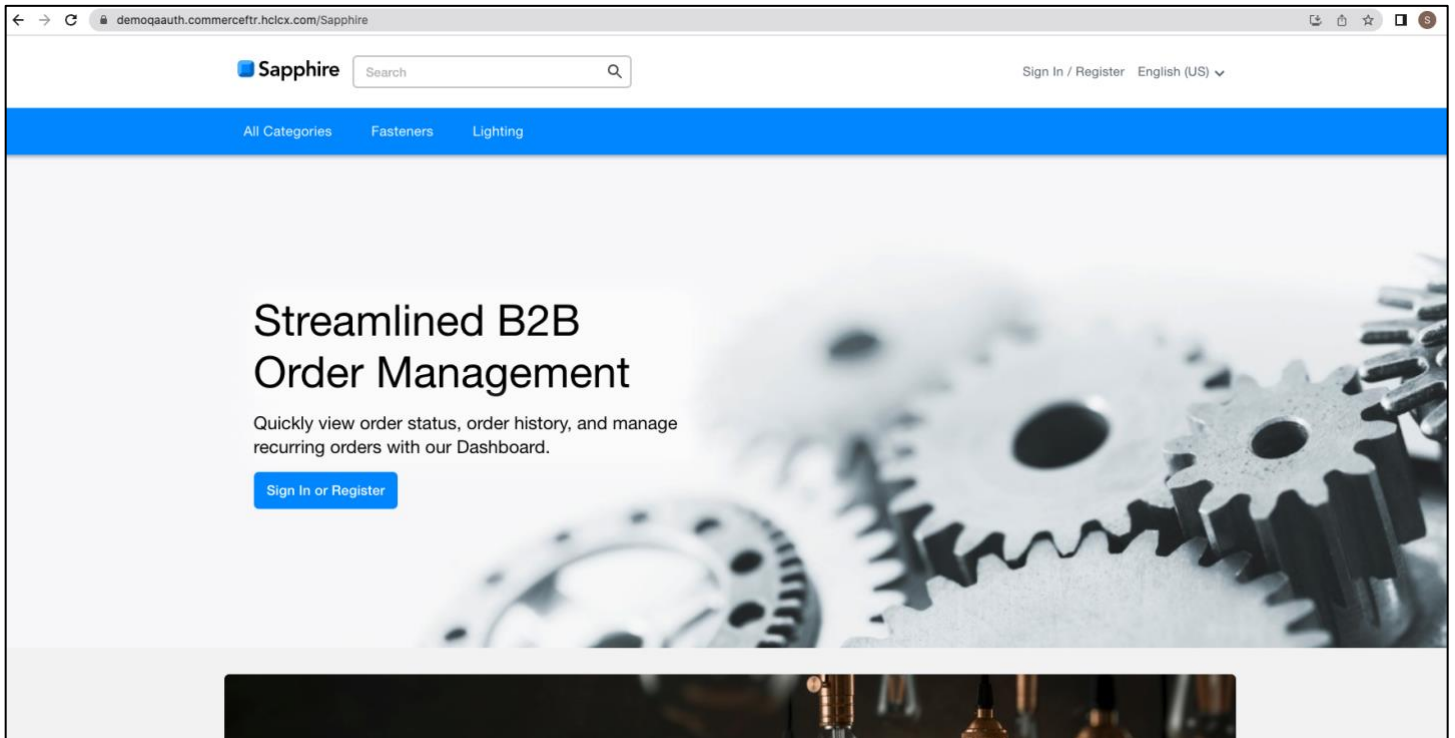


Emerald (B2C) Live instance:
<https://demoqalive.commercefr.hclcx.com/Emerald>



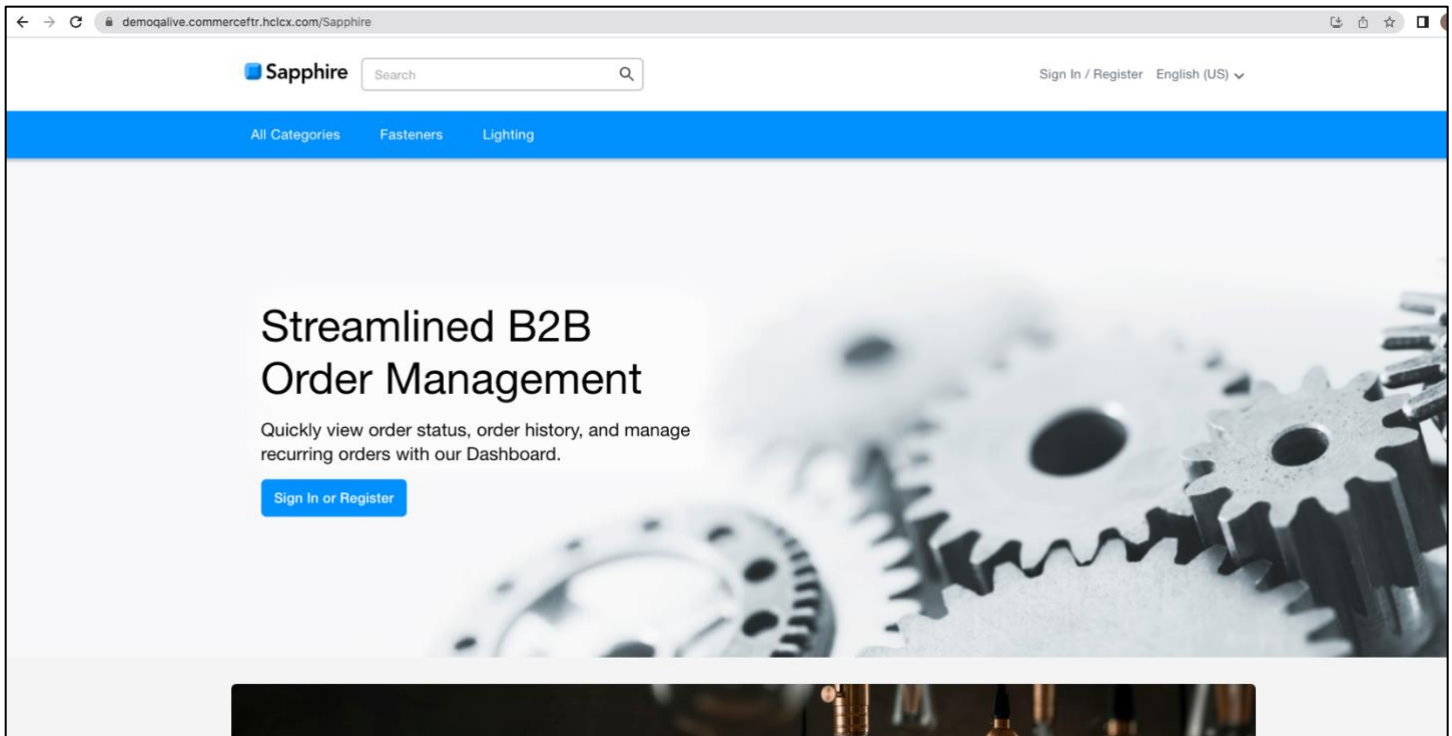
Sapphire (B2B) Auth instance:

<https://demoqaauth.commerceftr.hclcx.com/sapphire>



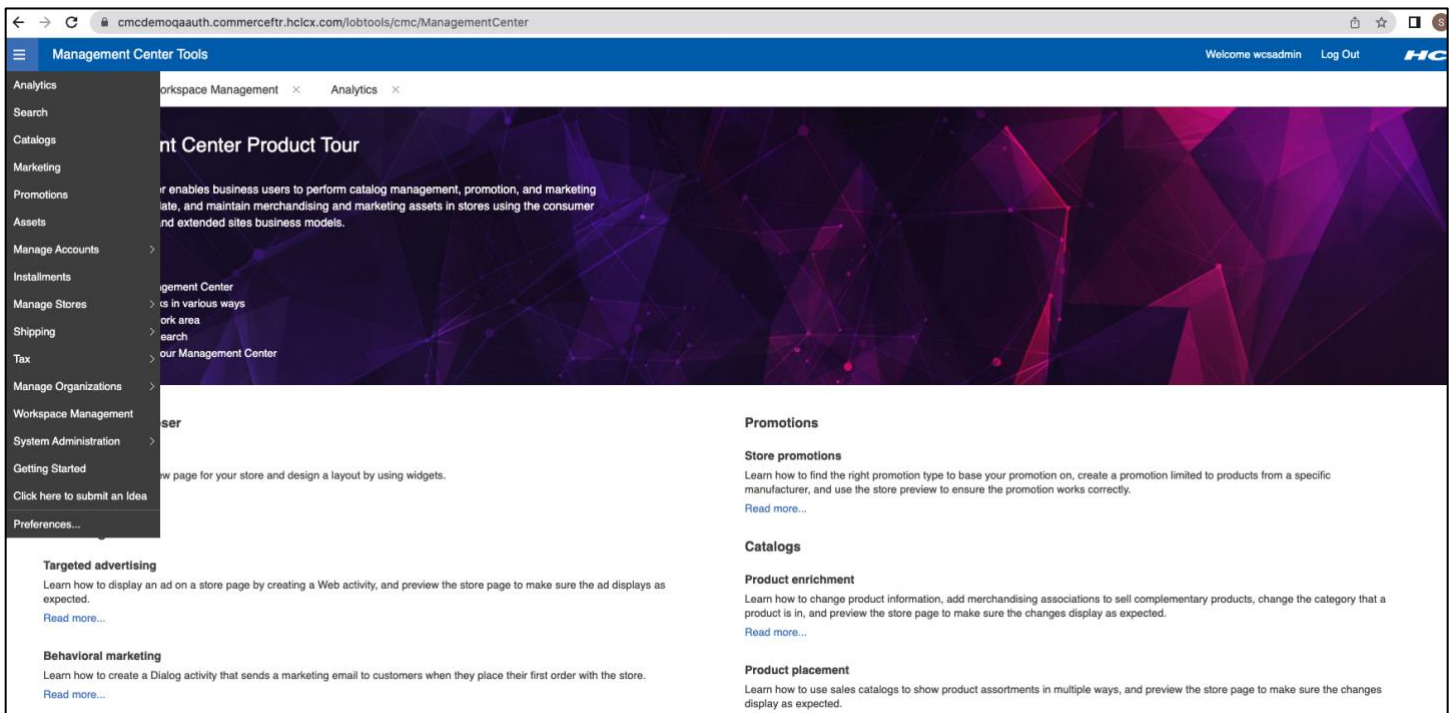
Sapphire (B2B) Live instance:

<https://demoqalive.commerceftr.hclcx.com/sapphire>

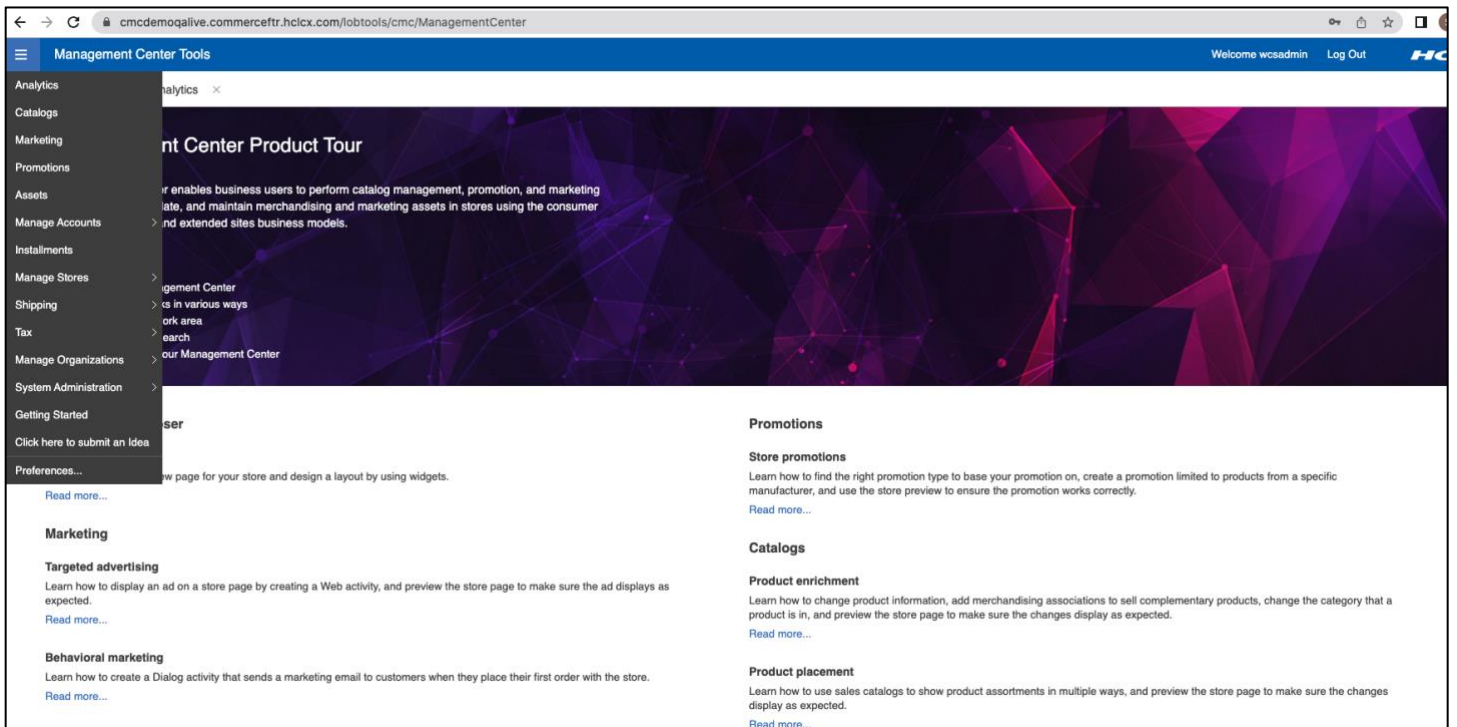


Management Center for HCL Commerce (with TLS enabled)

Management Center Auth:

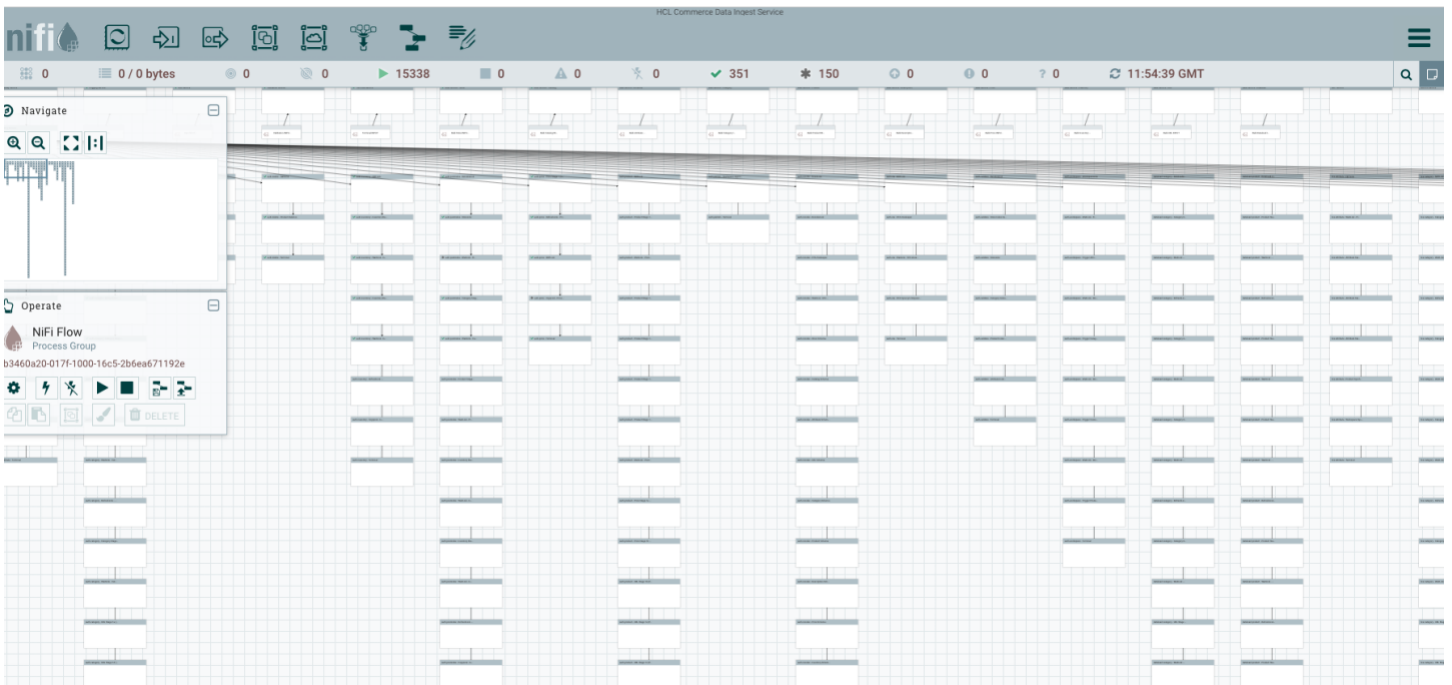


Management Center Live:



Nifi:

Note: Nifi is an internal search application, and it is recommended to be accessible only internally. Or, to check NiFi, you can port-forward to the NiFi pod.



HCL Commerce Solr-based search application deployment

The focus of this document is on the newer Elasticsearch-based solution that was introduced with HCL Commerce 9.1. However, can also use the HCL Commerce Helm Chart to deploy a Solr-based search environment with similar steps mentioned before. This was the search solution introduced in previous versions of HCL Commerce, and remains fully supported in HCL Commerce 9.1.

By setting the deployment parameter **common.searchEngine** to *Solr* in the *values.yaml* file, HCL Commerce is deployed with the Solr-based search engine configuration. In this deployment, it will deploy a Search Master into the *auth* group, and a Search Repeater and Search Slave into the *live* group.

It will not deploy the Elasticsearch-based search solution. In this configuration, the new React-based reference store can not be deployed, as this store solution requires Elasticsearch query services to function.

For more details, see [Deploying HCL Commerce Version 9.1 on Kubernetes](#) in the HCL Commerce documentation.

7.2. Maximizing Uptime & Availability (DAS-002)

HCL Commerce delivers Docker images and Helm Charts. Customers can deploy Commerce on multiple Amazon EKS clusters in multiple regions, and then connect applications hosted in Amazon EKS clusters in multiple AWS Regions over private network using [inter-region VPC peering](#) and employ a data replication strategy that fits recovery point (RPO) and recovery time (RTO) objectives. Customers can also use AWS Global Accelerator routing capabilities to simplify directing traffic to multiple AWS Regions whether you need to implement active-failover, active-active, or more complex scenarios.

HCL Commerce customers must set up DB2 or Oracle Database themselves, and should be encouraged to follow their database solution instructions to set up High Availability architecture. Commerce uses IBM WebSphere Application Server (tWAS) and Liberty server Datasource to connect to the database. Both TWAS and Liberty DataSource support high availability Data access to DB2 and Oracle.

7.3. Different Deployment Configurations (DAS-003)

HCL Commerce supports single-AZ, multi-AZ, or multi-region deployments. However, multi-region deployment is not recommended, because of the complexity of deployment and its negative performance impacts. A multi-AZ deployment is recommended, because it can ensure availability and stability for HCL Commerce during availability zone disruption, or planned system maintenance. For a single-AZ deployment, the HCL Commerce Helm Chart supports increasing replicas to prevent unexpected failures for a single node. Overall, for use in a production environment, HCL Commerce users should endeavor to set up the environment with a multi-AZ deployment model to achieve a balance between performance and availability.

7.4. Testing / Troubleshooting (DAS-004)

The first step to verify the HCL Commerce deployment is to check if all the Kubernetes pods are up and running properly. It requires about ten minutes for all the pods to be up and running in sequence depending on the capacity of your Kubernetes worker nodes. If the user has enabled auto search index creation for Elasticsearch, then should expect some extra time for the search index job to be completed successfully. Users can also trigger the build index job for either Elasticsearch-based or Solr-based search manually after all the Commerce services are up and running to verify them.

The next step for testing is to access to environments and perform some sanity tests, such as going to the storefront to complete a shopping flow or navigate to Commerce management center to create a promotion. Overall, users should try to access different Commerce URLs and make sure they are accessible and try some basic functionalities as a starting point for the testing process. See more instructions provided in steps 4, 5 and 6 in Help Center (<https://help.hcltechsw.com/commerce/9.1.0/install/tasks/tdeploykubern91-commerce.html>) to monitor the deployment and access the environment.

The following URLs can be used to access store and business tooling:

- An Aurora storefront: <https://commercehostname/wcs/shop/en/auroraesite>
- An Emerald storefront (The new React-based reference store): <https://commercehostname/Emerald>
- Management Center for HCL Commerce: <https://commercehostname/lobtools/cmc/ManagementCenter>

For further troubleshooting help, see [Troubleshooting](#) in the HCL Commerce documentation.

8. Health Check

8.1. Assess & Monitor Application Health (HLCH-001)

HCL Commerce 9.1 has extensive monitoring capabilities based on Prometheus-style metrics. The HCL Commerce pods make metrics available on internal endpoints that are consumed by Prometheus through a process known as scraping. HCL Commerce also provides a set of Grafana dashboards which are designed to bring attention to critical metrics, highlight problems, and assist users with the resolution of bottlenecks.

For users to utilize HCL Commerce native monitoring capabilities, the first step is to install a Prometheus-compatible monitoring system. Specifically, kube-prometheus-stack is the most popular and recommended option for installing Prometheus. It is an all-in-one install that includes Prometheus Operator (ServiceMonitor support), Grafana, and pre-configured instrumentations for Kubernetes monitoring.

For more details about the installation process, see the [Prometheus and Grafana Install](#) readme on the HCL-TECH-SOFTWARE github account.

Once Prometheus and Grafana are installed, users can import the HCL Commerce Grafana dashboards. Grafana dashboards can be installed individually by importing the .json files into the Grafana UI, or by using the Commerce Dashboards Helm Chart to define them all at once. Users can monitor a comprehensive set of metrics giving a detailed view into the performance and health of the servers.

For more information about metrics that are displayed in the provided dashboards, see the [HCL Commerce - Dashboards for Grafana](#) readme on the HCL-TECH-SOFTWARE github account.

9. Backup and Recovery

9.1. Automated Backup of Necessary Components (BAR-001)

HCL Commerce does not provide any automated tools or scripts for backups. However, several suggestions for backup and recovery are provided:

- Because HCL Commerce applications are stateless, and depend on the database for the recovery, it is strongly recommended to follow the database platform documentation to back up the entire database.
- Customers must always back up unique configurations and customizations. All configuration information and customizations should be recorded so that they can be used to re-create a duplicate test or production environment in the case of an emergency or failure. In addition, all source and deployment related codes should be backed up as well for recovery purposes.

9.2. Restoring Data from a Backup (BAR-002)

Refer to the documentation for your database platform on proper procedures for restoring backup files.

To restore the backup of your HCL Commerce database, see the following:

- [Restore overview](#) in the IBM Db2 documentation.
- [Database Backup and Recovery User's Guide](#) in the Oracle Database documentation.

9.3. Recovery from Failure (BAR-003)

HCL Commerce applications are stateless. All data is stored in the database. When an application failure occurs, health check detects the failure and redeploys a new pod. The new pod will read data from database and recover the application state.

Therefore, HCL Commerce depends on the database to work. Customer must follow their database documentation to setup High Availability (HA) mode and correctly configure backup and recovery policies.

9.4. Recovery from Availability Zone Failure (BAR-004)

HCL Commerce does support multi-AZ or multi-region deployment. However, a multi-region deployment is not recommended due to the complexity of the deployment, and performance implications.

A multi-region or multi-AZ Commerce deployment can achieve the greatest possible fault tolerance and stability, and it can prevent disruption caused by availability zone failure. When there is a failure in one zone, the EKS cluster can detect it and switch to a working zone.

In addition, as mentioned before, HCL Commerce applications are stateless, and depend on the database for any recovery. An availability zone failure does not impact the data stored in database. For these reasons, recovery is achievable when the availability zone is back, even if only a single availability zone is utilized.

9.5. Manage Service Limits for Disaster Recovery (BAR-005)

If there are enough resources for deployment as mentioned in section 6.1, customers can achieve disaster recovery based on the database, configurations, customizations, and source code that they have backed up.

Customers must regularly view and manage the quotas for AWS services using AWS Service Quotas dashboard.

9.6. Recovery Time Objective (RTO) and Recovery Point Objective (RPO) (BAR-006)

An estimated time for RTO and RPO cannot be estimated since these values are different for each customer.

These values depend on the individual operations strategy of each customer, and depend on factors such as the size and complexity of environments, customizations, and the database backup strategy that are in place.

10. Routine Maintenance

10.1. Rotating Programmatic Credentials and Crypto Keys (RM-001)

Sensitive data in the HCL Commerce database is encrypted using the seller specified merchant key.

This encryption key can be changed using the [*MigrateEncryptedInfo* utility](#).

10.2. Software Patches and Upgrades (RM-002)

Updates for HCL Commerce production environments are delivered as Docker images.

To update a production environment, you must download the new images, and rebuild them as custom images to include your custom code. Then, deploy the new custom images.

For more information, see [Deploying updated Docker images with Kubernetes](#) in the HCL Commerce documentation.

10.3. Managing Licenses (RM-003)

HCL Commerce manages license entitlements through the HCL Flexnet portal.

Only entitled customers can legally download HCL Commerce binaries.

When deploying HCL Commerce, customers must set the **ACCEPT LICENSE** environment variable to *true* in the Helm Chart `values.yaml` configuration file.

The associated license files are included in each of the HCL Commerce Docker images.

10.4. Managing AWS Service Limits

AWS users (customers) must manage their own Service Limits and Quotas in the AWS Service Quotas Console. For more information, see [How do I manage my AWS service quotas?](#) in the AWS documentation.

11. Emergency Maintenance

11.1. Handling Fault Conditions (EMER-001)

Customers must contact HCL and open a case to describe the fault conditions that they are facing. The engineering team investigates any reported issues and performs one of the following approaches to resolving product issues:

- Providing an updated Docker image that includes the required product fix.
- Providing a patch with instructions on how to apply it, to resolve the environment issue directly.

11.2. Recovering the Software (EMER-002)

Customers must back up their database, configurations, customizations, and source code.

With these backups, customers can recover their HCL Commerce environment to the point at which they encountered issued.

12. Support

12.1. How to Receive Support (SUP-001)

[HCL Software Support portal](#) is the central hub for HCL product and customer support.

For HCL Commerce product support, you must register, and gain access based on your organization's entitlements. For more information, see [How to register to HCL Software portals - eCommerce, License & Download, Customer Support](#).

Details about the Support processes are found in the [HCL Support Guide](#).

If urgent support is required, and a customer cannot login (for example, forgotten password), they can reach out to our Customer Assistance Center (CAC) team by following the Chat, Phone or Guest case links in the "Contact us" portlet on the main [HCL Software Support](#).

Instructions for creating a support case after registering can be found on [How to create a HCL support case](#).

12.2. Technical Support Tiers (SUP-002)

There is a general standard support tier available to all current customers.

Customers can upgrade to [Premium Software Support](#) for an additional fee, which be customized to some extent.

12.3. Support Tiers & SLAs (SUP-003)

Priority	Business Impact	Definition	Initial Response Time
1	Critical	<p>An issue with the product or service that impacts critical business functionality in your production environment and requires immediate attention. No procedural workaround is available.</p> <p><i>Note: 24x7 assistance requires you to ensure availability of necessary technical resources to assist HCL Support with any requests related to problem determination for the case.</i></p>	1 Hour (Premium) 2 Hour (Standard)
2	Significant	An issue with the product or service that significantly limits functionality and requires prompt attention. A business deadline may be in jeopardy.	2 Business Hours
3	Minor	An issue or question related to the product or service that has low impact to business operations.	2 Business Hours
4	None	A question (possibly non-technical) that has no impact to your business operations.	2 Business Hours