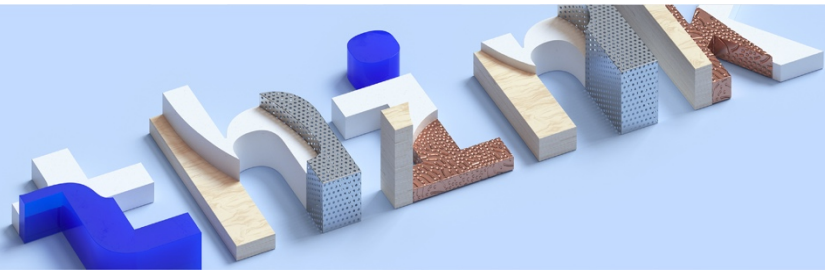


think 2018



Lab Center – Hands-on Lab

Session 3370

**Session Title Kubernetes on IBM Power Systems
with IBM Cloud Private**

Khalid Ahmed, IBM, khalida@ca.ibm.com

Table of Contents

Disclaimer	3
Introduction	5
IBM Cloud Private and Kubernetes	6
Part 1: Overview	6
Part 2: Setting up or accessing your cluster	6
Managing deployments in an IBM Cloud Private cluster	7
Part 3: Creating deployments	7
Part 4: Scaling deployments	7
Manually scaling deployments	7
Optional: Creating Auto-scaling policies	8
Monitoring the status of your cluster and applications	9
Managing and troubleshooting your cluster using the Kubernetes CLI	11
Part 5: Setting up the Kubernetes CLI	11
Part 6: Using the Kubernetes CLI	11
Automating the build process for Docker images	12
Part 7: Accessing the Jenkins dashboard	12
Part 8: Setting up task automation with Jenkins	13
Deploying a microservices based application on a Kubernetes cluster	18
Part 9: Deploying the Acme air app	18
Part 10: Accessing the Acme air app	20
Part 11: (Optional) Building the Acme air Helm chart	21
Appendix – useful links	27
We Value Your Feedback!	28

Disclaimer

IBM's statements regarding its plans, directions, and intent are subject to change or withdrawal without notice at IBM's sole discretion. Information regarding potential future products is intended to outline our general product direction and it should not be relied on in making a purchasing decision.

The information mentioned regarding potential future products is not a commitment, promise, or legal obligation to deliver any material, code or functionality. Information about potential future products may not be incorporated into any contract.

The development, release, and timing of any future features or functionality described for our products remains at our sole discretion I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve results like those stated here.

Information in these presentations (including information relating to products that have not yet been announced by IBM) has been reviewed for accuracy as of the date of initial publication and could include unintentional technical or typographical errors. IBM shall have no responsibility to update this information. **This document is distributed "as is" without any warranty, either express or implied. In no event, shall IBM be liable for any damage arising from the use of this information, including but not limited to, loss of data, business interruption, loss of profit or loss of opportunity.** IBM products and services are warranted per the terms and conditions of the agreements under which they are provided.

IBM products are manufactured from new parts or new and used parts.

In some cases, a product may not be new and may have been previously installed. Regardless, our warranty terms apply."

Any statements regarding IBM's future direction, intent or product plans are subject to change or withdrawal without notice.

Performance data contained herein was generally obtained in controlled, isolated environments. Customer examples are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual performance, cost, savings or other results in other operating environments may vary.

References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business.

Workshops, sessions and associated materials may have been prepared by independent session speakers, and do not necessarily reflect the views of IBM. All materials and discussions are provided for informational purposes only, and are neither intended to, nor shall constitute legal or other guidance or advice to any individual participant or their specific situation.

It is the customer's responsibility to insure its own compliance with legal requirements and to obtain advice of competent legal counsel as to the identification and interpretation of any relevant laws and regulatory requirements that may affect the customer's business and any actions the customer may

need to take to comply with such laws. IBM does not provide legal advice or represent or warrant that its services or products will ensure that the customer follows any law.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products about this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products. IBM does not warrant the quality of any third-party products, or the ability of any such third-party products to interoperate with IBM's products. **IBM expressly disclaims all warranties, expressed or implied, including but not limited to, the implied warranties of merchantability and fitness for a purpose.**

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents, copyrights, trademarks or other intellectual property right.

IBM, the IBM logo, ibm.com and [names of other referenced IBM products and services used in the presentation] are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at: www.ibm.com/legal/copytrade.shtml.

© 2018 International Business Machines Corporation. No part of this document may be reproduced or transmitted in any form without written permission from IBM.

U.S. Government Users Restricted Rights — use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM.

Introduction

The Kubernetes project grew out of the world of online micro-services-based applications as an orchestration framework for containerized applications. With the explosive momentum of Docker containers, Kubernetes has become the de-facto standard for orchestrating and managing containerized apps in production environments. In this workshop we will demonstrate how Kubernetes can run on Power systems, using IBM Cloud Private, a Kubernetes based container management stack.

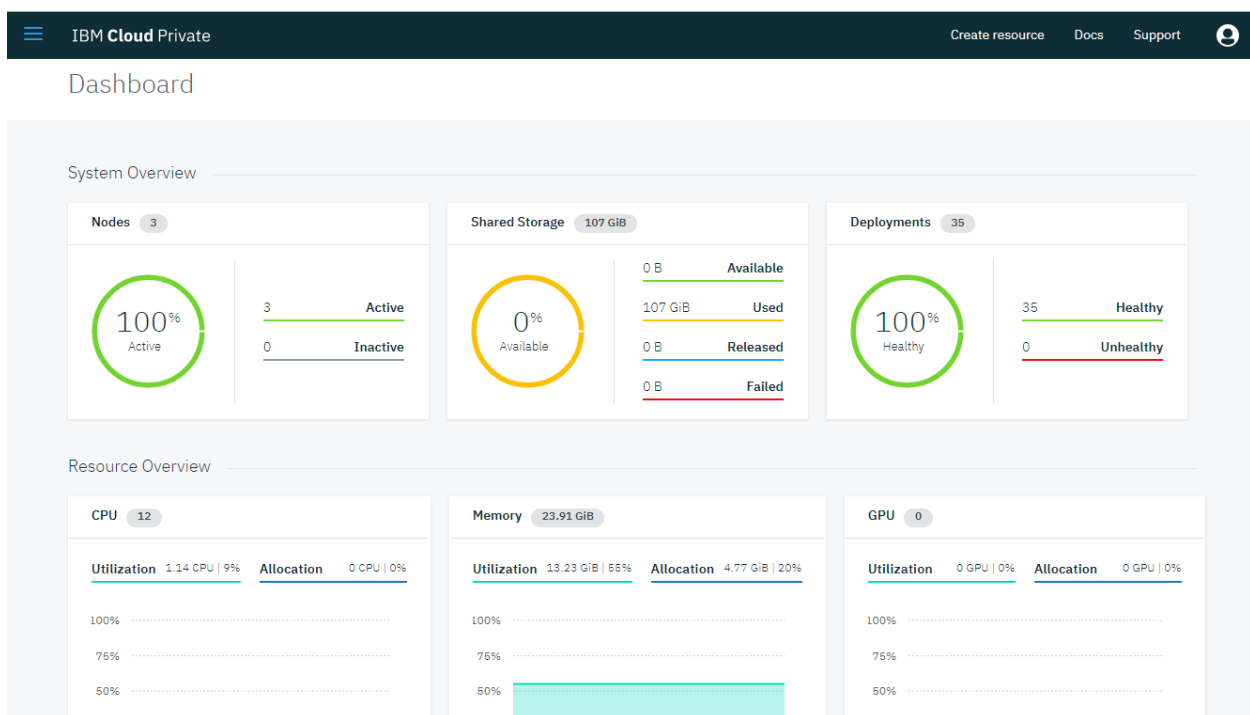
You will gain access to IBM Cloud Private on Power-based systems on the cloud as well as on a developer laptop. You will also get a chance to onboard a sample application to Kubernetes and learn about features of IBM Cloud Private through hands on training.

IBM Cloud Private and Kubernetes

Part 1: Overview

IBM Cloud Private is a server platform for developing and managing on-premises, containerized applications.

It is an integrated environment for managing containers that includes the container orchestrator Kubernetes, a private image repository, a management console, and monitoring frameworks.



Part 2: Setting up or accessing your cluster

Instructions on setting up or accessing your cluster will be provided by your lab instructor.

Managing deployments in an IBM Cloud Private cluster

Part 3: Creating deployments

- __ 1. Log in to the IBM Cloud Private management console.
- __ 2. From the navigation menu, select **Catalog > Helm charts**.
- __ 3. From the Helm charts catalog, locate the **ibm-websphere-liberty** deployment.
- __ 4. Review the readme for requirements and configuration settings.
- __ 5. Click **Configure**.
- __ 6. Provide a **Release name**. For example, “demo”.
- __ 7. Select the namespace that matches your username. For example, if you are user1, use the namespace1 namespace.
- __ 8. Accept the license agreements.
- __ 9. Click **Install**.
- __ 10. Click **View Helm release**. Then click on **demo**. Review information about the services, deployments and pods that are created.
- __ 11. From the navigation menu, select **Workloads > Deployments > demo-ibm-websphere-liberty**. Review additional information about the deployment including details about pods, metrics, logs, and events.

From the **Expose details** section, you can find the endpoint link for accessing your deployment.

Note: Replace http with https in the endpoint link to access the **websphere-liberty** service.

Part 4: Scaling deployments

Manually scaling deployments

- __ 1. From the navigation menu, select **Workloads > Deployments**.
- __ 2. For the deployment that you want to scale, select **Action > Scale**. A Scale Deployment form displays.
- __ 3. Enter the number of pods required.

- If the number of pods entered is less than the current Desired Replica, a scale in action is triggered.
- If the number of pods entered is greater than the current Desired Replica, a scale-out action is triggered.

__ 4. Click **Scale Deployment**.

Optional: Creating Auto-scaling policies

__ 1. From the navigation menu, select **Configuration > Scaling Policies**.

__ 2. Click **Create Policy**.

__ 3. Enter the policy details. Policy details can be provided either in a JSON format or by completing the fields in the **Create Policy** form.

__ 4. To create a policy, the following parameters are required:

- A policy name
- A target - the name of the application that will use this policy
- Maximum number of replications - this value is the maximum number of replications that are allowed during a scale up

__ 5. You can also set the following optional parameters:

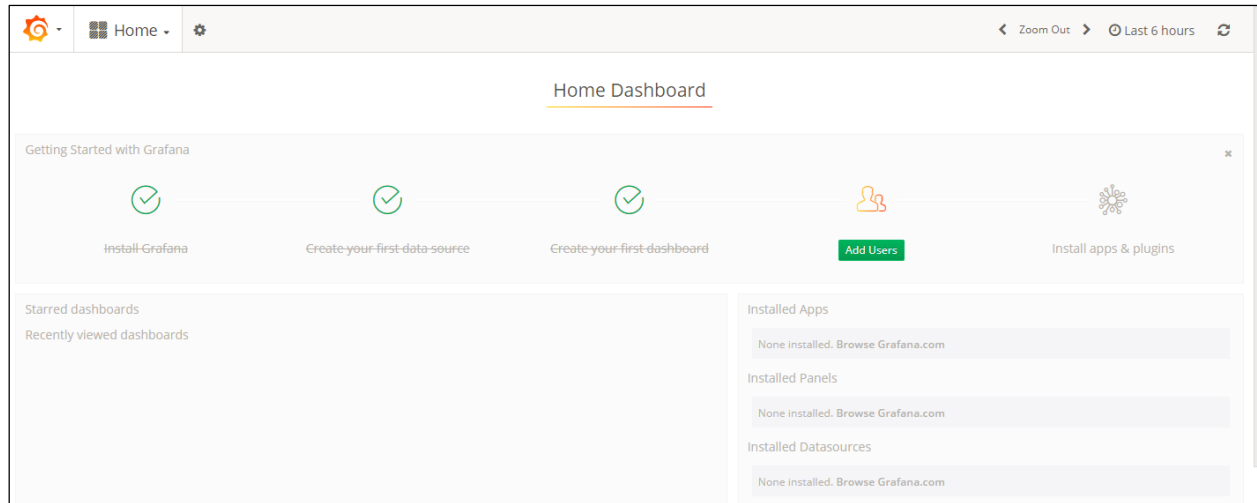
- Minimum number of replications, which has a default of 1, is optional.
- Specifying the resource limit for CPU is optional. If you do not specify the resource limits for your container, the values are set to unlimited.

__ 6. Click **Create**.

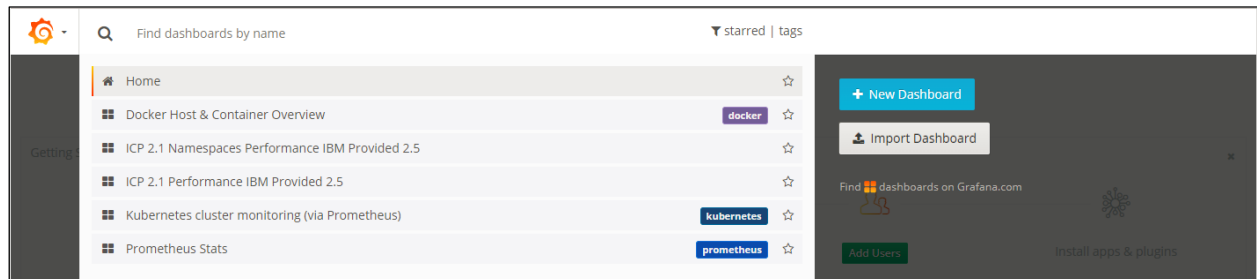
Monitoring the status of your cluster and applications

The monitoring dashboard uses Grafana and Prometheus to present detailed data about your cluster nodes and containers

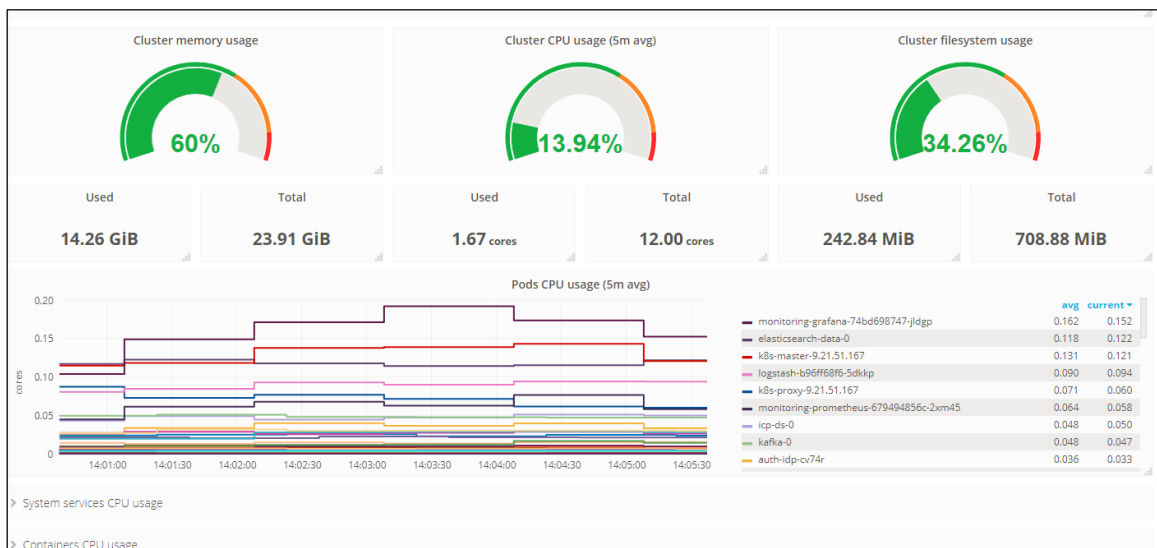
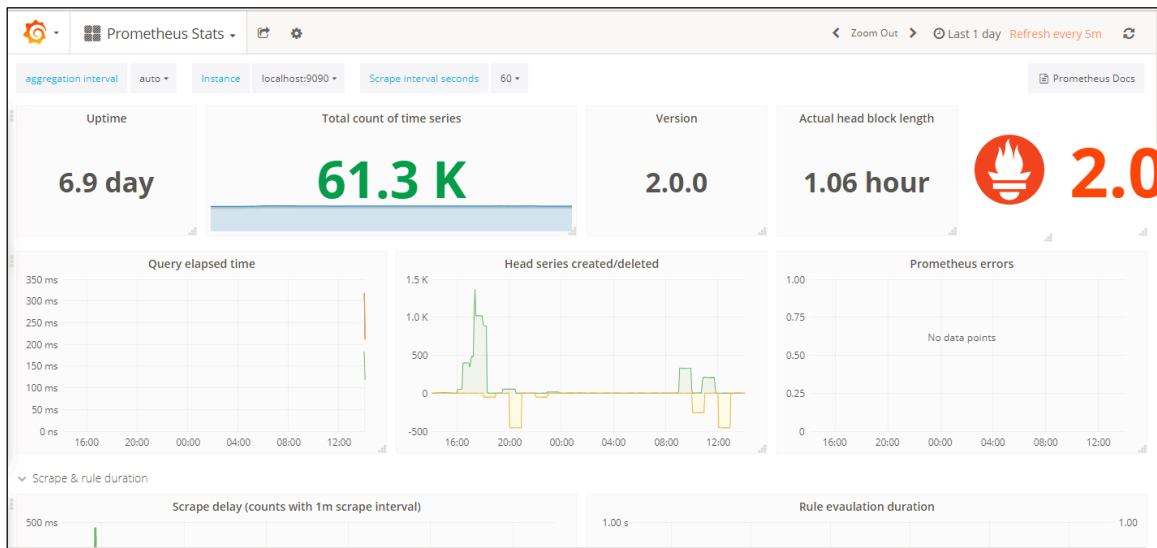
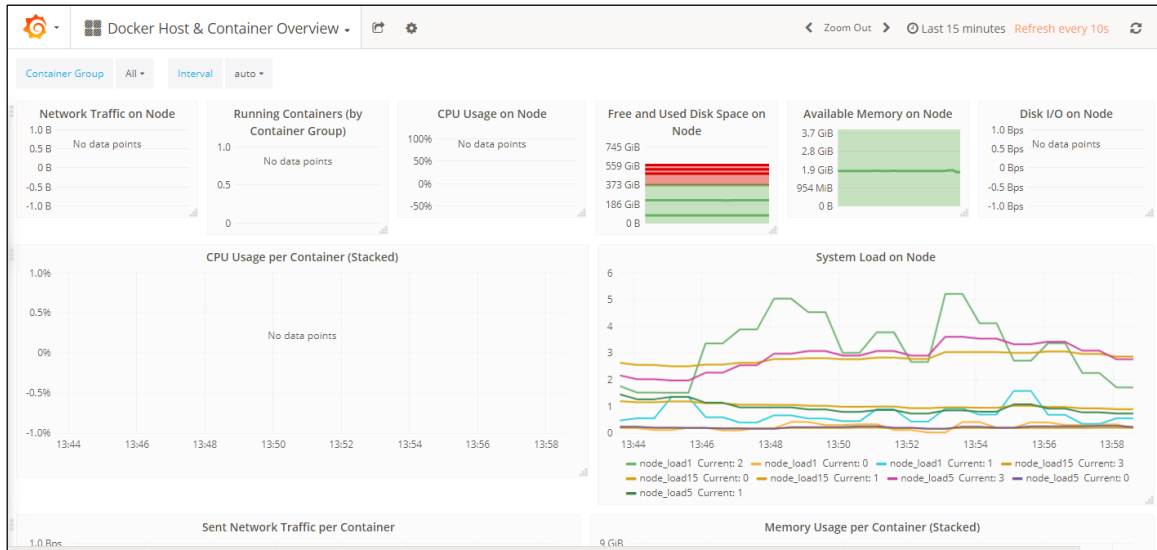
__ 1. From the navigation menu, select **Platform > Monitoring**.



__ 2. Click the drop-down icon located next to **Home**.



__ 3. Click on each available dashboard to review the status information on various system components and processes.



Managing and troubleshooting your cluster using the Kubernetes CLI

Part 5: Setting up the Kubernetes CLI

__1. Install kubectl for IBM Cloud Private.

```
docker run -e LICENSE=accept --net=host -v /usr/local/bin:/data ibmcom/icp-  
inception-ppc64le:2.1.0.1-ee cp /usr/local/bin/kubectl /data
```

__2. From the dashboard view, select **User Account > Configure Client**.

The user account tab can be found in the upper right corner of your screen. This tab displays the user name for the logged-on user.

__3. Copy and paste the configuration information to your command line and press **Enter**.

Part 6: Using the Kubernetes CLI

```
kubectl get nodes  
kubectl describe node <node>  
kubectl get pods  
kubectl describe pod <pod>  
kubectl get namespaces  
kubectl describe namespace <ns>
```

For more information about using kubectl with IBM Cloud Private, see [Accessing your IBM® Cloud Private cluster by using the kubectl CLI](#).

Automating the build process for Docker images

A typical CI/CD pipeline uses the following process:

- Developers commit code to a SCM, which triggers a Jenkins build pipeline.
- Jenkins then builds the code and runs tests.
- If the tests pass, the code is deployed to various environments, such as a staging server and then fully tested and verified content is moved to a production server.

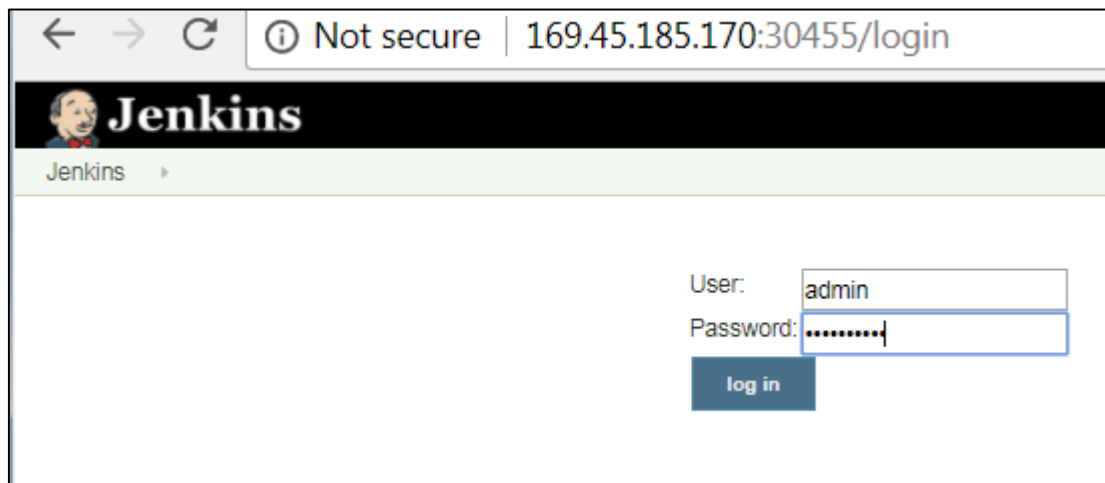
In this exercise, you will set up a pipeline to build and automatically push Docker images to the IBM Cloud Private image registry.

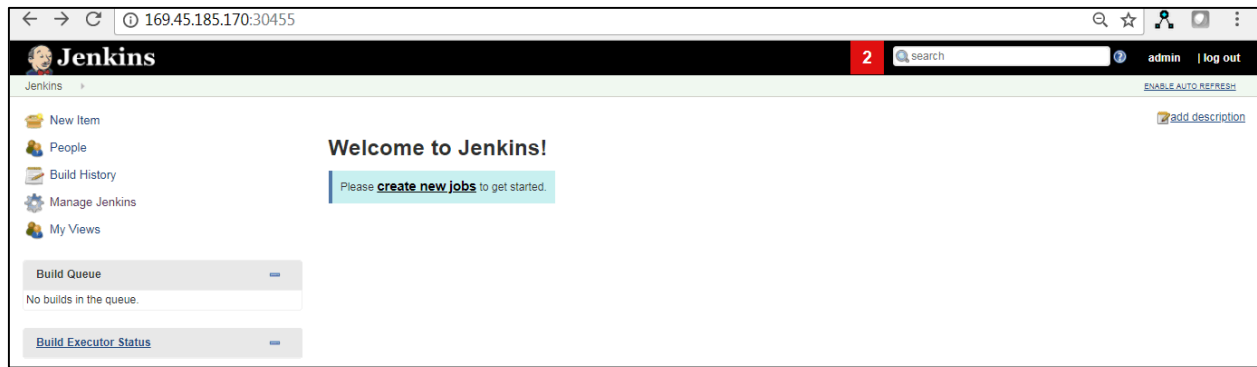
Jenkins is deployed, and the required plugins are installed on the available IBM Cloud Private cluster. You can now explore the system configuration and try to create a pipeline to build Docker images.

Note: DO NOT modify any settings on the **System Configuration** page.

Part 7: Accessing the Jenkins dashboard

__1. Using the provided URL, access the Jenkins dashboard.



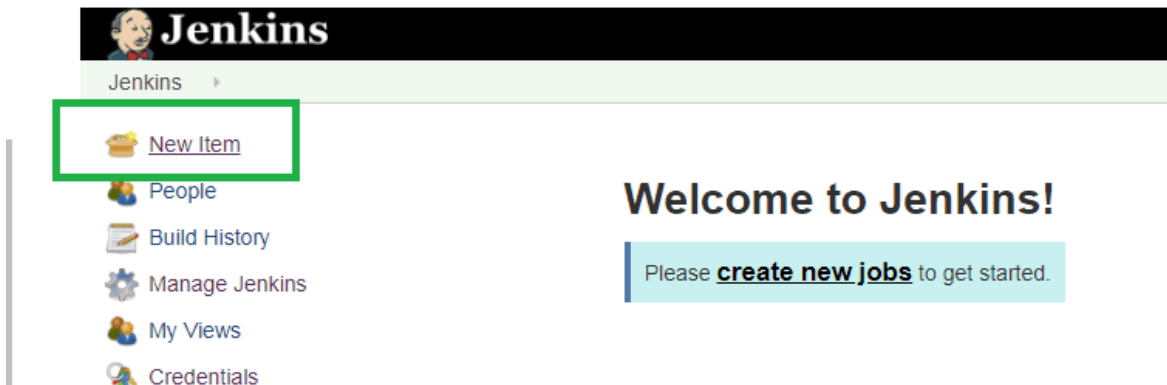


__2. View the system configuration. Select **Manage Jenkins** > **Configure System**.

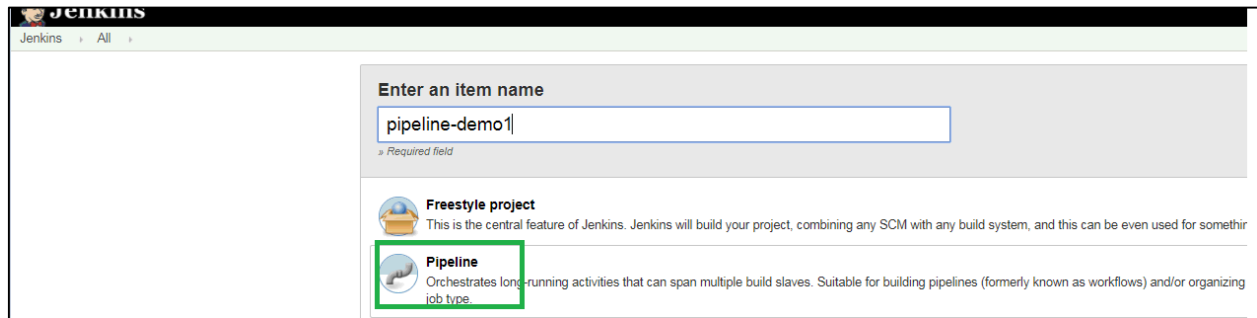


Part 8: Setting up task automation with Jenkins

__1. Click **New Item**.



__2. Enter an item name and select **Pipeline**.



__3. In the pipeline definition field input the following script.

```
pipeline {
  agent none
  stages {
    stage('Checkout Code') {
      parallel {

        stage('Build ppc64le') {
          agent {
            label "jenkins-demo3"
          }
          steps {
            script{
              checkout([$class: 'GitSCM', branches: [[name:
                '*/master']], doGenerateSubmoduleConfigurations: false, extensions: [[$class:
                'RelativeTargetDirectory', relativeTargetDir: 'src'], [$class:
                'CleanBeforeCheckout']], submoduleCfg: [], userRemoteConfigs: [[url:
                'https://github.com/sudeeshjohn/sampleflaskapp.git']]])
              appName = "default/flask-app-user1"
              registryHost = "mycluster.icp:8500/"
              imageName =
                "${registryHost}${appName}:${env.BUILD_ID}-ppc64le"
              env.BUILDIMG=imageName
              docker.withRegistry('https://mycluster.icp:8500/'
, 'docker'){
                def pcImg =
docker.build("mycluster.icp:8500/default/flask-app-user1:${env.BUILD_ID}", "-
f src/Dockerfile.ppc64le src/")
                sh "cp /root/.dockercfg
/home/jenkins/.dockercfg"
                pcImg.push()
                pcImg.withRun{springboot ->
git
'https://github.com/sudeeshjohn/sampleflaskapp.git'
                sh "ls"
                }
              }
            }
          }
        }
      }
    }
  }
}
```

```

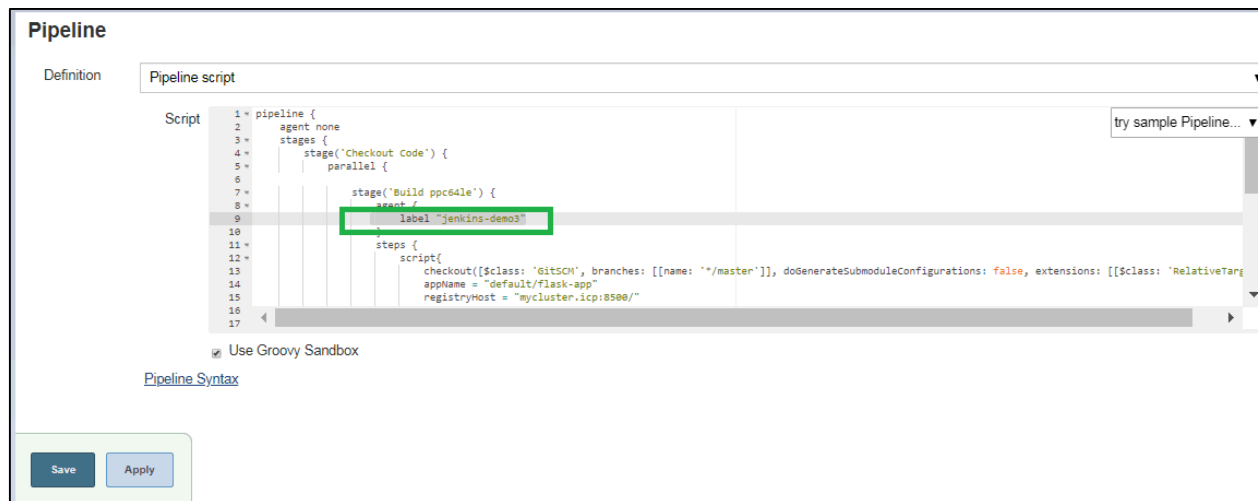
    }
  }
}

}

}

```

__5. Change the label field to your deployment name. For example, **jenkins-demo3**.



__6. To avoid conflicts in the image name that is pushed to the registry, you need to change the image name (appName parameter) in the script to match your user name.

For example,

appName="default/flask-app-user1" (for user1)

appName="default/flask-app-user2" (for user2)

__7. Click **Save**.

__8. Build the pipeline.

Jenkins

[Jenkins](#) > [pipeline-demo1](#) >

[Back to Dashboard](#)
[Status](#)
[Changes](#)
[Build Now](#)
[Delete Pipeline](#)
[Configure](#)
[Full Stage View](#)
[Pipeline Syntax](#)

Pipeline pipeline-demo1

[Recent Changes](#)

Stage View

Build History
[trend](#)

#1

Mar 17, 2018 3:43 PM

[RSS for all](#)
[RSS for failures](#)

#1

Mar 17

21:13

No Changes

Average stage times:

Checkout Code	Build ppc64le
316ms	4min 44s
316ms	(paused for 1hr 16min)

Permalinks

- Last build (#1), 1 hr 21 min ago

Jenkins

[Jenkins](#) > [pipeline-demo1](#) > [#1](#)

[Back to Project](#)
[Status](#)
[Changes](#)
Console Output
[View as plain text](#)
[Edit Build Information](#)
[Git Build Data](#)
[No Tags](#)
[Docker Fingerprints](#)

Console Output

Started by user [admin](#)

Running in Durability level: MAX_SURVIVABILITY

```
[Pipeline] stage
[Pipeline] { (Checkout Code)
[Pipeline] parallel
[Pipeline] [Build ppc64le] { (Branch: Build ppc64le)
[Pipeline] [Build ppc64le] stage
[Pipeline] [Build ppc64le] { (Build ppc64le)
[Pipeline] [Build ppc64le] node
[Build ppc64le] Still waiting to schedule task
[Build ppc64le] jnlp-9pbhn is offline
[Build ppc64le] Running on jnlp-9pbhn in /home/jenkins/works
```

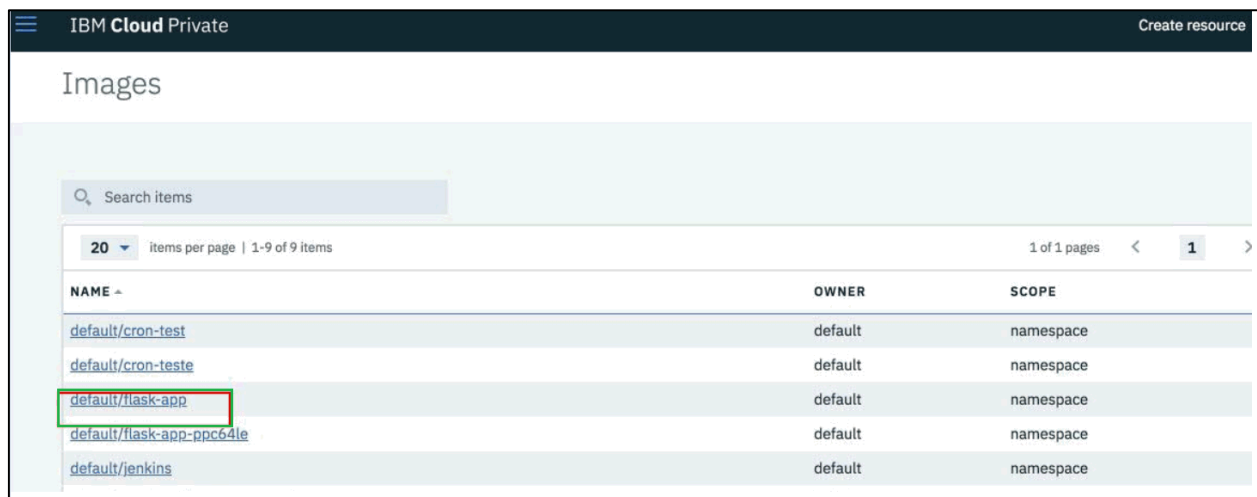


```
Jenkins  ▶ pipeline-demo1  ▶ #2

[Build ppc64le] + is
[Build ppc64le] Dockerfile
[Build ppc64le] Dockerfile.ppc64le
[Build ppc64le] Jenkinsfile
[Build ppc64le] README.md
[Build ppc64le] app
[Build ppc64le] conf
[Pipeline] [Build ppc64le] // script
[Pipeline] [Build ppc64le] }
[Pipeline] [Build ppc64le] // node
[Pipeline] [Build ppc64le] }
[Pipeline] [Build ppc64le] // stage
[Pipeline] [Build ppc64le] }
[Pipeline] // parallel
[Pipeline] }
[Pipeline] // stage
[Pipeline] End of Pipeline
Finished: SUCCESS
```

__ 9. When the job successfully completes, locate the newly built image in IBM Cloud Private dashboard.

From the IBM Cloud Private navigation menu, select **Catalog -> Images**.



NAME ^	OWNER	SCOPE
default/cron-test	default	namespace
default/cron-teste	default	namespace
default/flask-app	default	namespace
default/flask-app-ppc64le	default	namespace
default/jenkins	default	namespace

Deploying a microservices based application on a Kubernetes cluster

Microservices allow a large application to be decomposed into independent constituent parts, with each part having its own realm of responsibility. To serve a single user or API request, a microservices-based application can call many internal microservices to compose its response.

This sample microservices application simulates a fictional airline company called – Acme Air and consists of the following services:

- acmeair-mainapp: The Front End and the main service
- acmeair-as: The Authentication service
- acmeair-cs: The Customer service
- acmeair-fs: The Flight service
- acmeair-bs: The Booking service
- acmeair-ss: The Support service

Each of the service also depends on a database. MongoDB is used as the database. In this setup, all the services run on Power nodes.

The code used in this article is available here – <https://github.com/bpradipt/acmeair>

Optional: You can build the Acme air helm chart, or you can deploy the chart that is already available in your cluster. To build the Acme air helm chart, see [Part 11: \(Optional\) Building the Acme air Helm chart](#).

Part 9: Deploying the Acme air app

- __ 1. From the navigation menu, select **Catalog > Helm charts**. In the search field, type “acmeair”.
- __ 2. Click **Configure**.
- __ 4. Provide a **Release name**. For example, “acme-demo”.
- __ 5. Select the namespace that matches your username. For example, if you are user1, use the namespace1 namespace.

Docs user1

IBM Cloud Private Create resource Support

[← View all](#)

Configure acmeair V 0.0.1

Configuration

A Sample and Benchmark (wasperf version) application Edit these parameters for configuration

Release name ⓘ <input type="text" value="acme-demo"/>	Target namespace ⓘ <input type="text" value="namespace1"/>
---	--

__6. Replace clusterIngress- host with the dummy DNS host name of the proxy node.

In this environment, the proxy node is the same as the master node. For example,

- user 1: “user1-cluster-1519122384897-master.openpowercontainer.com”
- user 2: “user2-cluster-1519122384897-master.openpowercontainer.com”
-
- user 30: “user30-cluster-1519122384897-master.openpowercontainer.com”

IBM Cloud Private Create resource Support

clusterIngress

host <input type="text" value="user1-cluster-1519122384897-master.openpowercontainer.com"/>

bookingDB

name <input type="text" value="booking-db1"/>	dbName <input type="text" value="acmeair_bookingdb"/>
image <input type="text" value="jjacobso/mongodb-ppc64le:latest"/>	port <input type="text" value="27017"/>

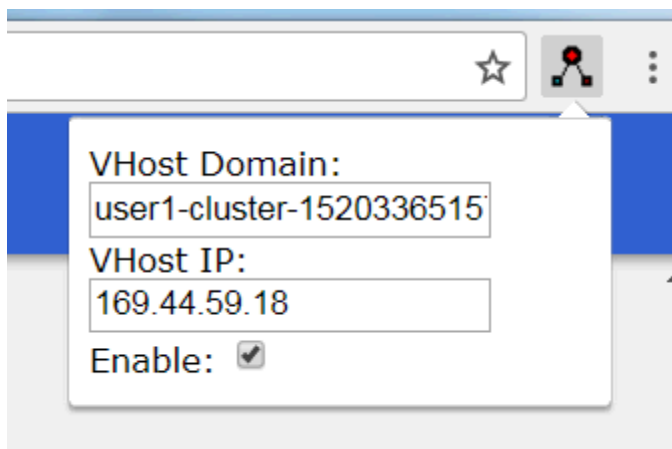
__ 8. Click **Install**.

__9. Click **View Helm release**. Locate your acme-air release. Review information about the services, deployments and pods that are created.

Part 10: Accessing the Acme air app

Note: Dummy hostnames are used in the ingress config file, each user must add a chrome extension for a virtual host. This allow you to map the dummy hostname to the proxy/master node's IP address.

For more details, see [Virtual Host \(Chrome web store\)](#).



__1. When the application is in a deployed state, you can access the service by using the following link: http://<DNS_HOST_NAME_OF_PROXY_NODE>/acmeair

For example, <http://user1-cluster-1519122384897-master.openpowercontainer.com/acmeair/>

__2. Click on the **Configure the Acmeair environment** link at the bottom of the home page to configure the environment and to populate the database.



Once the database is populated, you can use the menu options for performing various operations such as login, search for available flights, book flights, cancel bookings and logging out of the application.

Part 11: (Optional) Building the Acme air Helm chart

These instructions are for building the Acme air Helm chart on IBM Cloud Private environments that are running on Power systems.

Build the Docker image

__ 1. As root user, log on to the master node and run the following commands:

```
$ sudo su -
```

Install pre-req packages

```
$ sudo apt-get install git gradle openjdk-8-jdk
```

Set JAVA_HOME

```
$ export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-ppc64el
```

```
$ export PATH=$JAVA_HOME/bin:$PATH
```

Build the packages

```
$ git clone https://github.com/bpradipt/acmeair.git
```

```
$ cd acmeair
```

```
$ gradle build
```

__2. Build the Docker images

```
$ for i in acmeair-fs acmeair-as acmeair-cs acmeair-bs acmeair-ss  
acmeair-mainapp; do cd $i && sudo docker build -t $i -f  
Dockerfile.ppc64le . && cd .. ; done
```

__3. View the Docker images

```
$ docker images | grep acmeair
```

Load Docker images on the worker nodes

__4. On the master node, save the images on to a tar file.

```
sudo docker save -o <path to image tar file> <image name:tag> <image  
name:tag>..
```

For example,

```
sudo docker save -o ./acmeair-images.tar acmeair-mainapp:latest  
acmeair-ss:latest acmeair-bs:latest acmeair-cs:latest acmeair-  
as:latest acmeair-fs:latest
```

```
gzip acmeair-images.tar
```

__5. On each worker node, copy the tar.gz file.

```
scp <tar.gz file name> <target path on worker node>
```

For example,

```
scp acmeair-images.tar.gz cluster-1519122384897-worker1:/root/
```

```
scp acmeair-images.tar.gz cluster-1519122384897-worker2:/root/
```

__6. On each worker node, load the images.

```
ssh <worker node name>
```

```
sudo docker load -i <path to image tar.gz file>
```

```
sudo docker images | grep acme
```

For example,

```
root@cluster-1519122384897-worker1:~# sudo docker load -i ./acmeair-  
images.tar.gz
```

```
root@ cluster-1519122384897-master.openpowercontainer.com:~# docker  
images | grep acme
```

acmeair-mainapp			latest
642a6e34058e	25 minutes ago	534MB	

acmeair-ss			latest
0c5505e5a1e1	25 minutes ago	533MB	

acmeair-bs			latest
d70580d2a9db	25 minutes ago	535MB	

acmeair-cs			latest
8469561c0267	25 minutes ago	535MB	

acmeair-as			latest
9e6dc0c2973c	26 minutes ago	535MB	

acmeair-fs			latest
eb23816d51d5	26 minutes ago	535MB	

```
root@ cluster-1519122384897-master.openpowercontainer.com:~#
```

Package the helm chart

For more details, see

https://www.ibm.com/support/knowledgecenter/en/SSBS6K_2.1.0.1/app_center/add_package.html#package_chart

__7. Setup Helm CLI

For information on how to setup the helm CLI, see

https://www.ibm.com/support/knowledgecenter/en/SSBS6K_2.1.0.1/app_center/create_helm_cli.html.

__i. On the master node, go to the location where the acmeair project is cloned. Navigate to the `helm_chart` directory.

```
cd helm_chart
```

__ii. Package your chart and confirm that the package was added.

```
helm package acmeair; ls -l
```

If the package was added, the output of the command lists the `acmeair-0.0.1.tgz` file.

__8. Add the chart to the internal repository that is provided with IBM Cloud Private.

__i. Install the IBM Cloud Private CLI, see [Installing the IBM® Cloud Private CLI](#).

__ii. Log in to your cluster from the IBM Cloud Private CLI and access the Docker private image registry.

```
bx pr login -a https://<cluster_CA_domain>:8443 --skip-ssl-validation
```

For example,

```
$ bx pr login -a https://169.44.59.18:8443 --skip-ssl-validation
```

```
Login method invokedAPI endpoint: https://169.44.59.18:8443
```

```
Username> admin
```

```
Password>
```

```
Authenticating...
```

```
OK
```

```
Select an account:
```

```
1. ICP Account (id-icp-account)
```


Enter a number> 1

Targeted account: ICP Account (id-icp-account)

\$

__iii. Install the Helm chart.

```
bx pr load-helm-chart --archive <helm_chart_archive> [--  
clustername <cluster_CA_domain>]
```

For example,

```
$ scp -i <keyfile> ubuntu@169.44.59.18:/home/ubuntu/acmeair-  
0.0.1.tgz .
```

```
acmeair-0.0.1.tgz          100%      3912      3.8KB/s   00:01
```

```
IBM@mjulie MINGW64 ~
```

```
$ bx pr load-helm-chart --archive ./acmeair-0.0.1.tgz --  
clustername 169.44.59.18
```

Loading helm chart

```
{ "url": "https://169.44.59.18:8443/helm-repo/charts/index.yaml" }
```

OK

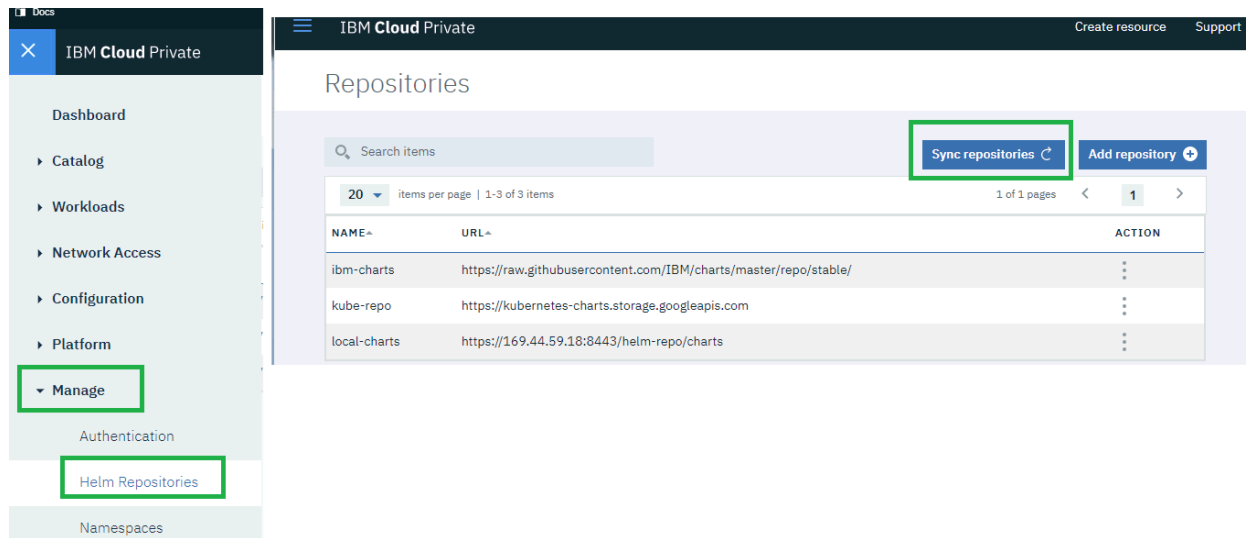
__iv. Update the package repository by using the IBM Cloud Private cluster management console.

__a. From the navigation menu, select **Manage > Helm Repositories**.

__b. Click **Sync Repositories**.

__c. From the navigation menu, select **Catalog > Helm Charts**.

The new Helm chart is loaded into the IBM Cloud Private catalog.



Next: Deploy the Acme air app, see [Part 9: Deploying the Acme air app.](#)

Appendix – useful links

- Developer Works community – <http://ibm.biz/cloud-private>
- IBM Cloud Private on IBM Knowledge Center – https://www.ibm.com/support/knowledgecenter/SSBS6K/product_welcome_cloud_private.html

We Value Your Feedback!

- Don't forget to submit your Think 2018 session and speaker feedback! Your feedback is very important to us – we use it to continually improve the conference.
- Access the Think 2018 agenda tool to quickly submit your surveys from your smartphone, laptop or conference kiosk.

