Marketplace docs

HCL Software

Table of Contents

Welcome to HCLSoftware Marketplace

Quick HCLSoftware Marketplace Introduction

Welcome to HCLSoftware Marketplace, the easy way to go cloud native with HCLSoftware and partner products. Marketplace lets you try out these products in our sandbox environment, and helps you get them running in containers, orchestrated by Kubernetes, in the cloud of your choice. Kubernetes opens up the benefits of cloud native technology, including lightning-fast deployment and updates, built-in resilience, and elastic scaling.

Explore the Catalog

HCLSoftware Marketplace has a catalog of ready-to-use HCLSoftware Products, Business Solutions, and partner content for easy deployment to a cloud native environment. Most of the entries in the catalog are packaged as container images and Helm charts, for simple installation and configuration in Kubernetes. A few catalog items link to multi-tenant cloud (SaaS) offerings.

To view the entire catalog, select **Catalog** on the header menu. Click on the card to learn more about the product or Business Solution. The Catalog Details page shows important information and documentation for using that product. Search the catalog by typing keywords in the Search Box or by using the filters.

Try out a Business Solution or Product in our Trial Kubernetes Sandbox

Launching a sandbox is quick and easy. From the catalog page:

1. Click **LAUNCH IT** on the catalog card or from the Catalog Details page to directly startup a sandbox.

2. Once deployed to the sandbox, you can view the solution via the Sandbox Links. **OPEN**.

3. Copy the login credentials listed below the button to access the Sandbox Links.

4. If you would like to see more technical information related to Kubernetes about the deployed solution, click to open the **Sandbox Admin Console** button. Copy the login credentials listed below the button to access the Sandbox Admin Console.

Download a Product for Solution Install

HCLSoftware Marketplace is designed to allow solutions to deploy to any Kubernetes environment, giving you several options: open source Kubernetes for a self-managed environment, public cloud vendors like Google Kubernetes Engine or Amazon Elastic Kubernetes Service, or private cloud platforms that include Kubernetes capability within their data center or cloud account. The Marketplace deployment package is a Helm chart that we call a Marketplace Solution (a .tgz file).

To download a product, click on the **Download Helm Chart** (download icon) found on the top right header of the Catalog Details page, or from the catalog card itself.

Download Helm Chart

Detailed information for installing Solutions can be found in the following Marketplace Guides:

- Installing Solutions: Getting Started with Solutions in a Google Cloud Platform Trial Account Tutorial
- Installing Solutions: Overview for Kubernetes Administrators
- Installing Solutions: ss Instructions
- Applying your Own Domain Name and SSL Cert
- Supported Kubernetes Environments

While Marketplace does provide a sandbox (Kubernetes cluster) for limited testing of solutions, it is not a deployment platform. You can read more in the Marketplace Guide entitled Supported Kubernetes Environments.

Marketplace Access Requests

Access to HCLSoftware Marketplace and our container registry are controlled through your HCLSoftware ID. If you do not already have one, an HCLSoftware ID is created for you upon submittal and approval of the access request form. There are two levels of access available:

- HCLSoftware Marketplace site at https://marketplace.hcl-software.com - run HCLSoftware products in the provided cloud native sandbox
- HCLSoftware container registry at https://hclcr.io - download container images on any products for which you have a license

Gaining Access:

To obtain access to HCLSoftware Marketplace, please fill out the form at this link. Access is granted when **one of the two following criteria is met:**

1. Belong to an organization (Customer or Business Partner) that owns an active HCLSoftware entitlement. Please register with your business email so this relationship can be verified. If you register as a Customer or Business partner, but your email cannot be verified, you will be contacted for further details.
2. Enter an access code that was provided to you by an HCLSoftware Sales representative. If you are not yet affiliated as a Customer or Business Partner and you need an access code, fill out the form and you will be contacted by the HCLSoftware Sales. Once you obtain the access code, return to the Request Access landing page and fill out the form again, inputting your assigned access code into the corresponding field. When the form is submitted, you will be granted access to Marketplace immediately and sent an email containing login information.

Guides

Marketplace Guides

Welcome to the HCLSoftware Marketplace Guide section of our Doc site. Here, you will find technical, step-by-step instructions for how to get the most out of Marketplace. These guides are intended to address specific resources within Marketplace and provide in-depth instructional support on a range of topics. For shorter Question & Answer format information, please visit our FAQ Page. If there are any topics you believe necessitate a Marketplace Guide, please send your suggestions to this email address.

Solution & Catalog Overview

Marketplace Catalog

Our Marketplace Catalog has two categories:

- **Business Solutions** integrate one or more HCLSoftware products, or partner products, with configuration and data, and using step-by-step instructions, they illustrate how the products can be used to solve business challenges
- **Products** are complete HCLSoftware product offerings, available to try and download in a cloud-native form factor

Most catalog assets have Helm charts which can be deployed in our trial sandbox, and some can be downloaded for installation in your own Kubernetes cluster. A few catalog entries provide links to hosted cloud (SaaS) services.

Marketplace Business Solutions

Business Solutions help you to experience product capabilities, with well-documented scenarios you can try out in the product itself. **Demo Packs** contain demo assets like data, configuration, and applications, that are installed with a product to provide a richer demo experience. Some demo packs provide integrations between two or more products.

Navigating the Catalog

Our Marketplace Catalog is listed from Business Solutions to Products, descending down the page. Use filters or keywords in the search box to find the items that best fit your needs.

When you click on a card from the catalog, a populated details page will tell you important information about what the Business Solution or product includes.

Trying out catalog items in the Sandbox

Most catalog items have Helm charts that can be installed into the Marketplace Sandbox, a Kubernetes cluster provided for you to try out our software for a limited time.

Note: This is not a production environment and you should not upload any sensitive data to the products running in the Sandbox. Most products can also be installed in your own Kubernetes environment, by downloading the solution Helm chart. However, some catalog assets are for sandbox use only, and solutions that include such items will not be available for download. Check the documentation for the individual items for any restrictions on supported environments.

Requesting a Quote

Some catalog tiles have information about requesting a product quote. Click 'Request Quote' from the Catalog Detail page, if applicable, fill out the form and submit. An HCLSoftware team member will reach out to further discuss your request. The quote form is a preliminary inquiry; not yet a final purchase.

Placing an Online Order

Some products have an option to place an online order. Click 'Buy' from the tile or from the Catalog Detail page. Clicking on 'Buy' will navigate you to the purchase description page. The purchase description page explains what comes with the order. To place an order, fill out the form, and Continue with purchase. You will receive a confirmation email with instructions on how to submit your final payment.

Solution Overview

The Helm charts provided through the Marketplace Catalog represent Marketplace Solutions. A Solution is a Kubernetes application that includes one or more HCLSoftware products and a set of common services. Each product and common service is managed through its own Helm chart, and these charts are packaged as peer child charts into the parent Solution chart. This provides a convenient packaging of the desired products, with essential services, such as ingress and monitoring, resulting in a self-contained deployment package that is portable across different Kubernetes environments.

All services are managed through their applicable Helm charts. When a solution is installed into Kubernetes, the services run in a private network, communicating with each other using internal IP addresses that are not accessible by services outside the solution’s network. All access from outside the solution, such as from your browser or local application, is made through the single external IP address for the solution. This external IP address connects to the routing component, or ingress controller. Marketplace uses Ambassador or Emissary as the ingress controller for each solution. It provides a useful control point for all traffic coming into the solution network, as in the example of applying access control.

Marketplace Common Services

These lightweight services are pre-integrated with our products to provide self-contained packages that are ready to deploy into any Kubernetes environment. The common services are fully supported for use with our products, under your existing product license. The intent is to help you avoid (or at least minimize) dependencies on services provided by your deployment platform, which may lock you in to a specific cloud vendor.

Sandbox Admin Console

The Sandbox Admin Console application provides a simplified solution administration experience, including:

- Kubernetes cluster info, which is filtered to only show the information for the solution resources; this is useful if you are new to Kubernetes or are operating in a large cluster, but are only concerned with the solution application
- URLs for the services and products selected from the Marketplace Catalog - these include links to product GUIs, REST API swagger UIs, and base URLs for REST APIs
- Links to other common services in the solution:
    - the Grafana monitoring dashboard
    - the KeyCloak GUI to manage users, passwords, and access controls
    - the Prometheus GUI for low-level access to monitoring data
- Access to logs for all pods in the solution

If you have launched the Solution in the Marketplace Sandbox, a link to 'View Sandbox Admin Console' is displayed in the Sandbox Information view.

If you have installed the Solution in your own environment, the URL for the Sandbox Admin Console will be displayed at completion of the Helm install. It can also be constructed using the external-IP assigned to the Ambassador or Emissary services as follows:

```
https://Marketplace-console.{external-IP}/
```

Refer to the Initial Login Credentials guide for your Marketplace Sandbox login credentials.

Access Control Service (ACS)

ACS provides both authentication and authorization controls for traffic accessing the solution external IP. When included in a solution, the ACS is registered as the authentication service for Ambassador or Emissary and by default, and it will be called to examine every request. The following default users and passwords are created for each new solution:

Userid	**Default password**	**Access**
user	Refer to Initial Login Credentials	catalog services
sol-admin	Refer to Initial Login Credentials	all services (including Sandbox Admin Console)

Users and passwords can be managed through the Keycloak component that is included in ACS. A link to the Keycloak GUI is available in the Sandbox Admin Console application as follows: https://Marketplace-kc.{solution-ip}/auth/admin/MarketplaceSolution/console.

Monitoring Dashboard

The Monitoring service includes Prometheus, to gather and store monitoring data, and Grafana, to display that data in visual dashboards.

Solution Install

Installing Solutions Overview for Kubernetes Administrators

These instructions are designed for experienced Kubernetes users and list the steps for solution install.

- If you don't have access to Kubernetes, follow the steps found in the GCP Trial Account, and watch our Getting Started in Marketplace video to view where to download a chart in Marketplace.
- If you need more details on any step, see the Step-by-Step Solution Install with Kubernetes guide.
- Supported versions of Kubernetes, Helm and Cert-Manager are listed in the Kubernetes Environments & Requirements guide.

You will need the following:

- Cert-Manager must be installed in the target cluster
- Each solution must be in a separate namespace
- Create a pull secret using your HCL repository credentials

Kubernetes Environments & Requirements

There are many options for utilizing your own Kubernetes environment:

- **Open source** Kubernetes can be used for a self-managed environment
- Most **public cloud** vendors provide supported Kubernetes installations, for example Google Kubernetes Engine and Amazon Elastic Kubernetes Service
- Some **private cloud** platforms allow Kubernetes to run in your data center or public cloud account

HCLSoftware Marketplace is designed to allow solutions to run in any Kubernetes environment. Solution dependencies relate to core Kubernetes only, with no use of vendor-specific services. With a growing number of Kubernetes vendors, however, solutions are tested on specific sets as listed below. If you run Marketplace Solutions in a different environment, we do our best to address any issues but may need to work with you directly to debug problems that appear to be specific to your environment.

Please note that individual products and services in the Marketplace Catalog may document their own support policies for different Kubernetes versions and providers.

Tested Kubernetes Environments

- Google Kubernetes Engine
- Amazon Elastic Kubernetes Service
- Azure Kubernetes Service
- Red Hat OpenShift Container Platform
    - OpenShift security policies mean that most Helm charts will not work out of the box. Please check the documentation for the individual catalog items to see if instructions are provided for installation on OpenShift.

Supported Kubernetes Versions

The Kubernetes project couples frequent releases with a strict compatibility policy, to allow rapid innovation of the platform but to minimize impacts of those updates to existing applications. Most update issues can be avoided by using only generally available (stable) APIs, but even those may sometimes be deprecated and eventually removed. Solutions built on the current version of Marketplace are supported when run on the following tested Kubernetes versions:

- 1.29
- 1.31

- 1.32

Cluster Requirements and Limitations

- Cert-Manager must be installed in the cluster; version 0.15.1 or later
- Each solution must be installed in a separate namespace
- The Helm 3 client must be used to install solutions; supported version: 3.3.0 or later
- When the cluster is k8s v1.23 or newer, the minimum version of Cert-Manager is 1.8.2 and the minimum version of Helm is 3.10.x

Step-by-Step Solution Install with Kubernetes

These instructions are designed for those who have an existing Kubernetes environment, and detail the steps for solution install. Supported versions of Kubernetes, Helm, and Cert-Manager are listed in the Kubernetes Environments & Requirements guide. If you need more details on any step, see the Getting Started with Solutions in a Google Cloud Platform Trial Account Tutorial guide, which also provides a script that automates most of the setup.

You will need permissions to install to the Kubernetes cluster. Use of a cluster that supports dynamic PV is recommended.

To install your solution:

1. Install a supported version of Helm 3

   ```
   wget https://hclcr.io/files/sofy/scripts/get-helm3.sh &&
   source get-helm3.sh
   ```

2. Install Cert-Manager in your cluster

   - **Note:** *When the cluster is k8s v1.23 or newer, the minimum version of Cert-Manager is 1.8.2 and the minimum version of Helm is 3.10.x*

   - **How to Upgrade Cert-Manager**:

     1. Untar your current solution.
     2. Navigate to the templates folder at *solution/templates*.
     3. Delete all the cert-manager resource files associated to the Marketplace Chart, including those in the specific Product chart. (An example may be *cert-ca-issuer.yaml, cert-issuer.yaml, certificate-ca.yaml, and certificate.yaml*.)
     4. Next, upgrade your release with this new chart. This will cause Helm to remove the resources from step 3.

    5. Uninstall cert-manager using **helm uninstall cert-manager -n cert-manager**
    6. Reinstall cert manager (make sure you have jetstack in your helm repos): **helm install cert-manager jetstack/cert-manager --namespace cert-manager --create-namespace --version v1.8.2 --set installCRDs=true**
    7. Download your upgraded solution.
    8. Upgrade helm release using the new solution.
    9. Verify if your certificates are working as expected.

3. Installation of Emissary Ingress Custom Resource Definitions (CRDs)

    - The below commands install Emissary-ingress CRDs in the emissary-system namespace
    - Commands to install CRDs:

    ```
    kubectl apply -f https://app.getambassador.io/yaml/
    emissary/3.9.1/emissary-crds.yaml

    kubectl wait --timeout=90s --for=condition=available
    deployment emissary-apiext -n emissary-system
    ```

4. Create a namespace for your solution (optional)

    - Each solution must be installed in a separate namespace; if you install into the default namespace you can omit the --namespace parameters on the remaining commands

    ```
    kubectl create namespace [name]
    ```

5. Create an image pull secret in the solution namespace

    ```
    kubectl create secret docker-registry [secret-name] --
    docker-server=hclcr.io --docker-username=[Marketplace
    userid] --docker-password=[CLI secret] --
    namespace=[solution namespace]
    ```

    - To set your Harbor CLI secret, login to the container registry at hclcr.io with your HCL/Marketplace credentials, selecting LOGIN VIA OIDC PROVIDER. To get access to the container registry, please see [Accessing Marketplace]().
    - Open your User Profile (in the top right corner, click on the dropdown for your username) and enter a secret string of your choice
    - Use this string as the CLI secret in the command shown above

6. Use Helm to install the solution chart

    - The solution chart can be downloaded from the Marketplace application, on the Catalog Details screen

7. Summary of the variables above:

    - **[release-name]** A Helm release name that is not already used

    - **[file-name]** The file name of your downloaded solution
    - **[solution-namespace]** The namespace you are installing your solution into
    - **[secret-name]** The name of your pull secret created in Step #4

8. You may need to set additional value overrides for the specific contents of your solution, see the documentation for those products and services in the Marketplace Catalog.

9. **Default value for solutions generated in HCLSoftware Marketplace is:**

   ```
   --set global.hclImageRegistry="hclcr.io/Marketplace"
   ```

10. That can be overridden to point to another registry (if you are hosting images locally), for example:

    ```
    --set global.hclImageRegistry=registry.io/project
    ```

11. If you are using an AWS EKS cluster, add the below annotation to the solution's Ambassador or Emissary service so it can receive an external IP:

    ```
    kubectl annotate svc [solution-ambassador-or-emissary-svc]
    -n [solution-namespace] service.beta.kubernetes.io/aws-
    load-balancer-internal=0.0.0.0/0
    ```

12. **Note:** The process of assigning an external LoadBalancer IP address to the annotated service may take a few minutes.

    ```
    kubectl get cm [releasename]-domain -o yaml
    ```

    The output will be similar to the following. You will need the value of the HOST field:

    ```
    apiVersion: v1
    data:
        HOST: 10.190.16.62.nip.io
        HOST_PROTOCOL: https
    kind: ConfigMap
    ```

13. Access the Sandbox Admin Console

    - Once the install has completed and all pods are ready, enter the HOST into this link to access the Sandbox Admin Console app in your browser: https://Marketplace-console.${HOST}.

    - You will see some warnings about the certificate used in the solution; it is safe to accept these and proceed to the Sandbox Admin Console application.

- Log in to the application using the initial password for the 'sol-admin' user that is generated at solution install, and stored in a Kubernetes secret with the name:

  {{ .Release.Namespace }} {{ .Release.Name }}-acs-default-credentials.

- For example, if you installed your solution using the release name 'gcp1' into the default namespace, you would view the password using the below command line:

  ```
  kubectl get secret gcp1-acs-default-credentials -o
  json | jq -r ".data.admin" | base64 --decode ; echo
  ```

- The Sandbox Admin Console provides information about all parts of the solution, as well as links to the home pages of the included products and services.

- The GCP Trial Install guide provides detail on viewing the status of the install process and various troubleshooting tips.

Installation Tutorials

GCP Trial Account Tutorial

These instructions are designed for those new to Kubernetes, and have simplified steps to:

- Create a trial account in Google Cloud Platform (GCP)
- Create a Kubernetes cluster using Google Kubernetes Environment (GKE), in the GCP Trial Account
- Prepare the cluster for install of a Marketplace Solution
- Install and access the Marketplace Solution in the cluster

You can install more than one solution in a cluster, if there are sufficient resources, but each solution must be installed in a separate namespace. The steps shown below will install a solution into the cluster's default namespace. To repeat the installation with additional solutions, rerun the setup and specify a different namespace, then install the solution into the new namespace.

You will need the following to get started:

- A valid credit card (Google states that you will not be charged for the trial account)
- A Solution chart downloaded from Marketplace to your local file system; click on the **Download Helm Chart** (the download icon) found

on the top right header of the Catalog Details page, or from the catalog card itself.
- Your CLI secret for the HCL docker registry; instructions to obtain your CLI secret will be provided further down the documentation when it is needed

Note: If you already have a GCP Account, skip to section II.

I. Create a GCP Trial Account

1. Create a Gmail account (unless you want to use an existing account):

2. Signup for a new Gmail account

3. Create a Google GCP Trial Account: https://cloud.google.com/gcp/

4. Click the **Get started for free** button
5. Enter the email of existing account or account created earlier

Note: You will be asked to provide credit card information. Google states that it will not be charged unless you explicitly upgrade from the free trial to a paid account.

II. Sign-In to GCP, Create a Top-Level Project and a New Kubernetes Cluster

1. Login here to the GCP console using your account information

    - Once logged in, you should land in the **Kubernetes Engine > Cluster** view
    - You will be prompted to create a project

2. Select **Create Project** to build your top-level GCP project

    - No organization is required

3. Select **Create Cluster**

4. On the left-hand side, click on **Cluster basics**
5. Name your cluster
6. Use Zonal clusters with the default version of GKE

Note: Initially we'd recommend using the static version (requires manual updates) and not the release channel.

1. Next, on the left-hand side, click on **Node Pools** then select **default-pool**

    - By default, the node pool will have 3 nodes - we recommend to modify this to 2 nodes. Select size number of nodes: 2

2. Next, size you nodes according to your solution; select **Nodes** within the **default-pool**
3. The Catalog Details page includes estimated resource needs for the solution
4. Select Machine Type
    - For example, for a solution that requires 6 vCPU and 13 GB memory, you could select e2-standard-4 (2 nodes of 4 vCPU, 16 GB memory)
5. Click **Create** - your cluster should take around 3-5 minutes to be ready

---

III. Connect to your Cluster Using Google Cloud Shell

1. In the Kubernetes Clusters view, click **Connect** next to your newly created cluster
2. Select the button to **Run in Cloud Shell**
3. After accepting a one-time prompt, the shell will be launched with your first command "gcloud container..." pre-typed
4. Click **Enter** to execute this command which connects kubectl to your cluster
    - Verify you are connected to your cluster with the following command that should show more than a dozen pods already running in in the kube-system namespace in your cluster:

      ```
      kubectl get pods --all-namespaces
      ```
5. The minimum version of Helm for HCLSoftware Marketplace is documented in Supported Kubernetes Environments **Versions / Cluster Requirements and Limitations**. To install a supported version of Helm, please run the following command:

   ```
   wget https://hclcr.io/files/Marketplace/scripts/get-helm3.sh && source get-helm3.sh
   ```

Note: If you have used CloudShell in GCP previously, your CloudShell may still be using an older version of Helm. Please check your version of Helm running the below command:

```
helm version
```

If your Helm is below our supported version, you can reset your cloud shell and start our instructions again.

IV. Prepare the Cluster for Running Marketplace Solutions

This step will install Cert-Manager in the cluster, and create an 'image pull secret' in the namespace that allows access to the container registry where the HCL images are held.

Use this command in your google cloud shell to download and run the setup script:

```
wget https://hclcr.io/files/Marketplace/scripts/gcp-trial-
setup-harbor.sh && source gcp-trial-setup-harbor.sh
```

- You will be prompted to enter your username and CLI secret. To obtain this, follow the steps below:
    - Log into the container registry at https://hclcr.io with the **LOGIN VIA OIDC PROVIDER** button using your HCL/Marketplace ID credentials; if you need to create a username in the registry, it is recommended that you use your email address; if you need access to the registry, see Accessing Marketplace
    - In the top right corner, click on the dropdown for your username to get to your “User Profile”
    - From your User Profile you can copy the pre-generated CLI secret, or you can enter a secret of your choice; we recommend you enter a string that you will remember, to avoid returning to the registry each time you need the CLI secret
    - Note: This script does take a few minutes and has a bit of a pause when installing Cert-Manager

---

V. Install a Marketplace Solution

Once the above steps have been completed, all the required prerequisites will be installed. Now you are ready to install a Marketplace Solution.

Note: In the commands below, the --namespace flag is only required if you are not using the default namespace; it is included here to help if you use a non-default namespace.

1. Upload your solution chart to the Cloud Shell.
    - From the three-dot menu, click **Upload File** and navigate to the chart in your local file system

   Note: Upload File does not overwrite existing files in your cloud shell filesystem, so if you modify your solution and upload a new copy, be sure to delete the old file first. You can use the 'ls' command to list files and 'rm *filename*' to delete a file.

2. Install your solution as follows: A Helm install requires a release-name, which you can choose. If you don’t specify one you must include the --generate-name flag:

   Additional value overrides can be added to the Helm install command as needed (for example if you have used a non-default name for the image pull secret):

Override	**Command Line Argument**
Custom ImagePullSecret name	--set global.MarketplaceImagePullSecret={secret-name}
Any other value overrides	--set {name}={value}

   Now you can use kubectl or Helm commands to manage your deployed solution generated by Marketplace.

3. Monitor your solution pods to determine when the installation has completed and the pods are ready.

   ```
   kubectl get pods --namespace default
   ```

   The solution will be ready to access when all pods are in *Running* or *Completed* state, and the *Running* pods are all **READY**. For example, the output should be similar to this, where the **READY** column indicates when the running pods are ready to use. Pods that have completed are used to initialize other services and will not be in ready state:

```
   NAME                                                READY
STATUS      RESTARTS   AGE
   gcp1-access-control-service-5759f5fdbd-srd4j        1/1
Running     0          74m
   gcp1-acs-kc-postgresql-0                            1/1
Running     0          74m
   gcp1-alexgcp1-kube-state-metrics-6d775b968b-kksjj   1/1
Running     0          74m
   gcp1-ambassador-88b456cbd-l7vph                     1/1
Running     0          74m
   gcp1-ambassador-88b456cbd-lj94l                     1/1
Running     0          74m
   gcp1-ambassador-88b456cbd-z2g6f                     1/1
Running     0          74m
   gcp1-anchor-657c5c5569-zq5c7                        1/1
Running     0          74m
   gcp1-grafana-769b8f7bb4-cck27                       2/2
Running     0          74m
   gcp1-grafana-job-jxjqw                              0/1
Completed   1          74m
   gcp1-keycloak-0                                     1/1
Running     0          74m
   gcp1-openldap-5f5866945b-ng6ft                      1/1
Running     0          74m
   gcp1-product-design-mongo-bdf84fb6f-fdv62           1/1
Running     0          74m
```

```
   gcp1-product-design-redis-master-0                  1/1
Running     0          74m
   gcp1-product-designer-client-5dc979df4f-2bwqk       1/1
Running     0          74m
   gcp1-product-designer-server-7f5b59f4f4-bfllh       1/1
Running     0          74m
   gcp1-product-runtime-c4d66cc54-qjhqf                1/1
Running     1          74m
   gcp1-prometheus-server-757bd5c746-fq7t4             2/2
Running     0          74m
   gcp1-snoop-788f87594f-mkvz4                         1/1
Running     0          74m
   gcp1-Marketplace-console-d66cf776c-494xc                   1/1
Running     0          74m
   gcp1-solution-controller-7c8bcfd59f-cktvq           1/1
Running     0          74m
```

If you see pods with a status of ErrImagePull or ImagePullBackOff, check that you are installing to the correct namespace.

If the pods seem to remain in *Pending* status for a long time, there may not be sufficient resources in the cluster. You can use the GCP dashboard to examine cluster resources, or run this command to query a specific pod:

```
kubectl describe pods {pod name} --namespace default
```

The last line of the output gives a useful diagnosis of the problem, for example:

```
Warning  FailedScheduling  40s (x24 over 28m)  default-scheduler  0/2
nodes are available: 2 Insufficient memory.
```

4. Once the pods are ready, find the external IP for the solution, which will be assigned to the Ambassador or Emissary service:

   - Ambassador

     ```
     kubectl get svc [release-name]-ambassador --namespace
     default
     ```

   - Emissary

     ```
     kubectl get svc [release-name]-emissary-ingress --
     namespace default
     ```

   The output should be similar to this:

```
  NAME                 TYPE           CLUSTER-IP     EXTERNAL-IP
PORT(S)                                                 AGE
  gcpprod-ambassador   LoadBalancer   10.48.1.192    35.226.228.226
80:31299/TCP,443:31379/TCP,2222:30537/TCP,31116:32031/TCP,3030:31394/
TCP,3031:31525/TCP,3032:31565/TCP,3033:30959/TCP,3034:32123/TCP,
3035:31855/TCP,3036:30515/TCP,3037:31103/TCP,3038:32227/TCP,
```

```
3039:31334/TCP,3040:32044/TCP,3041:32125/TCP,3042:30903/TCP,
3043:31559/TCP   17m
```

The external IP in the above example is 35.226.228.226. Enter the EXTERNAL-IP into this link to access the Sandbox Admin Console app in your browser: https://Marketplace-console.EXTERNAL-IP.nip.io/. You will see some warnings about the certificate used in the solution; it is safe to accept these and proceed to the Sandbox Admin Console application. Log in to the application using the initial password for the 'sol-admin' user that is generated at solution install, and stored in a Kubernetes secret with the name {{ .Release.Namespace }} {{ .Release.Name }}-acs-default-credentials. For example, if you installed your solution using the release name 'gcp1' into the default namespace, you would view the password using the below command line:

```
kubectl get secret gcp1-acs-default-credentials -o json |
jq -r ".data.admin" | base64 --decode ; echo
```

The Sandbox Admin Console provides information about all parts of the solution, as well as links to the home pages of the included products and services.

5. When you are finished with your solution, you can uninstall it with this command:

   ```
   helm uninstall {release-name} --namespace default
   ```

   Be aware that the trial credit in your account will be used for resources assigned to the cluster, even if there is nothing running in it. If you don't plan to use your cluster for a while, you may consider deleting it and then recreating when you need it again.

VI. Security of your GCP Cluster

GKE is not secure by default. Any resources with an External IP in your new cluster will be accessible. There are a few important things you should do to lock down your cluster:

Create Master Authorized Network for Your Cluster

1. Navigate to **Kubernetes Engine > Clusters**
2. Edit your cluster and set Master authorized networks to *Enabled*. This will ensure that your cluster API can only be accessed by GCP (in your Cloud Shell). If you want to use a local kubectl to connect to your cluster, you can add your own IP address as well (e.g 1.2.3.4/32)

Create a Firewall Rule for your Cluster

1. Navigate to your GCP account Firewall rules page

2. Lock down your Firewall rules and stay on top of them
    - GCP creates some wide open firewall rules allowing ssh and other protocols to your GCP resources. The allowed client IP addresses are set to “0.0.0.0/0” which effectively means open to the internet. We will show you how to delete those below
    - Also when deploying “LoadBalancer” services in GKE, you will get a public IP address for the service and firewall rules will be automatically created letting the internet get to the service’s exposed ports. To address this:
    - First create a firewall rule to allow your IP address to access everything:
        - **VPC network** > **Firewall rules** > **Create Firewall Rule**:
            - **Name:** let-me-in (or whatever name you like)
            - **Targets:** All instances in the network
            - **Source IP Ranges:** {your IP}/32. For example 1.2.3.4/32
            - **Protocols and Ports:** “Allow all”
    - Regularly review your Firewall rules and delete any that have 0.0.0.0/0 in the IP Range. Here is a one-liner that will do that:

      ```
      gcloud compute firewall-rules list --
      format="table(name,sourceRanges.list():label=SRC_RANG
      ES)" |grep "0.0.0.0/0" | grep -Eo '^[^ ]+' | while
      read line; do gcloud compute firewall-rules delete
      $line; done
      ```

Solution Access

Initial Login Credentials

- Click **Launch it** to deploy an instance of an HCLSoftware Product or Business Solution. Once the sandbox is ready to use, the buttons in the section called **Sandbox Links** will become active. These links point directly to your Product or Business Solution where you can start experiencing it hands-on. **Copy the provided product Login ID and password.** These are different from your Marketplace login.
- These are the initial passwords only, and will not apply if they have already been reset/changed. If you are unable to login, please contact Marketplace Support.

Sandbox Admin Console

- The Sandbox Admin Console is accessible once your Business Solution or product has been deployed. **Each solution will generate its own unique password during install.** When deploying to the sandbox, the generated Sandbox Admin Console password will be displayed in the

left-hand section of your Sandbox Details page. When deploying in your own environment, instructions on accessing the password will be output by the Helm install command.
- The default Sandbox Admin Console username is **sol-admin**

Products and Services

- Once you are logged in to the Sandbox Admin Console, you can also view the initial passwords for your services or products. Navigate to your Dashboard and click on General Information located on the card of your product, demo, or service.
- These are the initial passwords only, and will not apply if they have already been reset/changed. If you are unable to login, please contact Marketplace Support.

Access Solutions Programmatically

Most HCLSoftware products provide REST APIs; these are listed in the API Directory. The code examples shown below are written in Java, using the HTTP Client that was new in Java version 11 (java.net.http.HttpClient), but with REST you can choose from many programming languages and REST libraries for your application.

Discovering REST API Documentation for HCLSoftware Products

There are two ways to discover the REST APIs provided by HCLSoftware products. Both rely on the API being documented using the Swagger v2 or OpenAPI v3 standard:

- **Marketplace Catalog**
  - Click on an entry in the API Directory list, or on a catalog card to view its documentation. The *API Documentation* tab will contain available REST API documentation. In the catalog, this is simply a rendering of the documentation; there is no live instance of the service available, so the REST API method cannot be run in this environment. The documentation is provided as reference for your application coding.
- **Swagger UIs in Deployed Solutions**
  - In some cases, a live Swagger or OpenAPI UI is available within a service or product once it is deployed in an installed solution. Once deployed, links to these UIs are shown in the Catalog Detail page,

or via the Sandbox Admin Console in the *General Information* for each entry, under the *API Explorer* tag.

Discovering REST API Base URLs in Deployed Solutions

REST API base URLs are displayed in the Sandbox Admin Console, in the *General Information* for each entry, under the *API BASE* tag.

Handling Self-Signed Certificates

By default, Marketplace generates a self-signed SSL/TLS certificate for each solution. For production use, it is recommended that you override this with a certificate generated for your own domain name. In development, you may choose to operate with the provided certificate.

In Java applications, one approach to self-signed certificates is to override the default trust manager with one that does not validate certificate chains:

```
import java.net.http.HttpClient;
import java.security.GeneralSecurityException;
import java.security.cert.X509Certificate;
import javax.net.ssl.SSLContext;
import javax.net.ssl.TrustManager;
import javax.net.ssl.X509TrustManager;

...

        // Create a trust manager that does not validate
certificate chains
        TrustManager[] trustAllCerts = new TrustManager[] {
                new X509TrustManager() {
                    public
java.security.cert.X509Certificate[] getAcceptedIssuers() {
                        return new X509Certificate[0];
                    }
                    public void checkClientTrusted(

java.security.cert.X509Certificate[] certs, String authType) {
                    }
                    public void checkServerTrusted(

java.security.cert.X509Certificate[] certs, String authType) {
                    }
                }
        };

        // Create the all-trusting trust manager
        SSLContext sc = null;
        try {
            sc = SSLContext.getInstance("SSL");
            sc.init(null, trustAllCerts, new
java.security.SecureRandom());
        } catch (GeneralSecurityException e) {
        }

        // Also need to tell the client to not compare request
host names with the certificate content
        final Properties props = System.getProperties();

props.setProperty("jdk.internal.httpclient.disableHostnameVer
ification", Boolean.TRUE.toString());

        // set the all-trusting trust manager on the client
builder
        HttpClient.Builder builder = HttpClient.newBuilder();
        builder.sslContext(sc);

        // Create HTTP Client to send requests to solution
services
        HttpClient sol_client = builder
                .version(Version.HTTP_1_1)
                .build();
```

Authenticate to Obtain a JSON Web Token (JWT)

If the Access Control Service (ACS) is included in the solution, you first need to authenticate to ACS and then receive a JWT to include on any subsequent API call. Authentication is achieved through a GET request to the *https://Marketplace-auth.{external.ip}.nip.io/login* endpoint using the HTTP Basic authentication protocol.

There are two User IDs (userids) that are created for every solution and you can add more User IDs if you wish.

Userid	**Default password**	**Access**
user	Refer to Initial Login Credentials guide	catalog services
sol-admin	Refer to Initial Login Credentials guide	all services (including Sandbox Admin Console)

To access Sandbox Admin Console, use the *sol-admin* administrator id.

The User ID (userid) and password must be Base64 encoded and included in the **Authorization** HTTP header, as shown in the example below:

```
        String idpw = "user:pass";
        String encodedString =
Base64.getEncoder().encodeToString(idpw.getBytes());
        // replace with your own solution's external ip
address
        ext_ip = "34.67.88.109.nip.io";

        HttpResponse response = null;
        try {
            String url = "https://Marketplace-auth." + ext_ip
+ "/login";

            HttpRequest request = HttpRequest.newBuilder()
                    .uri(URI.create(url))
                    .GET()
                    .timeout(Duration.ofSeconds(150))
                    .header("accept", "application/json")
                    .header("Authorization", "Basic
"+encodedString)
                    .build();

            // Send a request using the HTTPClient that was
created with the all-trusting trust manager
            response = sol_client.send(request,
BodyHandlers.ofString());

            switch (response.statusCode()) {
            case (200):
                // Success - extract JWT from Auth header and
save it for future requests
                HttpHeaders headers = response.headers();
                List auths =
headers.allValues("Authorization");
                // Header requires 'Bearer' before actual
token value
                token = "Bearer "+(String)response.body();
                // You may want to persist the token at this
point
                break;
            case (404):
                System.out.println("Marketplace-auth login not
found, perhaps ACS not included in this solution");
                break;
            case (500):
                // Workaround: ACS may take a short time to
complete initialization
                System.out.println("Marketplace-auth login
returned 500, will retry once after a short pause");
                Thread.sleep(60000);
                loginToSolution(idpw, soldomain);
                break;
            default:
                System.out.println("Marketplace-auth login
failed: "+response.statusCode());
                System.out.println(response.body());
            }
        }catch(Exception e) {
            e.printStackTrace();
        }
```

The JWT will be valid for 5 minutes and then will expire, after which re-authentication will be necessary. If you wish to query the expiry time of the token, the code below shows how to do that using the Auth0 java-jwt library.

```
import com.auth0.jwt.JWT;
import com.auth0.jwt.exceptions.JWTDecodeException;
import com.auth0.jwt.interfaces.DecodedJWT;
...
        DecodedJWT jwt = null;
        String jwtString = token.replace("Bearer ","");

        try {
            jwt = JWT.decode(jwtString);

            Date expiryTime = jwt.getExpiresAt();
            Date now = new Date();
            System.out.println("token expires at:
"+expiryTime);
            System.out.println("time now is: "+now);
            tokenExpired = now.after(expiryTime);
            if (tokenExpired)
                 System.out.println("Token for solution access
is already expired - expect 401/403 the re-authentication");
                 // Could choose to re-login here, but an
unexpired token may still expire between this point and the
next API call
            } catch (JWTDecodeException exception){
                 System.out.println("JWT decode failed for
token: "+jwtString);
                 exception.printStackTrace();
            }
```

Call the Product REST API

Once you have the URL for the REST method you want to call, an HTTP client that will handle self-signed certificates (unless you have applied your own domain/certificate), and the authorization token (if ACS is used in your solution), then you are ready to make a call to a product REST API.

There are a couple of workarounds shown in the example below, with a counter to limit the retry attempts.

```
        private static void callApi() {
        HttpResponse response = null;
        apiCallCounter++;

        if (apiCallCounter > apiCallRetryLimit) {
            System.out.println("Reached max attempts to call
solution REST API, giving up");
            return;
        }
        boolean tokenExpired = false;

        try {
            String url = "https://test-data-synth." + ext_ip +
"/datasynth/1.0/data/ccVisa?count=5";

            HttpRequest.Builder builder =
HttpRequest.newBuilder();
            if (acs) {
                builder.header("Authorization", token);  // be
careful not to add null token (-> NPE)
            HttpRequest request = builder
                    .uri(URI.create(url))
                    .GET()
                    .timeout(Duration.ofSeconds(90))
                    .header("accept", "application/json")
                    .build();

            response =
                    sol_client.send(request,
BodyHandlers.ofString());
            switch (response.statusCode()) {
            case (200):
                System.out.println("tds GET returned 200");
                JSONArray testDataJson = new
JSONArray((String)response.body());
                System.out.println(testDataJson.toString(4));
                break;
            case (307):
                System.out.println("*** RC 307 Redirect -
suggests protected method was called without auth header
data");
                break;
            case (401):
            case (403):
                System.out.println("*** Authentication/
Authorization issue, RC: "+response.statusCode());
                System.out.println("May be token expiry, re-
authenticate (once) and retry");
                // call method with login logic shown above
                loginToSolution("user:pass", ext_ip);
                apiCallCounter = apiCallRetryLimit -1;
                callApi();
                break;
            case (404):
                System.out.println("*** 404: tds records not
found... service may not be included or not running");
                break;
            case (503):
                System.out.println("*** 503 returned - wait 5
seconds then try again");
                Thread.sleep(5000);
                callApi();
            break;
            default:
                System.out.println("*** tds home access
failed: "+response.statusCode());
                System.out.println("response body:
"+response.body());
                System.out.println("response headers::
"+response.headers());
            }

        }catch(Exception e) {
            e.printStackTrace();
        }
    }
```

Solution Configuration

Managing User Access to a Solution

Access to the service endpoints in a solution is controlled by the Access Control Service (ACS). Access to all service endpoints is restricted to known users who must authenticate to ACS, either through an on-screen login (for browser access) or a programmatic login (for application access).

A number of default users and roles are created to allow initial access and setup of the solution. If you need to change or add to the default users, this can be done after the solution has been deployed by accessing the administrative console of the included Keycloak service.

Adding and Changing Users, Passwords, and Roles in a Deployed Solution

1. To update your solution's users and credentials, you need to access Keycloak.

    - Copy and paste the below link into your domain search bar, replacing **${SOLUTION_DOMAIN}** with your personal Solution Domain name

      ```
      https://Marketplace-kc.${SOLUTION_DOMAIN}/auth/admin/
      MarketplaceSolution/console
      ```

    - Log in with the sol-admin credentials

2. On the left of the screen, under **Manage**, select **Users**.

    - To edit an existing User:
        - Choose View all users
        - Select the user you wish to edit
        - On the **Details** tab, you can select one or more **Required Actions** (such as Update Password), impersonate the user, or make other adjustments.
        - To add a new User:
            - Choose **Add user**
            - Enter the appropriate information, at a minimum the username will be added. **Click save**.
            - Move to the Credentials tab and set a temporary password.
            - At this point, the new user will be able to log into the solution, set a new password, and access catalog services. They will not be able to access the Sandbox Admin Console.
            - **Note:** The user will have to set a new password in order to activate their account.

3. If you wish to allow the new user to access the Sandbox Admin Console, do the following:

    - Go to the **Role Mappings** tab in Keycloak.
    - Select the **solution-admin** role and add it for that user; the change should take effect immediately.

4. Close the Keycloak window when you have completed your changes.

Applying a Domain and SSL Certificate to Your Solution

By default, a Marketplace Solution will create its own self signed SSL certificate. This is meant for initial development and testing and will throw many warnings and cause issues on most HTTP clients, including browsers and some of the most common programmatic client libraries. There are options to apply your own domain name and a recognized SSL certificate to a solution; these quick start instructions will describe the simpler method, using the Let's Encrypt service to dynamically generate a certificate.

You will need the following prerequisites:

- A cluster in GCP configured to run a solution
- A domain name, for example one purchased from https://domains.google.com/

This document will go through the following steps:

- Configuring your domain name and DNS zone
- Creating a GCP service account with the DNS Administrator role
- Creating a Kubernetes secret with the service account credentials
- Configuring your solution to use the domain name and request certificate generation at install
- Accessing your solution using the domain name and validating the certificate

---

I. Configure your domain name and DNS zone

1. Log into your GCP account at https://console.cloud.google.com
2. Navigate to **Network Services** > **Cloud DNS** and click **Create Zone**.
3. Enter a name for the zone and your domain name in 'DNS name' then click **create**.
4. When you click on the newly created zone, you will see a list of DNS servers, which need to be copied into your domain name configuration.
5. Log into https://domains.google.com; in 'My Domains' click on the domain name that you are using for the solution.
6. Navigate to **DNS** then select **Use custom name servers**.
7. Cut and paste the DNS servers (one at a time, do not include the trailing '.') from the GCP Cloud DNS page to the Google domain name entry, then save.

II. Create a GCP service account with the DNS administrator role

1. Log into your GCP trial account at https://console.cloud.google.com.

2. Navigate to **IAM & Admin** > **Service Accounts** and click on **Create service account**.
3. Choose a name for the service account and ID (or accept the default ID); make a note of your service ID.
4. In the menu of roles, find **DNS** and select **DNS Administrator**, then continue.
5. Click on **Create key** and select **JSON**. The key will be generated - download to your local machine.
6. Find the downloaded key file and rename it to "credentials.json," then upload that file to your GCP Cloud Shell.

III. Create a Kubernetes secret with the service account credentials

```
kubectl create secret generic gcp-service-account-secret --
from-file=./credentials.json
```

IV. Configure your solution to use the domain name and to request certificate generation at install

There are two ways to do this: a) unpack, edit, and then repack the solution Helm chart, or b) use the existing Helm chart but provide the configuration as override on the Helm install command

Option A: Unpack, edit, and repack the solution Helm chart 1. Untar the solution chart and edit the top-level 'values.yaml' file. Near the top of the file you will see 'global:' and nested under that, 'domain: - edit the value of the domain property to be a subdomain of your domain name, for example:

```
global:
  domain: "sol1.alexmul.dev"
```

1. Search for the *certificate:* property in the same file, and add/edit these values. The GCP project ID can be seen by clicking the down-arrow next to your project name in the GCP console.

```
certificate:
  type: "production"
  email: ${YOUR_GCP_ACCOUNT_EMAIL}
  dns:
    type: "clouddns"
    project: ${YOUR_GCP_PROJECT_ID}
    serviceAccountSecretRef:
      name: ${YOUR_GCP_SERVICE_ACCOUNT_SECRET_NAME}
      key: "credentials.json"
```

- For example:

```
certificate:
  type: "production"
  email: "Marketplace.mulholland@gmail.com"
  existingCertificateSecret: ""
  dns:
    type: "clouddns"
    project: "dark-airway-256814"
    region: ""
    accessKeyID: ""
    serviceAccountSecretRef:
      name: "gcp-service-account-secret"
      key: "credentials.json"
```

1. Optional: Edit the top-level *chart.yaml* file and increment the version number of the chart. It is a best practice to increment the version number when any change is made to the Helm chart.

2. Repackage the Helm chart with the *Helm package* command.

3. Install your modified Helm chart:

```
helm install [my-release-name] [my-solution.tgz] --set
global.hclImagePullSecret=[secret-name]
```

Option B: Install your original Helm chart with additional configuration override on the install command, similar to this:

```
helm install udeploy alex-deploy-0.1.0.tgz --set
global.hclImagePullSecret=secret-
name,global.hclImagePullSecret=secret-
name,global.domain=sol1.alexmul.com,certificate.type=producti
on,certificate.email=Marketplace.mulholland@hcl.com,certifica
te.dns.type=clouddns,certificate.dns.project=dark-
airway-256814,certificate.dns.serviceAccountSecretRef.name=gcp
-service-account-
secret,certificate.dns.serviceAccountSecretRef.key=credentials
.json
```

V. Access your solution using the domain name and validiate the certificate

For example:

```
https://Marketplace-console.sol1.alexmul.dev
```

Override the Default Monitoring Dashboard

Solutions from Marketplace have the following monitoring services prepackaged in them: * Prometheus * Grafana

You can access the default monitoring services through their respected gatewayIP through your solution:

- For Prometheus: metrics.[gatewayIP].nip.io
- For Grafana: dashboard.[gatewayIP].nip.io

Accessing metrics through Prometheus

To access metrics of your solution, go to metrics.[gatewayIP].nip.io. Here, different metrics can be viewed and executed.

Registering a custom dashboard using Grafana

Note: For a custom dashboard you need a configmap with JSON data. If you already have it, it should look similar to this:

```
{
  "annotations": {
    "list": [
      {
        "builtIn": 1,
        "datasource": "-- Grafana --",
        "enable": true,
        "hide": true,
        "iconColor": "rgba(0, 211, 255, 1)",
        "name": "Annotations & Alerts",
        "type": "dashboard"
      }
    ]
  },
  "editable": true,
  "gnetId": null,
  "graphTooltip": 0,
  "id": 1,
  "iteration": 1570029666260,
  "links": [],
  "panels": [
    {
      "collapsed": false,
      "gridPos": {
        "h": 1,
        "w": 24,
        "x": 0,
        "y": 0
      },
      "id": 10,
      "panels": [],
      "title": "Pod",
      "type": "row"
    },
    {
      "aliasColors": {},
      "bars": false,
      "dashLength": 10,
      "dashes": false,
      "fill": 1,
      "gridPos": {
        "h": 7,
        "w": 24,
        "x": 0,
        "y": 1
      },
      "id": 8,
      "legend": {
        "alignAsTable": true,
        "avg": true,
        "current": true,
        "max": false,
        "min": false,
        "rightSide": true,
        "show": true,
        "total": false,
        "values": true
      },
      "lines": true,
      "linewidth": 1,
      "links": [],
      "nullPointMode": "null",
      "percentage": false,
      "pointradius": 5,
      "points": false,
      "renderer": "flot",
      "seriesOverrides": [],
      "spaceLength": 10,
      "stack": false,
      "steppedLine": false,
      "targets": [
        {
          "expr": "sort_desc(sum by (pod_name)
(rate(container_network_receive_bytes_total{pod_name=\"$pod\"}
[1m])))",
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "{{ pod_name }}",
          "refId": "A"
        }
      ],
      "thresholds": [],
      "timeFrom": null,
      "timeRegions": [],
      "timeShift": null,
      "title": "Network I/O",
      "tooltip": {
        "shared": true,
        "sort": 0,
        "value_type": "individual"
      },
      "type": "graph",
      "xaxis": {
        "buckets": null,
        "mode": "time",
        "name": null,
        "show": true,
        "values": []
      },
      "yaxes": [
        {
          "format": "decbytes",
          "label": null,
          "logBase": 1,
          "max": null,
          "min": null,
          "show": true
        },
        {
          "format": "short",
          "label": null,
          "logBase": 1,
          "max": null,
          "min": null,
          "show": true
        }
      ],
      "yaxis": {
        "align": false,
        "alignLevel": null
      }
    },
    {
      "aliasColors": {},
      "bars": false,
      "dashLength": 10,
      "dashes": false,
      "datasource": "Prometheus",
      "fill": 1,
      "gridPos": {
        "h": 7,
        "w": 24,
        "x": 0,
        "y": 8
      },
      "id": 12,
      "legend": {
        "alignAsTable": true,
        "avg": true,
        "current": false,
        "max": false,
        "min": false,
        "rightSide": true,
        "show": true,
        "total": false,
        "values": true
      },
      "lines": true,
      "linewidth": 1,
      "links": [],
      "nullPointMode": "null",
      "percentage": false,
      "pointradius": 5,
      "points": false,
      "renderer": "flot",
      "seriesOverrides": [],
      "spaceLength": 10,
      "stack": false,
      "steppedLine": false,
      "targets": [
        {
          "expr":
"kube_persistentvolumeclaim_resource_requests_storage_bytes{k
ubernetes_namespace=\"$namespace\",persistentvolumeclaim=\"$p
vc\"}",
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "{{ persistentvolumeclaim }}",
          "refId": "A"
        }
      ],
      "thresholds": [],
      "timeFrom": null,
      "timeRegions": [],
      "timeShift": null,
      "title": "Persistent Volume Claim",
      "tooltip": {
        "shared": true,
          "sort": 0,
          "value_type": "individual"
        },
        "type": "graph",
        "xaxis": {
          "buckets": null,
          "mode": "time",
          "name": null,
          "show": true,
          "values": []
        },
        "yaxes": [
          {
            "format": "decgbytes",
            "label": null,
            "logBase": 1,
            "max": null,
            "min": null,
            "show": true
          },
          {
            "format": "short",
            "label": null,
            "logBase": 1,
            "max": null,
            "min": null,
            "show": true
          }
        ],
        "yaxis": {
          "align": false,
          "alignLevel": null
        }
      },
      {
        "collapsed": true,
        "gridPos": {
          "h": 1,
          "w": 24,
          "x": 0,
          "y": 15
        },
        "id": 4,
        "panels": [
          {
            "aliasColors": {},
            "bars": false,
            "dashLength": 10,
            "dashes": false,
            "datasource": "Prometheus",
            "fill": 1,
            "gridPos": {
```

```
          "h": 8,
          "w": 24,
          "x": 0,
          "y": 2
        },
        "id": 2,
        "legend": {
          "alignAsTable": true,
          "avg": true,
          "current": true,
          "max": false,
          "min": false,
          "rightSide": true,
          "show": true,
          "total": false,
          "values": true
        },
        "lines": true,
        "linewidth": 2,
        "links": [],
        "nullPointMode": "null",
        "percentage": false,
        "pointradius": 5,
        "points": false,
        "renderer": "flot",
        "seriesOverrides": [],
        "spaceLength": 10,
        "stack": false,
        "steppedLine": false,
        "targets": [
          {
            "expr": "sum by(container_name)
(container_memory_usage_bytes{pod_name=\"$pod\",container_name
!=\"POD\"})",
            "format": "time_series",
            "hide": false,
            "interval": "",
            "intervalFactor": 1,
            "legendFormat": "Current: {{ container_name }}",
            "refId": "A"
          },
          {
            "expr": "avg by(container)
(kube_pod_container_resource_requests_memory_bytes{pod=\"$pod\
"})",
            "format": "time_series",
            "hide": false,
            "intervalFactor": 1,
            "legendFormat": "Requested: {{ container }}",
            "refId": "B"
          },
          {
              "expr": "avg by(container)
(kube_pod_container_resource_limits_memory_bytes{pod=\"$pod\",
container=\"$container\"})",
              "format": "time_series",
              "hide": false,
              "intervalFactor": 1,
              "legendFormat": "Limit: {{ container }}",
              "refId": "C"
            }
          ],
          "thresholds": [],
          "timeFrom": null,
          "timeRegions": [],
          "timeShift": null,
          "title": "Memory Usage",
          "tooltip": {
            "shared": true,
            "sort": 0,
            "value_type": "individual"
          },
          "type": "graph",
          "xaxis": {
            "buckets": null,
            "mode": "time",
            "name": null,
            "show": true,
            "values": []
          },
          "yaxes": [
            {
              "format": "bytes",
              "label": null,
              "logBase": 1,
              "max": null,
              "min": null,
              "show": true
            },
            {
              "format": "short",
              "label": null,
              "logBase": 1,
              "max": null,
              "min": null,
              "show": true
            }
          ],
          "yaxis": {
            "align": false,
            "alignLevel": null
          }
        },
        {
          "aliasColors": {},
          "bars": false,
          "dashLength": 10,
          "dashes": false,
          "fill": 1,
          "gridPos": {
            "h": 9,
            "w": 24,
            "x": 0,
            "y": 10
          },
          "id": 6,
          "legend": {
            "alignAsTable": true,
            "avg": true,
            "current": true,
            "max": false,
            "min": false,
            "rightSide": true,
            "show": true,
            "total": false,
            "values": true
          },
          "lines": true,
          "linewidth": 1,
          "links": [],
          "nullPointMode": "null",
          "percentage": false,
          "pointradius": 5,
          "points": false,
          "renderer": "flot",
          "seriesOverrides": [],
          "spaceLength": 10,
          "stack": false,
          "steppedLine": false,
          "targets": [
            {
              "expr": "sum by (container_name)
(rate(container_cpu_usage_seconds_total{image!
=\"\",container_name!=\"POD\",pod_name=\"$pod\"}[1m]))",
              "format": "time_series",
              "hide": false,
              "intervalFactor": 1,
              "legendFormat": "{{ container_name }}",
              "refId": "A"
            },
            {
              "expr":
"kube_pod_container_resource_requests_cpu_cores{pod=\"$pod\",
container=\"$container\"}",
              "format": "time_series",
              "intervalFactor": 1,
              "legendFormat": "Requested: {{ container }}",
              "refId": "B"
            },
            {
              "expr":
"kube_pod_container_resource_limits_cpu_cores{pod=\"$pod\",
container=\"$container\"}",
              "format": "time_series",
              "intervalFactor": 1,
              "legendFormat": "Limit: {{ container }}",
              "refId": "C"
            }
          ],
          "thresholds": [],
          "timeFrom": null,
          "timeRegions": [],
          "timeShift": null,
          "title": "CPU Uage",
          "tooltip": {
            "shared": true,
            "sort": 0,
            "value_type": "individual"
          },
          "type": "graph",
          "xaxis": {
            "buckets": null,
            "mode": "time",
            "name": null,
            "show": true,
            "values": []
          },
          "yaxes": [
            {
              "format": "short",
              "label": null,
              "logBase": 1,
              "max": null,
              "min": null,
              "show": true
            },
            {
              "format": "short",
              "label": null,
              "logBase": 1,
              "max": null,
              "min": null,
              "show": true
            }
          ],
          "yaxis": {
            "align": false,
            "alignLevel": null
          }
        }
      ],
      "title": "Container",
      "type": "row"
    }
  ],
  "schemaVersion": 18,
  "style": "dark",
  "tags": [],
  "templating": {
    "list": [
      {
        "current": {
          "selected": true,
          "text": "atit",
          "value": "atit"
        },
        "hide": 2,
        "label": null,
        "name": "namespace",
        "options": [
          {
            "selected": true,
            "text": "atit",
            "value": "atit"
          }
        ],
        "query": "atit",
        "skipUrlSync": false,
        "type": "textbox"
      },
      {
        "allValue": null,
        "current": {
          "text": "aa-anchor",
          "value": "aa-anchor"
        },
        "datasource": "Prometheus",
        "definition":
"label_values(kube_service_labels{namespace=\"$namespace\"},s
ervice)",
        "hide": 0,
        "includeAll": false,
        "label": null,
        "multi": false,
        "name": "service",
        "options": [],
        "query":
"label_values(kube_service_labels{namespace=\"$namespace\"},s
ervice)",
        "refresh": 1,
        "regex": "",
        "skipUrlSync": false,
        "sort": 1,
        "tagValuesQuery": "",
        "tags": [],
        "tagsQuery": "",
        "type": "query",
        "useTags": false
      },
      {
        "allValue": null,
        "current": {
          "text": "aa-anchor-85bcf65bc9-ghz5f",
          "value": "aa-anchor-85bcf65bc9-ghz5f"
        },
        "datasource": "Prometheus",
        "definition":
"label_values(kube_pod_info{namespace=\"$namespace\",pod=~\"$
service.*\"},pod)",
        "hide": 0,
        "includeAll": false,
        "label": null,
        "multi": false,
        "name": "pod",
        "options": [],
        "query":
"label_values(kube_pod_info{namespace=\"$namespace\",pod=~\"$
service.*\"},pod)",
        "refresh": 1,
        "regex": "",
        "skipUrlSync": false,
        "sort": 1,
        "tagValuesQuery": "",
        "tags": [],
        "tagsQuery": "",
        "type": "query",
        "useTags": false
      },
      {
        "allValue": null,
        "current": {
          "isNone": true,
          "text": "None",
          "value": ""
        },
        "datasource": "Prometheus",
        "definition":
"label_values(kube_pod_spec_volumes_persistentvolumeclaims_in
fo{pod=\"$pod\"}, persistentvolumeclaim)",
        "hide": 2,
        "includeAll": false,
        "label": null,
        "multi": false,
        "name": "pvc",
        "options": [],
        "query":
"label_values(kube_pod_spec_volumes_persistentvolumeclaims_in
fo{pod=\"$pod\"}, persistentvolumeclaim)",
        "refresh": 1,
        "regex": "",
        "skipUrlSync": false,
        "sort": 0,
        "tagValuesQuery": "",
        "tags": [],
        "tagsQuery": "",
        "type": "query",
        "useTags": false
      }
    ]
  },
  "time": {
    "from": "now-6h",
    "to": "now"
  },
  "timepicker": {
    "refresh_intervals": [
      "5s",
      "10s",
      "30s",
      "1m",
      "5m",
      "15m",
      "30m",
      "1h",
      "2h",
      "1d"
    ],
    "time_options": [
      "5m",
      "15m",
      "1h",
      "6h",
      "12h",
      "24h",
      "2d",
      "7d",
      "30d"
    ]
  },
  "timezone": "browser",
  "title": "Home",
  "uid": "0",
  "version": 1
}
```

If you do not have the needed JSON, you can visit the default Grafana dashboard and customize your data as needed.

1. Dashboard.[gatewayIP].nip.io
2. Dashboard settings > JSON Model

Note: The following is a template and the values will need to be changed accordingly.

```
{
  "annotations": {
    "list": [
      {
        "builtIn": 1,
        "datasource": "-- Grafana --",
        "enable": true,
        "hide": true,
        "iconColor": "rgba(0, 211, 255, 1)",
        "name": "Annotations & Alerts",
        "type": "dashboard"
      }
    ]
  },
  "editable": true,
  "gnetId": null,
  "graphTooltip": 0,
  "id": 2,
  "links": [],
  "panels": [],
  "schemaVersion": 18,
  "style": "dark",
  "tags": [],
  "templating": {
    "list": []
  },
  "time": {
    "from": "now-6h",
    "to": "now"
  },
  "timepicker": {
    "refresh_intervals": [
      "5s",
      "10s",
      "30s",
      "1m",
      "5m",
      "15m",
      "30m",
      "1h",
      "2h",
      "1d"
    ],
    "time_options": [
      "5m",
      "15m",
      "1h",
      "6h",
      "12h",
      "24h",
      "2d",
      "7d",
      "30d"
    ]
  },
  "timezone": "",
  "title": "abc",
  "uid": "o0axiI2Wk",
  "version": 1
}
```

Within your Helm chart in your service, create a configmap with the customized JSON. **Note:** If metrics need to be changed within your JSON, they can be accessed through Prometheus's metrics.

Creating a Custom Error Page

Custom Ambassador or Emissary Error Pages Resource

Marketplace solutions now have custom error page capability for the included Ambassador or Emissary ingress controller, using a new custom resource called **ErrorPage**. These pages can be global, applying to all pages in your solution, or they can be configured to apply to individual pages. You'll first add this new ErrorPage to your solution. To create a an ErrorPage inline, run a command following the steps below:

```
cat << EOF | kubectl apply -f -
apiVersion: "hclMarketplace.com/v1beta1"
kind: ErrorPage
metadata:
  name: "myerrorpage"
  namespace: "mynamespace"
spec:
  service:
    name: foo3-snoop
    bypass: true
EOF
```

You can also create the same resource running the following commands:

- **kubectl apply -f myerrorpage.yaml** if myerrorpage.yaml is the below:

```
apiVersion: "hclMarketplace.com/v1beta1"
kind: ErrorPage
metadata:
  name: "myerrorpage"
  namespace: "mynamespace"
spec:
  service:
    name: foo3-snoop
    bypass: true
```

1. To create a new ErrorPage after you've installed your solution, run the following command:

```
apiVersion: "hclMarketplace.com/v1beta1"
kind: ErrorPage
metadata:
  name: "myerrorpage"
  namespace: "mynamespace"
spec:
  statusCode: 404
  html: <html><head><title>my custom 404</title></
head><body><h1>Oops!</h1></body></html>
```

- This will create a new global ErrorPage that will be shown for all 404 status codes that happen for that specific solution.

- If you only wish to apply specific ErrorPages to a particular service mapping, follow the steps below:

```
apiVersion: "hclMarketplace.com/v1beta1"
kind: ErrorPage
metadata:
  name: "myerrorpage-foo"
  namespace: "mynamespace"
  labels:
    test: errorpages
spec:
  statusCode: 405
  html: <html><head><title>my custom 405 for foo</title></
head><body><h1>Foo Oops!</h1></body></html>
  service:
    name: release-foo
```

- It's important to understand that a service mapping with a set ErrorPage will ignore all global ErrorPages. In the example directly above, requests mapped to `release-foo` will only have a custom ErrorPage for status code `405` and not `404` or any other error page status codes set globally.

- If you want a particular service to ignore any global ErrorPages, you can create a null `ErrorPage` resource (with no html specified) as follows:

```
apiVersion: "hclMarketplace.com/v1beta1"
kind: ErrorPage
metadata:
  name: "myerrorpage-foo"
  namespace: "mynamespace"
  labels:
    test: errorpages
spec:
  service:
    name: release-foo
    bypass: true
```

1. To create the custom resource, the following are required: `statusCode` `and html` or `service` (where `name` is required). **Note:** You must have either a statusCode and html together or a service that also define the name - the following combinations are accepted:

    - An ErrorPage that defines statusCode and html, without a service.
    - An ErrorPage that does not define statusCode and html, but does define a service (with the name) and bypass property
    - An ErrorPage that defines a statusCode, with html and defines a service (with name)

2. The following would be example of an invalid ErrorPage:

3. An ErrorPage that defines a statusCode alone, an html alone

Support and Resources

Support & Resources

For further documentation support and resources, see the below information!

Support

- Marketplace Access Support Team
- HCLSoftware Customer Support Resources
- Feedback or questions on Marketplace Docs? Drop us a line!

Resources

- Marketplace Docs Home
    - Blog

    - FAQ
    - Glossary
    - Guides
    - Release Notes
- Marketplace Request Access Form
- HCLSoftware Marketplace Webpage
- HCL's Cloud Native Approach
- HCLSoftware Cloud Native White Paper
- Cloud Native-as-a-service: HCL Now

Tutorial Videos

How to Grow your Business with HCLSoftware Marketplace 2.0

Watch how HCLSoftware Marketplace 2.0 is expanding the value for HCL Business Partners by introducing new capabilities to help grow your business.

Getting Started in Marketplace

The following is a brief step-by-step video explaining how to launch a sandbox and how to access your Business Solution or product from the Sandbox Admin Console.

Blog

Marketplace Blog

You've made it to the Marketplace Blog area of our Doc site! Our most informal and most digestible content, blogs are meant to explore subjects relating to Marketplace, cloud native software, and the tech industry as a whole. Our blogs provide a wide array of knowledge in an easy-to-consume, more conversational format (lists, Q & A, short paragraphs, tutorial videos, interviews, etc.).

If you find a topic that piques your interest, look out for internal links that will lead you to more content within our Doc site. Our blog is a catalyst for conversation, so please feel free to send us content suggestions or feedback - happy reading!

Cloud Native

Cloud Native 101

In the 21st century, it's not uncommon to hear statements suggesting our online content lives "forever in the cloud," but what exactly is the cloud? The whole impetus of this sentiment hinges on the fact that the cloud is such a nebulous term and concept, even to those who work in a digital space. As a cloud-based software company, our own team at HCLSoftware often asks for clarity on cloud-related products and terms. As in any industry, there are discrepancies between exact definitions and delineations when it comes to terminology and its broader applications, but there usually exists a generally accepted scope of terms. Below, we sat down with our Global Technology Director Alex Mulholland to break down all things cloud-related.

The Cloud

1. **Q: What is the “cloud” and what are some examples?**

   Alex: A “cloud” is considered any shared/managed hosting environment for running applications. Some familiar applications you may recognize which utilize the cloud are: Slack, Zoom, Microsoft 365, Atlassian, Salesforce, and of course HCLSoftware Marketplace!

2. **Q: Is it a capital "c" in the word cloud? As in a reference to a company or application called Cloud?**

   Alex: No, the cloud is a lower-case term, as it does not belong to one company and is a general term. Businesses may use the word cloud in their names and choose to capitalize it, but the cloud and/or cloud computing is not proprietary to one company or business.

3. **Q: Can anyone access information or data in any cloud?**

   Alex: That depends on which type of cloud you operate. Clouds can be public (open source access for anyone with an internet connection), private (hosted by a company in their own datacentre for private use) or hybrid (simply a combination of public and private cloud resources, such as an application which includes content retrieved from an on-prem database but has processes running on a public cloud platform).

4. **Q: What type of data/software can utilize the cloud?**

   Alex: Nearly anything data-related can be “run on cloud.” A 30-year-old application could be installed on a Virtual Machine (VM) and be technically “running in the cloud.” There may be multiple VMs running on a single machine. Most applications can be run from the cloud, but some with performance-intensive data, such as video editing software or computer-intensive graphic design software and applications, are better suited off of the cloud.

5. **Q: What is the difference between “cloud-based" and “non-cloud” data management?**

   Alex: Cloud-managed data means that resources (mainly the Central Processing unit [CPU], memory, and file storage) are shared across multiple applications. By contrast, “non-cloud” data would include individual setup and management of custom infrastructure just to run any given application; non-cloud means each application has fixed, dedicated, local resources.

Cloud Native

1. **Q: What is a cloud native application?**

   Alex: Cloud native (applications) are modern apps that are specifically designed to take advantage of the cloud. They are specifically designed, or adapted, to take advantage of shared cloud resources and dynamic management (e.g. scaling or restarts).

2. **Q: What is the difference between the cloud and cloud native?**

   Alex: The cloud is mostly about hosting what you already but in a different place - in other words, getting servers out of your own data center and into somebody else’s. Cloud native is an entirely new way of building, deploying, and maintaining applications that depends on containerization, orchestration, and other technologies to build a robust, highly-scalable/autoscaling, and incredibly reliable IT infrastructure. Your application could utilize the cloud, but not be cloud native, but a cloud native application will always utilize the cloud.

3. **Q: So how does this relate to containers or a "containerized" application?**

   Alex: Containers, or standardized units of software that package up code, lend themselves well to cloud, being portable and (ideally) fast to start up. A containerized app can be 'placed' efficiently by cloud automation.

Marketplace

1. **Q: Marketplace is an application that is built to run within the cloud, right?**

   Alex: Yes, HCLSoftware Marketplace is a cloud-based Kubernetes application, whose containers are configured through Helm charts.

2. **Q: What exactly does Marketplace do within the cloud?**

   Alex: Marketplace is a website that provides a catalog of cloud native-enabled enterprise software products aka the HCLSoftware cloud native product catalog (we refer to it as The Catalog). We also have tools that will package these products with platform services, into a single Helm chart that installs and configures the complete solution.

3. **Q: Who are the typical customers?**

   Alex: HCLSoftware and Marketplace customers and business partners are those who want to get started quickly with Kubernetes, to get the benefits of cloud native technologies for their businesses. Our customers utilize Marketplace for many purposes including digital solutions, secure devops, security and automation, and platform components and tools, among many more. **No prior cloud experience needed!**

4. **Q: How do I move to a cloud native environment, without specific cloud native skills?**

> **Alex**: You can easily and quickly move your HCLSoftware products to the cloud of your choice through expert assistance and unlock the full potential of your products. Our HCL Now service will assist you with cloud migration and retention, so you can quickly become cloud enabled without having to retrain or hire new staff.

Wrapping Up

1. **Q: There are a lot of specific terms in this article, where can I go to learn more about cloud-related terminology?**

   > **Alex**: A: One of resources within the our Marketplace Documentation site is our Marketplace Glossary. This living document is updated over time and can help you learn more about cloud-related language. If you have any feedback or there’s a word we missed, please let us know at hclMarketplace@hcl.com.

   [Back to Top]

We hope this blog helped you learn about the cloud and bring the concept a little more down to earth. To learn more about the cloud, cloud native, or Marketplace, check out the wide variety of resources in our public-facing Marketplace Doc site or request access to Marketplace today!

Developer Practices

Migrate a Postgresql Database in Kubernetes with Helm Hooks

Introduction

Kubernetes has become an incredible tool for cloud application development and delivery. It has never been more easy to create scalable, resilient and accessible applications. However, there are still some components of an enterprise level application that can be particularly challenging in Kubernetes. One such component is a database. Evolution of Kubernetes has led to database vendors to transform their databases to run in a K8s cluster, but its not always a smooth process.

In our application, we use Keycloak which is an identity and access management service. Keycloak uses, although not limited to, Postgresql to save essential data. It is worth mentioning that our application along with any dependencies are packaged in a Helm chart, as we use Helm to deploy our application in a K8s cluster. Coming back, initially we used a standalone version of Postgresql i.e. only one instance of Postgresql pod was up. Naturally this caused problems down the lane as our application receieved more traffic. So we decided to change our Postgresql dependency to Postgresql HA. This meant we had could not just run a Helm upgrade and expect everything to be in a ready state. We had to backup our data, delete any old Postgresql instance and artifacts runnning in cluster. Then run Helm upgrade and restore the data.

Well that means a lot of manual intervention, and with it the probablity of something going wrong. Also this Helm chart is used by our user to deploy application in their own cluster and we wanted the upgrade to run without them executing any of the above steps.

Design Overview

After considering different options, we decided the best way for us is to use Helm hooks and Kubernetes Jobs to perform this migration.

Helm hook is a mechanism to intervene at certain point in a releases life cycle. For eg. if we want to create some Kubernetes objects before our applciation is deployed, we can use an Helm "pre-install" hooks to create such object. There are number of hooks available and you can learn more about them here. In the context of our problem we will be focusing on "pre-upgrade" and "post-upgrade" Helm hooks. We will be combining these hooks with K8s jobs. A k8s job is a mechanism that can be used to run a certain task to completion.

When you combine pre-upgrade Helm hook with a K8s job, Helm will run a job before upgrading any K8s component related to our application. Similarly the post-upgrade job, will run after Helm upgrade is completed. It is worth nothing that due to any reason if either of these jobs fails, it will result in Helm upgrade to fail and depending on the perference a rollback can be initiated.

Diving deeper

Pre-uprade

This stage take care of taking backup and cleaning up old postgresql K8s objects.

We first create a PVC that will hold the backup of old Postgresql instance.

After this, we will run the pre-upgrade job. This job deploys a pod that runs a Python script. This script will do following things:

- It will check if an upgrade is necessary, in case upgrade was already completed or POstgresql-HA is already deployed.

- Once it is determined that an upgrade is necessary, it will then create a configmap that keeps track of upgrade process. We will cover more about this later on.
- It will then perform a backup of postgresql DB and save that in the PV we had created earlier.
- After successful backup it will delete Kubernetes objects related to old Postgresql DB namely statfulset and persistent volumes. It will also scale down Keycloak, so that no calls are made to databse during migration.
- This concludes the pre-upgrade process and Helm can now run upgrade. In case there is a failure at any point, the upgrade process is terminated and we can view the logs of the pod associated with the pre-upgrade job to determine the failure.

Post-upgrade

On completing the upgrade process, Helm will invoke post-upgrade job. There is an init-container running in the pod associated with this job. It's job is to check if new Postgresql pods are up and running, after which we will a Python script is executed. It will:

- Check if pre-upgrade job was run and whether restore is necessary.
- If a restore is required, it'll restore the old database and scale up Keycloak.
- It'll also update the configmap specifying that the upgrade is completed.
- Similar to pre-upgrade job, if there is a failure, Helm upgrade will fail. It is then up to user to decide to either re-run Helm upgrade or rollback.

Ensuring state consistency

Our challenge was to make this upgrade as safe as possible. This means no data loss or the upgrade process leaving our K8s cluster in a bad state. We achieved this by keeping a track of our progress during upgrade via a ConfigMap. If for some reason the pre-upgrade job succeeds but post-upgrade job fails. We can run Helm upgrade again and post-upgrade job will know from the configmpa that a backup exists and proceed restoring the data. The configmap is also usual in not running these upgrade jobs again in case an upgrade was already performed earlier.

Advantages

- The key advanatge of this whole exercise was that we did not need an operator or a manual intervention to perform a safe migration.
- Tracking of migration process.
- We can reuse this framework to easily run more complex database upgrade scenarios like changing database vendors.

Always be cautious

- A Kubernetes resource that uses Helm hook, in our case jobs, configmaps, etc., are not managed by Helm release lifecycle. So we have to be careful on how these are created and destroyed.
- This is a solution for a niche problem that we had. This pattern can be used to solve various other problems. However, there are many emerging technologies to manage highly available database instance that takes care of backup, scaling and disruptions.

Conclusion

We came up with a pattern that can solve data migration and problems of similar nature when an application is deployed in Kubernetes using Helm. It is simple, extensible and only requires tools we are already using i.e. Helm and K8s objects.

Frequently Asked Questions

Below is a collection of Frequently Asked Questions from both Marketplace employees and users. This document is updated and expanded periodically for accuracy and competency. For any further questions you'd like answered, please check our Guides, Blog, or Release Notes. Still not seeing your question answered? Send suggestions over to this email.

Marketplace Access and Login Information

1. How do I get access to Marketplace?

- HCL Employees: Login using your current HCL Credentials from the Marketplace Homepage
- All other parties (HCL Customers, HCL Business Partners, or Prospective Customers): Complete and submit this form

2. How can I access the Business Solution or Product?

- Once you launch a sandbox, the login credentials will appear in the top left hand side of the Catalog Details page once fully deployed.

3. What are the initial login credentials for the Sandbox Admin Console?

- User: sol-admin

- Password: Each solution will generate its own unique password during install
- **Note:** These are the initial login credentials for the Sandbox Admin Console, and do not apply if they are reset or changed. If you are unable to login, please contact Marketplace Support.

The Marketplace Catalog

1. What is the Marketplace Catalog?

- The Catalog is all of HCL's Software offerings, in one place. Read about their features and explore capabilities, then launch and test them out in our Marketplace Sandbox environment.

2. In the Marketplace Catalog, what is the difference between a Product and a Business Solution?

- Our catalog offers different types of software - Products and Business Solutions. Products are HCL's cloud native product offerings that stand alone as an HCL entitlement. Business Solutions combine data, configuration, and applications, to be installed with a product to provide a richer, more robust software bundle.

3. How will I know which Product or Business Solution is right for me? Where do I start?

- Information on each HCLSoftware offering is listed on the catalog cards; click on the card to expand and learn more information. Our Catalog has search and filter capabilities that we refine and update regularly. Search for a relevant term in the search box or filter by category, name, or last updated products.

Marketplace Software & Sandbox

1. I just launched my sandbox – why can't I access it yet?

- Your application is still starting up! The Sandbox can take some time to deploy and start up – most solutions are available in just a few minutes, but more complicated applications may require up to an hour before they are ready to use. Check the progress icon in the Status section to see if its fully deployed.

2. Once my sandbox has deployed and I’m in the Sandbox Admin Console, are there any resources I can use to learn about the console and the Marketplace Sandbox?

- Yes – there are Sandbox Admin Console guides to aid your success in testing your solution in the sandbox. On the teal left-hand side navigation bar, there is a section called ‘Guides’ that provides helpful documentation.

3. If I want to install HCLSoftware into the cloud of my choice, how long will it take to download and deploy to my own environment?

- Downloading the associated Helm chart is nearly instantaneous. You can then deploy that Helm chart in your cloud environment of choice.

4. Is there a limit to the number of sandboxes I can launch at once?

- You are allotted 2 active sandboxes at one time.

5. I don’t want to deploy HCLSoftware externally, does HCL have an in-house Kubernetes hosting option?

- HCL offers an in-house native cloud service called HCL Now. Keep your solution and Kubernetes all-in-one place – find out more information.

Resources

1. What are Release Notes and how often are they published?

- Release Notes are Marketplace product updates on our existing offerings. They are located on our Doc site and are released around every 2 weeks. An archive of older release notes is also available.

2. Where can I find resources for Marketplace if I need help?

- Contact your HCL seller or Technical Advisor for assistance with Marketplace. For functional help within the Marketplace platform, refer to our resource Guides.

3. Where can I open a support ticket if I need help with Marketplace?

- Use this link to submit a support ticket.

4. Is there a list of all the HCLSoftware REST APIs?

- Yes - our API Directory is a robust list of all HCLSoftware APIs.

5. There are some terms I don't understand on the Doc site and within the Marketplace site; is there a list of definitions somewhere?

- Our Marketplace Glossary gives you definitions for general cloud-related terms, in addition to Marketplace-specific terms. We are always updating this resource, so check in from time to time.

[Back to Top]

Marketplace Glossary

The below terms are stylized specifically in terms of capitalization, spelling, and formatting. Definition sources and credits are listed at the bottom of the page.

To jump ahead alphabetically, please use the below internal links.

A C D F G H J K M N O P R S U V Y

Term	Definition
A	
API	Application Programming Interface; a set of functions and procedures allowing the creation of applications that access the features or data of an operating system, application, or other service. [1]
API Directory	Marketplace's robust list of APIs available for customer use, indicating which product the API belongs to and the description of its function.
AWS	Amazon Web Services; Amazon's on-demand cloud computing platform and offered API services.
Azure	Microsoft's cloud computing service for building, deploying, and managing applications and servicess.
B	
Business Solution	A catalog offering that demonstrates how product capabilities solve real-world business problems. It integrates one or more HCLSoftware products with configuration and data.
C	

Term	**Definition**
CLI	Command Line Interface; a text-based user interface that processes commands to a computer program in the form of lines of text and command codes.
cloud computing	Aka “the cloud” or servers that are accessed over the Internet, and the software and databases that run on those servers.[2]
cloud native	Technologies that empower organizations to build and run scalable applications in modern, dynamic environments such as public, private, and hybrid clouds. Containers, service meshes, microservices, immutable infrastructure, and declarative APIs are examples of cloud native software.[3]
cloud-ready	An application, software, or service designed to run on a cloud computing framework via the Internet.
containers	A unit of software that bundles software code and enables applications to run rapidly and reliably from one computing environment to another.
D	
Demos	Marketplace Demos provide ways to explore Product capabilities, with scenarios you can try out right within the product itself.
Demo Packs	Marketplace’s Demo Packs contain demo assets like data, configuration, and applications, that can be installed with a Product to provide a richer demo experience. **Note:** Demo Packs automatically bundle the dependent products.
DevOps	A methodology in which teams own an entire process from development to operations.[4]
Docker	An open platform for developing, shipping, and running applications; provides the ability to package and run an application in a loosely isolated environment called a container.[5]
E	
Emissary	Marketplace’s ingress controller for each solution, managing traffic from outside a Kubernetes cluster to services inside the cluster. [7]
G	
GCP	Google Cloud Platform; Google’s cloud computing suite that runs on the same infrastructure as Google does internally.
Grafana	An open source visualization and analytics software that allows you to query, visualize, alert on, and explore your metrics no matter where they are stored; Marketplace has integration capabilities with Grafana.[8]
H	
HCL Now	Our managed cloud service for HCLSoftware; Marketplace’s in-house “cloud native-as-a-Service” offering.

Term	**Definition**
HCLSoftware Marketplace	Short for ‘Solution Factory’ – a platform that utilizes Helm technology to combine HCL Products and APIs as cloud-ready building blocks into portable, deployable solutions.
Helm chart	A Helm package that contains information sufficient for installing a set of Kubernetes resources into a Kubernetes cluster. [9]
J	
json	Stands for: JavaScript Object Notation; a minimalist readable data format primarily used to transmit data between an application and server.
K	
Keycloak	An open source ‘Identity’ and ‘Access Management’ solution aimed at modern applications and services; Marketplace has integration capabilities with Keycloak. [10]
kubectl	The command-line tool that allows you to run commands against Kubernetes clusters, deploy applications, inspect and manage cluster resources, and view logs. [11]
Kubernetes	A portable, extensible, open source orchestration platform for managing containerized workloads and services, that facilitates both declarative configuration and automation. [12]
N	
node pools	A set of nodes within a Kubernetes cluster that share the same configuration (range, machine type, etc).
O	
on-prem	Aka on-premises software; software that is housed on a physical location and operates on a local server and computing infrastructure, opposed to being hosted on the cloud or via a remote facility.
open source	Applications, software, or services that are publicly accessible, thus able to be utilized, modified, or shared by anyone.
P	
preview content	Solutions built with preview content from the Marketplace Catalog can be built and deployed in Marketplace sandbox; however they can't be downloaded.
Products	Marketplace’s cloud native, complete HCLSoftware product offerings, in a Kubernetes-ready format.
Prometheus	An open source systems monitoring and alerting toolkit that includes built-in and active scraping, storing, querying, graphing, and alerting based on time series data; Marketplace has integration capabilities with Prometheus. [13]
R	

Term	**Definition**
Release Notes	Marketplace’s bi-weekly product update notation, providing users with useful details on software improvements, bug fixes, and feature rollouts.
REST API	Representational State Transfer (REST) Application Programming Interface (API); enables two computer systems to communicate over HTTP in a similar way to web servers or browsers.
S	
Marketplace Blog	Marketplace's technical blog, including a range of informational topics presented in various, digestible formats - made for a range of audiences, from the Marketplace user to the general cloud native developer audience.
Marketplace Catalog	Marketplace’s robust list of pre-built cloud native solutions that users can explore and test in our Sandbox environment.
Marketplace Common Services	Marketplace’s pre-integrated services; e.g. - integrations with Grafana, Keycloak, or Prometheus; use of Access Control Service (ACS), optional Monitoring Dashboard.
Marketplace FAQ	Marketplace's list of most frequently asked questions.
Sandbox Admin Console	Marketplace’s included dashboard to monitor health, logs, pods, etc. of your solution.
Marketplace Glossary	This list! Marketplace’s evergreen list of terms and definitions helpful for Marketplace customers and general cloud native audiences.
Marketplace Guide	Marketplace’s instructional resources for more information on specific and technical Marketplace topics; Marketplace Guides are written in a formal and directional format with specific topics and instructions.
Marketplace Sandbox	Marketplace’s in-house deployment testing environment – test out, modify, build and rebuild your solution for hours or weeks at a time.
Marketplace Solution	A pre-built, cloud native product offering selected from the Marketplace Catalog packaged into a unique Helm chart solution, ready for deployment into any cloud environment.
SSL certificate	A digital certificate that authenticates a website’s identity and enables an encrypted connection to the site.
SQL	Standard Query Language; SQL is the most common language for extracting and organizing data stored in a relational database.
Swagger UI	Swagger is an Interface Description Language for describing RESTful APIs expressed using json.
U	
UI/UX	User Interface/User Experience – both terms focus on the design of a user interface for machines and software with a focus on amplifying usability.

Term	Definition
V	
virtual machine	A software-based computer that exists within another computer’s operating system, often used for the purposes of testing, backing up data, or running SaaS applications.
Y	
yaml	YAML Ain’t Markup Language; a digestible data serialization language commonly used in applications and files where data is stored or transmitted.

References

1. Definition source: Oxford Languages and Google

2. Definition source: Cloudfare

3. Definition source: Cloud Native Computing Foundation (CNCF)

4. Definition source: Cloud Native Computing Foundation (CNCF)

5. Definition source: Docker

6. Definition source: Ambassador

7. Definition source: Grafana

8. Definition source: Helm documentation

9. Definition source: Keycloak

10. Definition source: Kubernetes documentation

11. Definition source: Kubernetes documentation

12. Definition source: Prometheus documentation

13. Definition source: Cloudfare