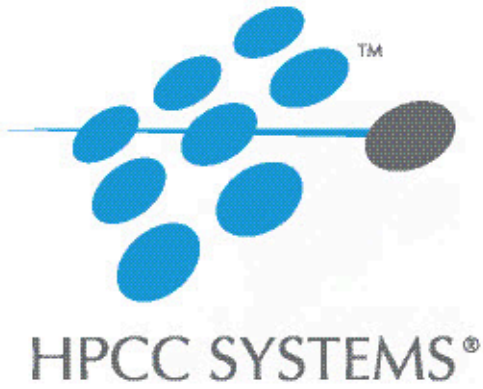
Containerized HPCC Systems® Platform

Equipe de documentação de Boca Raton



Containerized HPCC Systems® Platform

Equipe de documentação de Boca Raton

Sua opinião e comentários sobre este documento são muito bem-vindos e podem ser enviados por e-mail para <docfeedback@hpccsystems.com>

Inclua a frase **Feedback sobre documentação** na linha de assunto e indique o nome do documento, o número das páginas e número da versão atual no corpo da mensagem.

2021 Version 8.2.80-1

Visão geral do HPCC em contêineres

A partir da versão 8.0, a plataforma HPCC Systems® focará em deploys em contêineres. Isso é útil para implantações baseadas em nuvem (grandes ou pequenas) ou implantações de teste/desenvolvimento locais.

Os contêineres do Docker gerenciados pelo Kubernetes (K8s) são um novo ambiente operacional de destino, juntamente com o suporte contínuo para instalações tradicionais "bare metal" usando arquivos do instalador .deb ou .rpm. O suporte para instaladores tradicionais continua e esse tipo de implantação é viável para implantações bare metal ou configurações manuais na nuvem.

Esta não é uma mudança do tipo *"rehosting"*, em que a plataforma executa sua estrutura legada inalterada e trata os contêineres apenas como uma forma de fornecer *máquinas virtuais* e para serem executadas, mas uma mudança significativa em como os componentes são configurados, como e quando eles iniciam e onde armazenam seus dados.

Este livro se concentra em implantações em contêineres. A primeira seção é sobre o uso de contêineres Docker e gráficos Helm localmente. Docker e Helm fazem muito do trabalho para você. A segunda parte usa as mesmas técnicas na nuvem.

Para pequenas implantações locais (para desenvolvimento e teste), sugerimos o uso de Docker Desktop e Helm. Isto é útil para aprendizagem, desenvolvimento e teste.

Para implantações em nuvem, você pode usar qualquer tipo de serviço em nuvem, se for compatível com Docker, Kubernetes e Helm. Este livro, no entanto, se concentrará no Microsoft Azure para serviços em nuvem. As versões futuras podem incluir especificações para outros provedores de nuvem.

Se você deseja gerenciar manualmente sua implantação local ou na nuvem, ainda pode usar os instaladores tradicionais e o Configuration Manager, mas isso remove muitos dos benefícios que o Docker, Kubernetes e Helm fornecem, como instrumentação, monitoramento, escalonamento e custo ao controle.

Os sistemas HPCC seguem as convenções padrão sobre como as implantações do Kubernetes são normalmente configuradas e gerenciadas, portanto, deve ser fácil para alguém familiarizado com o Kubernetes e o Helm instalar e gerenciar a plataforma HPCC Systems.

Note: A versão tradicional bare-metal da plataforma de sistemas HPCC está madura e tem sido amplamente usada em aplicativos comerciais por quase duas décadas e é totalmente destinada para uso em produção. A versão em contêiner é nova e ainda não está 100% pronta para produção. Além disso, alguns aspectos dessa versão podem ser alterados sem aviso prévio. Nós encorajamos você a usá-lo e fornecer feedback para que possamos tornar esta versão tão robusta quanto uma instalação bare-metal.

Bare-metal vs Containers

Se você estiver familiarizado com a plataforma HPCC Systems, há algumas mudanças fundamentais a serem observadas.

Processoss e pods, não máquinas

Qualquer pessoa familiarizada com o sistema de configuração existente saberá que parte da configuração envolve a criação de instâncias de cada processo e a especificação de quais máquinas físicas devem ser executadas.

Em um mundo Kubernetes, isso é gerenciado dinamicamente pelo próprio sistema K8s (e pode ser alterado dinamicamente enquanto o sistema é executado).

Além disso, um sistema em contêiner é muito mais simples de gerenciar se você adotar o paradigma de um processo por contêiner, em que as decisões sobre quais contêineres precisam ser agrupados em um pod e quais pods podem ser executados em nós físicos de maneira automática.

Helm charts

No mundo em contêineres, as informações que o operador precisa fornecer para configurar um ambiente HPCC Systems são bastante reduzidas. Não há necessidade de especificar qualquer informação sobre quais máquinas estão em uso e por qual processo. Conforme mencionado acima, também não há necessidade de alterar muitas opções que podem ser dependentes do ambiente operacional, uma vez que muito disso foi padronizado no momento em que as imagens do contêiner foram criadas.

Portanto, na maioria dos casos, a maior parte das configurações devem ser ignoradas para usar o padrão. Como tal, o novo paradigma de configuração requer que apenas o mínimo de informações seja especificado e quaisquer parâmetros não especificados façam usos dos padrões apropriados.

O **environment.xml** padrão que incluímos em nossos pacotes bare-metal para descrever o sistema de nó único padrão contém aproximadamente 1300 linhas e é complexo o suficiente para que recomendamos o uso de uma ferramenta especial para editá-lo.

O **values.yaml** do helm chart padrão tem menos de 100 linhas e pode ser editado em qualquer editor e/ou modificado por meio das substituições da linha de comando. Também é autodocumentado com comentários extensos.

Static vs On-Demand Services

A fim de realizar a economia de custo potencial de um ambiente de nuvem e, ao mesmo tempo, aproveitar a escalabilidade quando necessário, alguns serviços que estão sempre ativos na tradição de instalações bare-metal são lançados sob demanda em instalações em contêiner.

Por exemplo, um componente eclccserver inicia um stub que requer recursos mínimos, onde a única tarefa é observar as workunits enviadas para compilação e lançar um job K8s independente para realizar a compilação atual.

Da mesma forma, o componente eclagent também é um stub que ativa um job K8s quando uma workunit é enviada e o stub Thor inicia um cluster apenas quando necessário. Usando esse design, não apenas a capacidade do sistema aumenta automaticamente para usar quantos pods forem necessários para lidar com a carga enviada, como também diminui para usar recursos mínimos (como uma fração de um único nó) durante os tempos de inatividade quando aguardando que os trabalhos sejam enviados.

Os componentes ESP e Dali estão sempre ligados, desde que o cluster K8s seja iniciado. Não é viável iniciá-los e interrompê-los sob demanda sem latência excessiva. No entanto, o ESP pode ser ampliado e reduzido dinamicamente para executar quantas instâncias forem necessárias para lidar com a carga atual.

Configurações de topologia – Clusters vs filas

Em implantações bare-metal, há uma seção chamada **Topologia** onde as várias filas às quais as workunits podem ser enviadas são configuradas. É responsabilidade da pessoa que edita o ambiente garantir que cada destino nomeado tenha as instâncias eclccserver, hThor (ou ROXIE) e Thor (se desejado) configuradas, para lidar com as workunit enviadas para a fila de destino.

Essa configuração foi bastante simplificada ao usar Helm charts para configurar um sistema em contêiner. Cada Thor nomeado ou componente eclagent cria uma fila correspondente (com o mesmo nome) e cada eclccserver escuta em todas as filas por padrão (mas você pode restringir a certas filas apenas se realmente quiser). Definir um componente do Thor automaticamente garante que os componentes do agente necessários sejam provisionados.

Deploy Local (Desenvolvimento e Teste)

Embora haja muitas maneiras de instalar uma plataforma HPCC Systems de nó único local, esta seção se concentra no uso do Docker Desktop.

Pré-requisitos

- Instalar Docker Desktop e **WSL 2**
- Habilitar integração Enable WSL no Docker
- Habilitar Kubernetes no Docker Desktop
- Instalar Helm

OU

- Instalar Docker Desktop e **Hyper-V**
- Habilitar Kubernetes no Docker Desktop
- Instalar Helm

OU

- Instalar Docker Desktop no macOS
- Habilitar Kubernetes no Docker Desktop
- Instalar Helm

Adicionar repositório

Para usar o helm charts do HPCC Systems, você deve adicioná-lo à lista de repositório do helm, conforme mostrado abaixo:

```
helm repo add hpcc https://hpcc-systems.github.io/helm-chart/
```

Reposta esperada:

```
"hpcc" has been added to your repositories
```

Para atualizar os último charts:

```
helm repo update
```

Resposta esperada:

```
Update Complete. Happy Helming!
```

Iniciar um sistema padrão

O helm chart padrão inicia um sistema de teste simples com Dali, ESP, eclccserver, duas filas eclagent (modo ROXIE e hThor) e uma fila Thor.

Para iniciar este sistema simples:

```
helm install mycluster hpcc/hpcc --version=8.2.2
```

O argumento --version é opcional, mas recomendado. Ele garante que você saiba a versão que você está instalando. Se omitido, a última versão não-desenvolvimento será instalada. Este exemplo usa 8.2.2, mas você deve usar a versão que deseja.

Resposta esperada:

```
NAME: mycluster
LAST DEPLOYED: Tue Mar 23 13:26:55 2021
NAMESPACE: default
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
Thank you for installing the HPCC chart.

This chart has defined the following HPCC components:
dali.mydali
dfuserver.dfuserver
eclagent.hthor
eclagent.roxie-workunit
eclccserver.myeclccserver
esp.eclwatch
esp.eclservices
esp.eclqueries
esp.esdl-sandbox
esp.sql2ecl
roxie.roxie
thor.thor
sasha.dfurecovery-archiver
sasha.dfuwu-archiver
sasha.file-expiry
sasha.wu-archiver
```

Para verificar o status:

```
kubectl get pods
```

Resposta esperada:

```
NAME                                    READY   STATUS    RESTARTS   AGE
eclqueries-7fd94d77cb-m7lmb             1/1     Running   0          2m6s
eclservices-b57f9b7cc-bhwtm             1/1     Running   0          2m6s
eclwatch-599fb7845-2hq54                1/1     Running   0          2m6s
esdl-sandbox-848b865d46-9bv9r           1/1     Running   0          2m6s
hthor-745f598795-ql9dl                  1/1     Running   0          2m6s
mydali-6b844bfcfb-jv7f6                 2/2     Running   0          2m6s
myeclccserver-75bcc4d4d-gflfs           1/1     Running   0          2m6s
roxie-agent-1-77f696466f-tl7bb          1/1     Running   0          2m6s
roxie-agent-1-77f696466f-xzrtf          1/1     Running   0          2m6s
roxie-agent-2-6dd45b7f9d-m22wl          1/1     Running   0          2m6s
roxie-agent-2-6dd45b7f9d-xmlmk          1/1     Running   0          2m6s
roxie-toposerver-695fb9c5c7-9lnp5       1/1     Running   0          2m6s
```

```
roxie-workunit-d7446699f-rvf2z                1/1     Running   0          2m6s
sasha-dfurecovery-archiver-78c47c4db7-k9mdz   1/1     Running   0          2m6s
sasha-dfuwu-archiver-576b978cc7-b47v7         1/1     Running   0          2m6s
sasha-file-expiry-8496d87879-xct7f            1/1     Running   0          2m6s
sasha-wu-archiver-5f64594948-xjblh            1/1     Running   0          2m6s
sql2ecl-5c8c94d55-tj4td                       1/1     Running   0          2m6s
thor-eclagent-6b8f564f9c-qnczz                1/1     Running   0          2m6s
thor-thoragent-56d788869f-7trxk               1/1     Running   0          2m6s
```

Observação: Pode demorar um pouco antes de todos os componentes estarem em execução, especialmente na primeira vez, pois as imagens do contêiner precisam ser baixadas do Docker Hub.

Uso padrão do sistema

Seu sistema agora está pronto para uso. O primeiro passo usual é abrir o ECL Watch.

Observação: Algumas páginas no ECL Watch, como aquelas que exibem informações de topologia, ainda não estão totalmente funcionais no modo em contêiner.

Use este comando para obter uma lista de serviços em execução e endereços IP:

```
kubectl get svc
```

Resposta esperada:

```
NAME                  TYPE         CLUSTER-IP      EXTERNAL-IP  PORT(S)           AGE
eclqueries            LoadBalancer 10.108.171.35   localhost    8002:31615/TCP    2m6s
eclservices           ClusterIP    10.107.121.158  <none>       8010/TCP          2m6s
eclwatch              LoadBalancer 10.100.81.69    localhost    8010:30173/TCP    2m6s
esdl-sandbox          LoadBalancer 10.100.194.33   localhost    8899:30705/TCP    2m6s
kubernetes            ClusterIP    10.96.0.1       <none>       443/TCP           2m6s
mydali                ClusterIP    10.102.80.158   <none>       7070/TCP          2m6s
roxie                 LoadBalancer 10.100.134.125  localhost    9876:30480/TCP    2m6s
roxie-toposerver      ClusterIP    None            <none>       9004/TCP          2m6s
sasha-dfuwu-archiver  ClusterIP    10.110.200.110  <none>       8877/TCP          2m6s
sasha-wu-archiver     ClusterIP    10.111.34.240   <none>       8877/TCP          2m6s
sql2ecl               LoadBalancer 10.107.177.180  localhost    8510:30054/TCP    2m6s
```

Observe que o serviço **eclwatch** está sendo executado em **localhost:8010**. Use esse endereço em seu navegador para acessar o ECL Watch.

Dentro do ECL Watch, pressione o botão ECL e em seguida vá para a aba Playground.

A partir daqui, você pode usar o ECL de exemplo ou inserir outras consultas de teste e escolher entre os clusters disponíveis para enviar suas workunit.

Terminar (Descomissionar) o sistema

Para verificar quais helm charts estão instalados atualmente, execute este comando:

```
helm list
```

Para interromper os pods do HPCC Systems, use o helm para desinstalar:

```
helm uninstall mycluster
```

Isso interrompe o cluster, exclui os pods e, com as configurações padrão e os volumes persistentes, também exclui o armazenamento usado.

Storage

Persistent Storage para um Deploy Local

Ao executar em um sistema de teste de nó único, como Docker Desktop, a classe de armazenamento padrão normalmente significa que todas as declarações de volume persistente (PVCs) são mapeadas para diretórios locais temporários na máquina host. Normalmente, eles são removidos quando o cluster é interrompido. Isso é bom para teste, mas para qualquer aplicativo real, você deseja o persistent storage.

Para persistir os dados com o Docker Desktop, a primeira etapa é garantir que os diretórios relevantes existam:

1. Crie diretórios de dados usando uma interface de terminal:

   Para Windows, use o seguinte comando:

   ```
   mkdir c:\hpccdata
   mkdir c:\hpccdata\dalistorage
   mkdir c:\hpccdata\queries
   mkdir c:\hpccdata\sasha
   mkdir c:\hpccdata\hpcc-data
   mkdir c:\hpccdata\mydropzone
   ```

   Para macOS, use este comando:

   ```
   mkdir -p /Users/myUser/hpccdata/{dalistorage,queries,sasha,hpcc-data,mydropzone}
   ```

2. Faça o download do HPCC Platform Helm charts.

   Eles estão disponíveis no repositório do **HPCC-Platform** do HPCC Systems no GitHub (https://github.com/hpcc-systems/HPCC-Platform).

   Se você quiser somente o helm charts, use o repositório **helm-chart** (https://github.com/hpcc-systems/helm-chart).

3. Abra uma janela de prompt e navegue até diretório **helm** do reposítório que você acabou de fazer o download.

4. Instalar o Helm chart do diretório **examples/local** no seu repositório local.

   Este chart cria volumes persistents baseada nos diretórios host que você criou anteriormente.

   ```
   # for a WSL2 deployment:
   helm install hpcc-localfile examples/local/hpcc-localfile
              --set common.hostpath=/run/desktop/mnt/host/c/hpccdata

   # for a Hyper-V deployment:
   helm install hpcc-localfile examples/local/hpcc-localfile --set common.hostpath=/c/hpccdata

   # for a macOS deployment:
   helm install hpcc-localfile examples/local/hpcc-localfile --set common.hostpath=/Users/myUser/hpccdata
   ```

   A opção **--set common.hostpath=** especifica o diretório base:

   O caminho **/run/desktop/mnt/host/c/hpccdata** fornece acesso ao arquivo de host do sistema para WSL2.

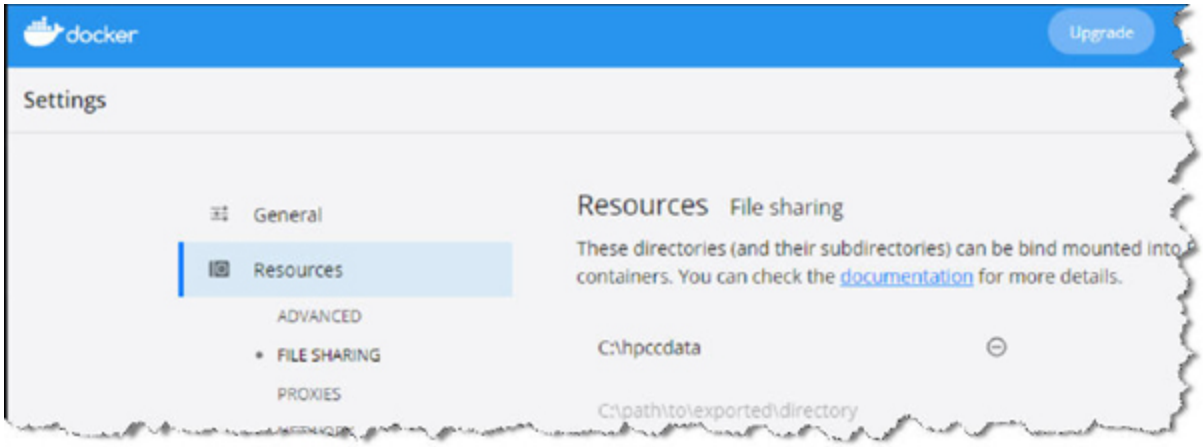
   O caminho **/c/hpccdata** fornece acesso ao arquivo de host do sistema para Hyper-V.

O caminho **/Users/myUser/hpccdata** fornece acesso ao arquivo de host do sistema para Mac OSX.

Observação: O valor passado para --set common-hostpath diferencia maiúsculas de minúsculas.

5. Se você estiver usando o Docker Desktop com Hyper-V, adicione o diretório dados compartilhado (no exemplo, C:\hpccdata) nas configurações do Docker Desktop.

   Isto **não** é necessário em ambientes macOS ou WSL 2.



6. Por fim, instale o hpcc Helm chart e forneça um arquivo yaml que fornece informações de armazenamento que usa os PVCs criados na etapa anterior.

   O diretório de exemplo contém um arquivo yaml de amostra que pode ser usado neste caso:

```
helm install mycluster hpcc/ --version=8.2.2
             -f examples/local/values-localfile.yaml
```

   O argumento --version é opcional, mas recomendado. Ele garante que você saiba a versão que você está instalando. Se omitido, a última versão não-desenvolvimento será instalada. Este exemplo usa 8.2.2, mas você deve usar a versão que deseja.

7. Para testar, crie alguns arquivos de dados e workunitso enviando ao Thor algum código ECL como o seguinte:

```
LayoutPerson := RECORD
  UNSIGNED1 ID;
  STRING15  FirstName;
  STRING25  LastName;
END;
allPeople := DATASET([ {1,'Fred','Smith'},
                       {2,'Joe','Jones'},
                       {3,'Jane','Smith'}],LayoutPerson);
OUTPUT(allPeople,,'MyData::allPeople',THOR,OVERWRITE);
```

8. Use o comando helm uninstall para encerrar seus clusters e, em seguida, reinicie.

9. Abra o ECL Watch e observe que suas workunits e os arquivos lógicos ainda estão lá.

Importar: Storage Plans e como usá-los

Os Storage Plans fornecem a flexibilidade para configurar onde os dados são armazenados em uma plataforma HPCC Systems, mas não aborda diretamente a questão de como obter dados para a plataforma em primeiro lugar.

As plataformas em contêiner oferecem suporte à importação de dados de duas maneiras:

- Fazendo o upload para uma Landing Zone e o Spray (não implementado na versão em contêiner)
- Copiando o Storage Plane e acessando diretamente

A partir da versão 7.12.0, uma nova sintaxe ECL foi adicionada para acessar arquivos diretamente de um plano de armazenamento. Isso é semelhante a sintax **file::** sintaxe usada para ler arquivos diretamente de uma máquina física, normalmente uma landing zone.

A nova sintaxe é:

```
~plane::<storage-plane-name>::<path>::<filename>
```

Onde a sintaxe do caminho e do nome do arquivo são as mesmas usadas com a sintex **file::**. Isso inclui exigir que letras maiúsculas sejam citadas com um símbolo ^. Para maiores detalhes, veja a seção Arquivos de Landing Zone Files no documento *ECL Language Reference.*

Se você o plano de armazenamento configurado conforme a seção anterior e você copiou o arquivo **originalperson** para **C:\hpccdata\hpcc-data\tutorial**, então referenciaro arquivo utilizando o sintax

```
'~plane::data::tutorial::originalperson'
```

Observação: O arquivo **originalperson** está disponível na página do HPCC Systems (https://cdn.hpccsystems.com/install/docs/3_8_0_8rc_CE/OriginalPerson)