Deploying a demo app to Kubernetes cluster on IBM Cloud

Aco Vidovič
Ecosystem Advocacy Group Leader
Ibm central & Eastern Europe

Manual deployment to IBM Kubernetes Service (IKS) cluster

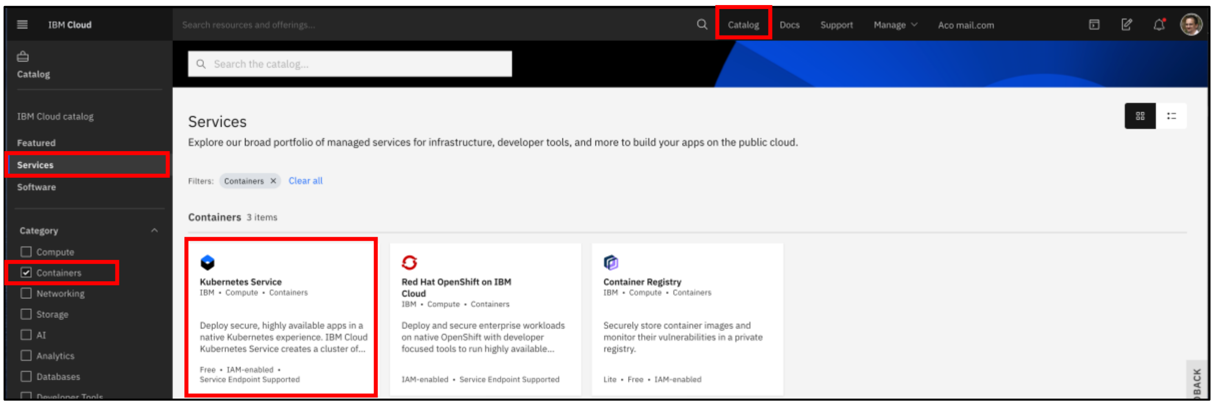
In this section, we will deploy our app manually to Kubernetes cluster on IBM Cloud.

1. Create your first Kubernetes cluster

Before you can deploy an app by using Kubernetes, start by **creating a cluster**. A cluster is a set of worker nodes that are organized into a network. The purpose of the cluster is to define a set of resources, nodes, networks, and storage devices that keep applications highly available.

To create a lite cluster do following steps:

a. Login to IBM Cloud.
b. In menu on top, select **Catalog**, then on the left select **Services**, then check **Containers**, then click on **Kubernetes Service** tile.



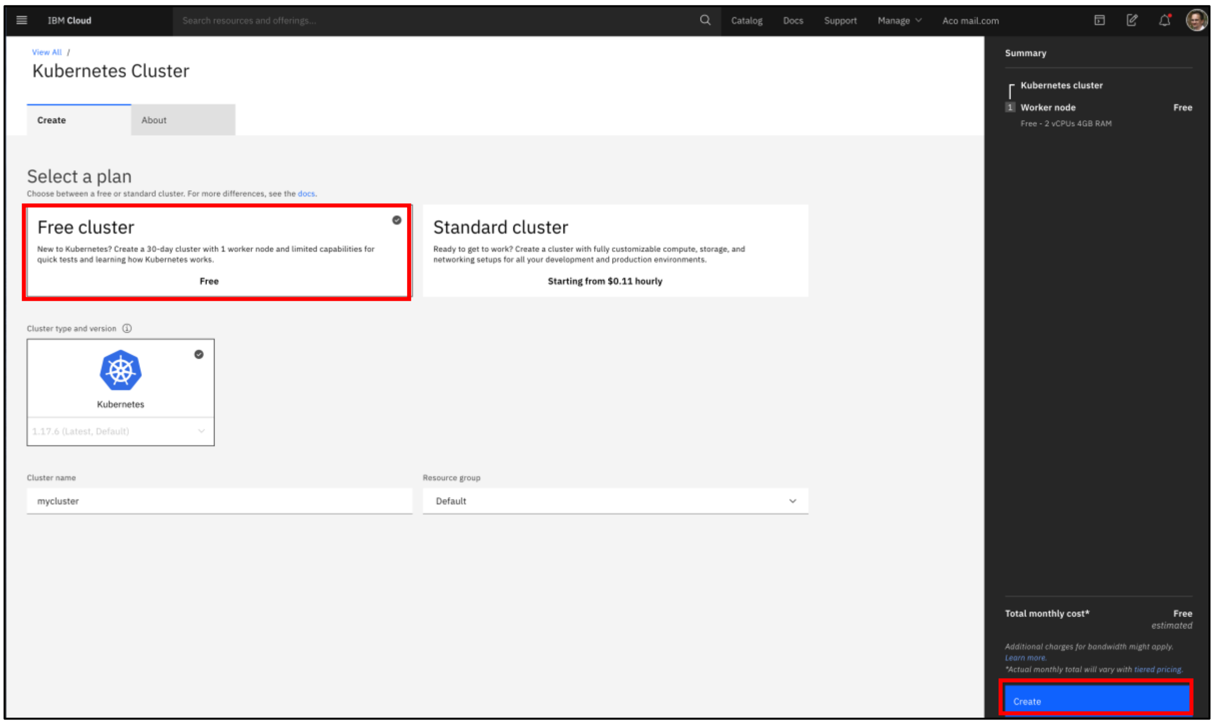
Services
Explore our broad portfolio of managed services for infrastructure, developer tools, and more to build your apps on the public cloud.

Kubernetes Service
IBM • Compute • Containers
Deploy secure, highly available apps in a native Kubernetes experience. IBM Cloud Kubernetes Service creates a cluster of...
Free • IAM-enabled • Service Endpoint Supported

Red Hat OpenShift on IBM Cloud
IBM • Compute • Containers
Deploy and secure enterprise workloads on native OpenShift with developer focused tools to run highly available...
IAM-enabled • Service Endpoint Supported

Container Registry
IBM • Compute • Containers
Securely store container images and monitor their vulnerabilities in a private registry.
Lite • Free • IAM-enabled

c. Select **Free cluster**. Keep the default values for Cluster name etc. Then click **Create** on the right hand side.


IBM Cloud | Search resources and offerings... | Catalog | Docs | Support | Manage | Aco mail.com

View All /

Kubernetes Cluster

Create | About

Select a plan

Choose between a free or standard cluster. For more differences, see the docs.

Free cluster

New to Kubernetes? Create a 30-day cluster with 1 worker node and limited capabilities for quick tests and learning how Kubernetes works.

Free

Standard cluster

Ready to get to work? Create a cluster with fully customizable compute, storage, and networking setups for all your development and production environments.

Starting from $0.11 hourly

Cluster type and version

Kubernetes

1.17.6 (Latest, Default)

Cluster name: mycluster

Resource group: Default

Summary

Kubernetes cluster

1 Worker node — Free

Free - 2 vCPUs 4GB RAM

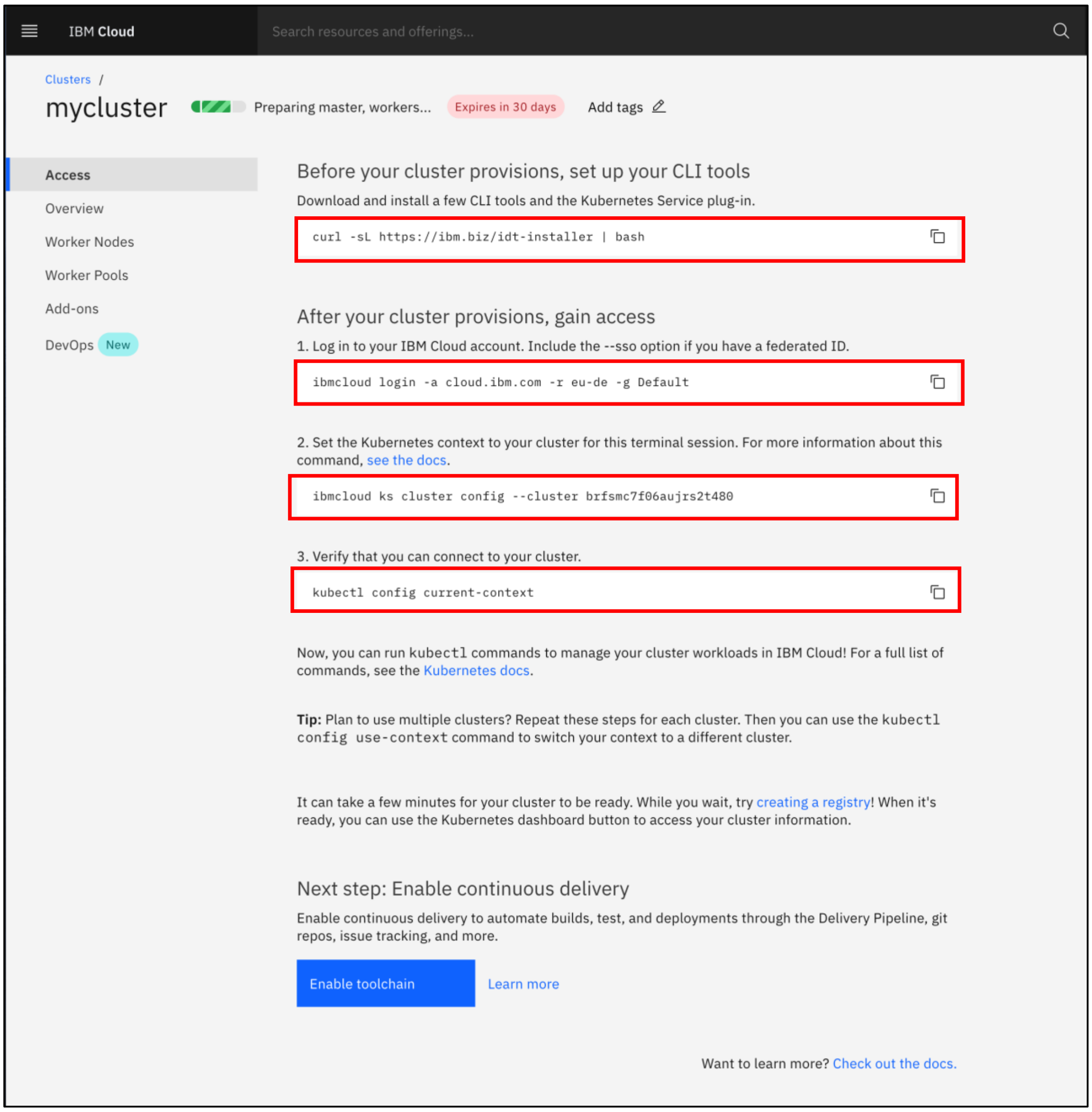
Total monthly cost* — Free estimated

Additional charges for bandwidth might apply. Learn more.

*Actual monthly total will vary with tiered pricing.

Create

d. On next screen, you will see a list of things to do **before and after** your Kubernetes cluster is provisioned. Follow instructions on this screen. It may take 10 minutes or so until the cluster is provisioned, so you will come back to this Clusters screen once your cluster is ready.


IBM Cloud

Search resources and offerings...

Clusters /

mycluster Preparing master, workers... Expires in 30 days Add tags

Before your cluster provisions, set up your CLI tools

Download and install a few CLI tools and the Kubernetes Service plug-in.

```
curl -sL https://ibm.biz/idt-installer | bash
```

After your cluster provisions, gain access

1. Log in to your IBM Cloud account. Include the --sso option if you have a federated ID.

```
ibmcloud login -a cloud.ibm.com -r eu-de -g Default
```

2. Set the Kubernetes context to your cluster for this terminal session. For more information about this command, see the docs.

```
ibmcloud ks cluster config --cluster brfsmc7f06aujrs2t480
```

3. Verify that you can connect to your cluster.

```
kubectl config current-context
```

Now, you can run `kubectl` commands to manage your cluster workloads in IBM Cloud! For a full list of commands, see the Kubernetes docs.

Tip: Plan to use multiple clusters? Repeat these steps for each cluster. Then you can use the `kubectl config use-context` command to switch your context to a different cluster.

It can take a few minutes for your cluster to be ready. While you wait, try creating a registry! When it's ready, you can use the Kubernetes dashboard button to access your cluster information.

Next step: Enable continuous delivery

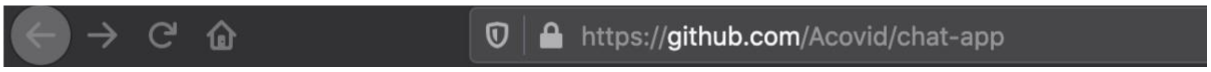
Enable continuous delivery to automate builds, test, and deployments through the Delivery Pipeline, git repos, issue tracking, and more.

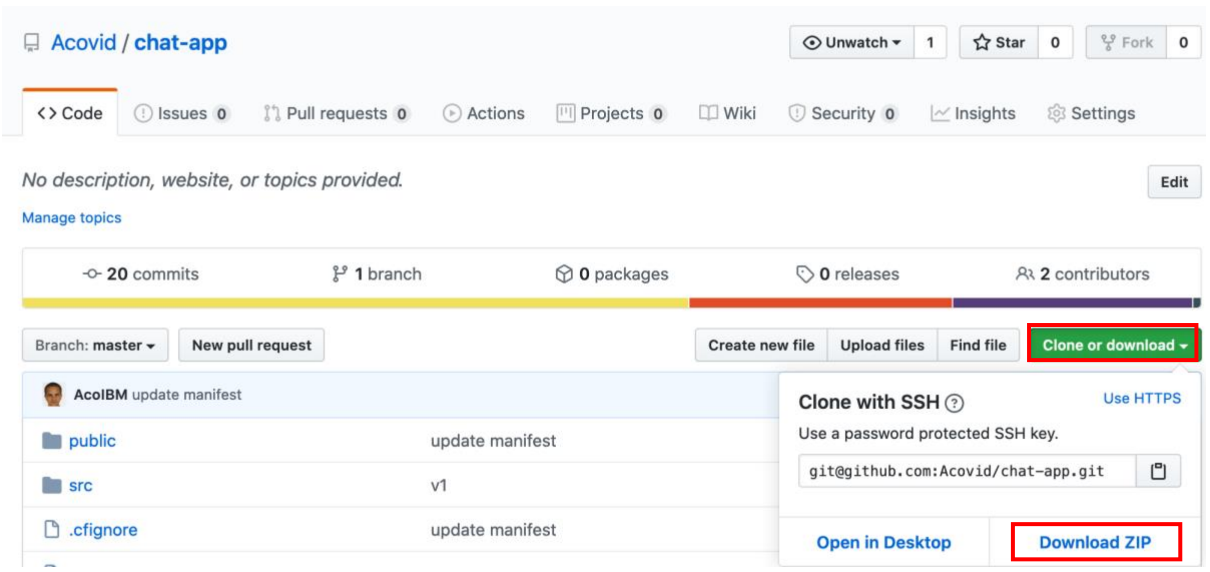
Want to learn more? Check out the docs.

2. While the cluster is being provisioned, download the demo app from the Github. This demo app can be used for chatting with other people. In your web browser, type in:

   https://github.com/Acovid/chat-app

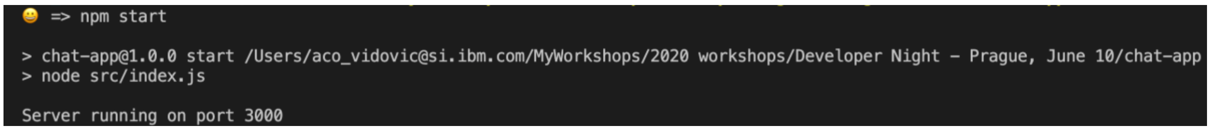
   Click on **Clone or download**, then **Download ZIP**.


https://github.com/Acovid/chat-app



3. After you downloaded the app, you can test it locally following instructions in file *readme.md*, section **Deploying the app locally**.

   Do not forget to **change directory** to where the source code is. Once you started the app with command `npm start`, you should see something like this:



```
😀 => npm start

> chat-app@1.0.0 start /Users/aco_vidovic@si.ibm.com/MyWorkshops/2020 workshops/Developer Night - Prague, June 10/chat-app
> node src/index.js

Server running on port 3000
```

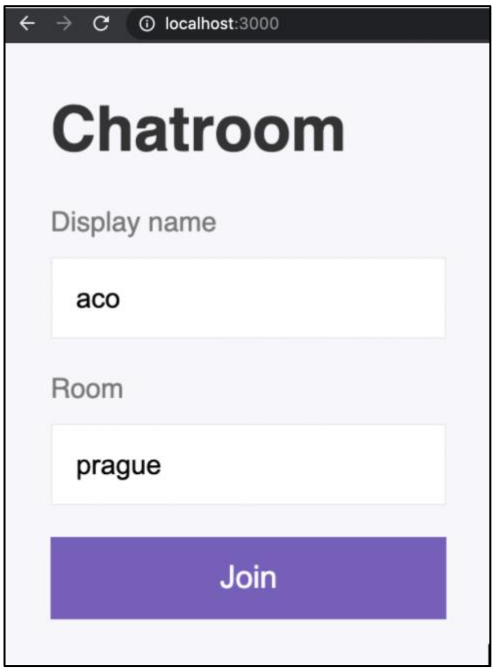
   Message **Server running on port 3000** tells you your demo app has started.

4. Open your app in the web browser, by typing in:

   `localhost:3000`

   Login with your user name and the name of the chatroom.

   Note: Any user name will work, it does not have to be defined upfront.
   Any chatroom name will work, too, but if you want to chat with other people, you should all **login to the same chatroom.**



localhost:3000

Chatroom

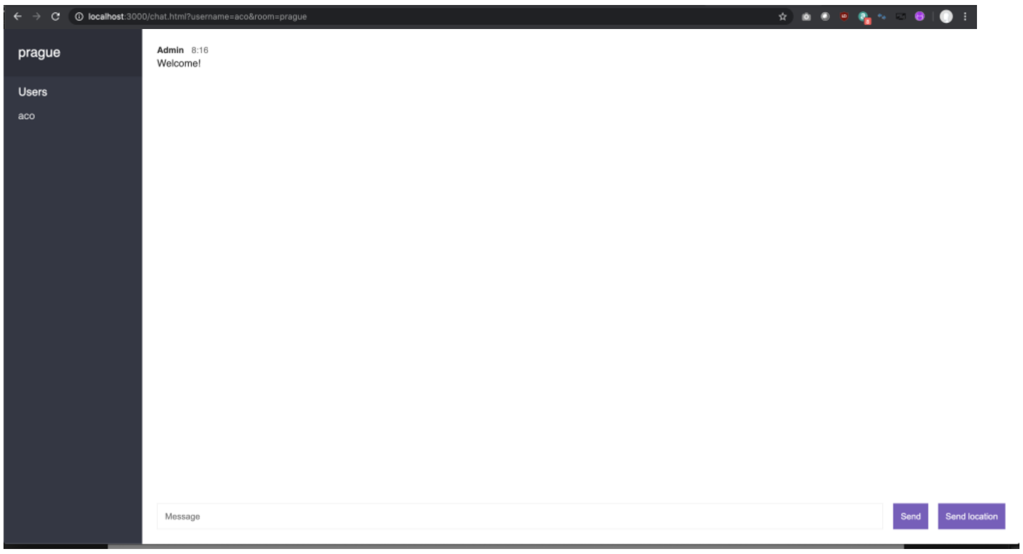
Display name

aco

Room

prague

When you see in your browser a screen like this, you are ready to chat with other pople:



localhost:3000/chat.html?username=aco&room=prague

prague

Users

aco

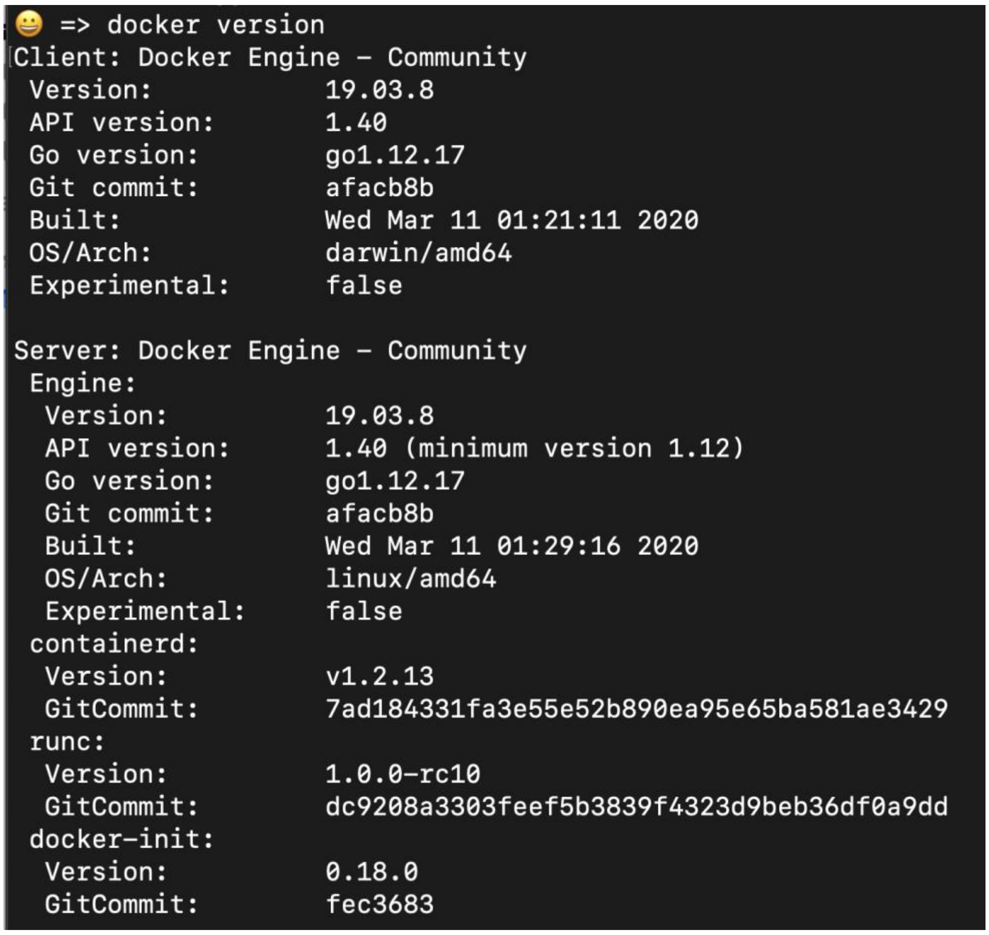
Admin 8:16
Welcome!

5. Check your local environment.

   a. Check if the Docker workstation runs on your workstation

```
docker version
```

You should see both, the client and the server.

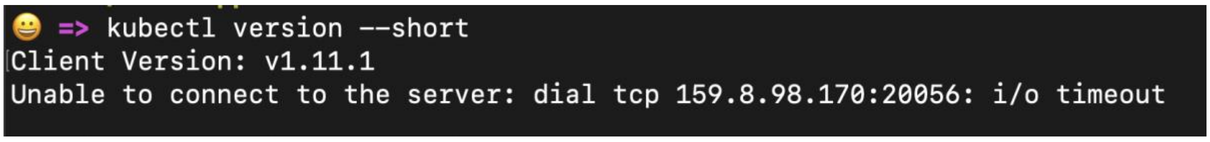


```
😀 => docker version
Client: Docker Engine - Community
 Version:           19.03.8
 API version:       1.40
 Go version:        go1.12.17
 Git commit:        afacb8b
 Built:             Wed Mar 11 01:21:11 2020
 OS/Arch:           darwin/amd64
 Experimental:      false

Server: Docker Engine - Community
 Engine:
  Version:          19.03.8
  API version:      1.40 (minimum version 1.12)
  Go version:       go1.12.17
  Git commit:       afacb8b
  Built:            Wed Mar 11 01:29:16 2020
  OS/Arch:          linux/amd64
  Experimental:     false
 containerd:
  Version:          v1.2.13
  GitCommit:        7ad184331fa3e55e52b890ea95e65ba581ae3429
 runc:
  Version:          1.0.0-rc10
  GitCommit:        dc9208a3303feef5b3839f4323d9beb36df0a9dd
 docker-init:
  Version:          0.18.0
  GitCommit:        fec3683
```

b. Check if Kubernetes CLI runs on your workstation.

```
kubectl version –short
```



```
😀 => kubectl version --short
Client Version: v1.11.1
Unable to connect to the server: dial tcp 159.8.98.170:20056: i/o timeout
```

The message **Unable to connect** is perfectly normal at this point, as you are not connected to any K8S cluster yet.

6. Create a Docker image from your source code:

```
docker build -t de.icr.io/acovid/chat-app .
```

7. Check if image was built:

```
docker images de.icr.io/acovid/chat-app:latest
```

8. Run container locally:

```
docker run -d -p 3001:3000 de.icr.io/acovid/chat-app:latest
```

9. Open your browser to check if your containerized app is running, by typing in

   `http://localhost:3001`

   Your app should look the same as when you ran it from port 3000.

10. Back on your Clusters screen follow the instructions to:
    a. login to your IBM Cloud account
    b. set the Kubernetes context (i.e. to which cluster you are connecting)
    c. verify you are connected to cluster

11. Create your private image registry on IBM Cloud.

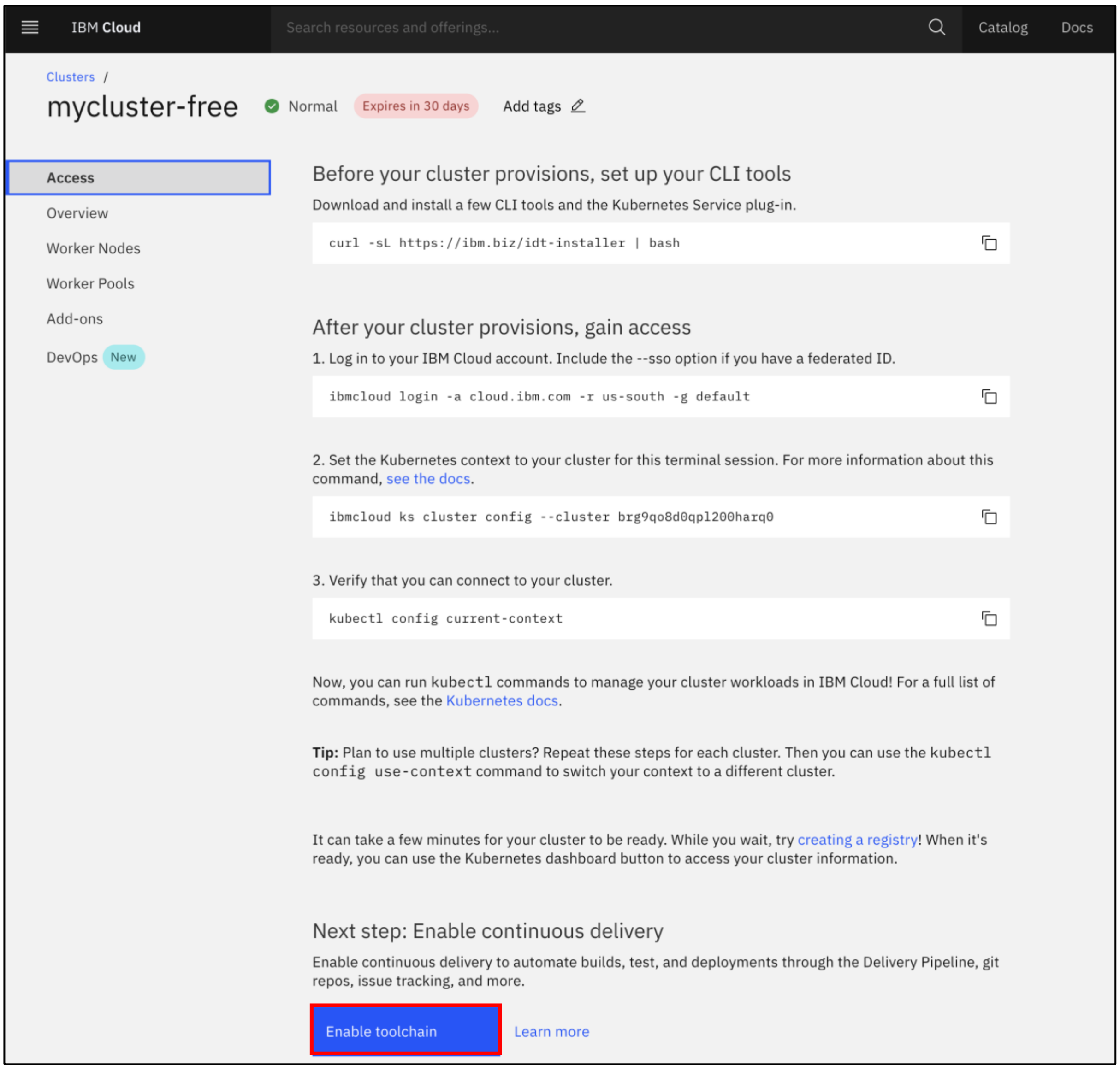
    ```
    ibmcloud cr namespace-add acovid
    ```

12. Login to your image registry

    ```
    ibmcloud cr login
    ```

    If your image registry is in Germany, its name will be `de.icr.io`
    If your image registry is in US, the name will be `us.icr.io`
    If in UK, the name will be `uk.icr.io`

13. In next few steps you will enable the CI/CD toolchain.
    a. In your web browser, return to the **Clusters** page and click on **Enable toolchain**.



mycluster-free Normal Expires in 30 days Add tags

Before your cluster provisions, set up your CLI tools

Download and install a few CLI tools and the Kubernetes Service plug-in.

```
curl -sL https://ibm.biz/idt-installer | bash
```

After your cluster provisions, gain access

1. Log in to your IBM Cloud account. Include the --sso option if you have a federated ID.

```
ibmcloud login -a cloud.ibm.com -r us-south -g default
```

2. Set the Kubernetes context to your cluster for this terminal session. For more information about this command, see the docs.

```
ibmcloud ks cluster config --cluster brg9qo8d0qpl200harq0
```

3. Verify that you can connect to your cluster.

```
kubectl config current-context
```

Now, you can run `kubectl` commands to manage your cluster workloads in IBM Cloud! For a full list of commands, see the Kubernetes docs.

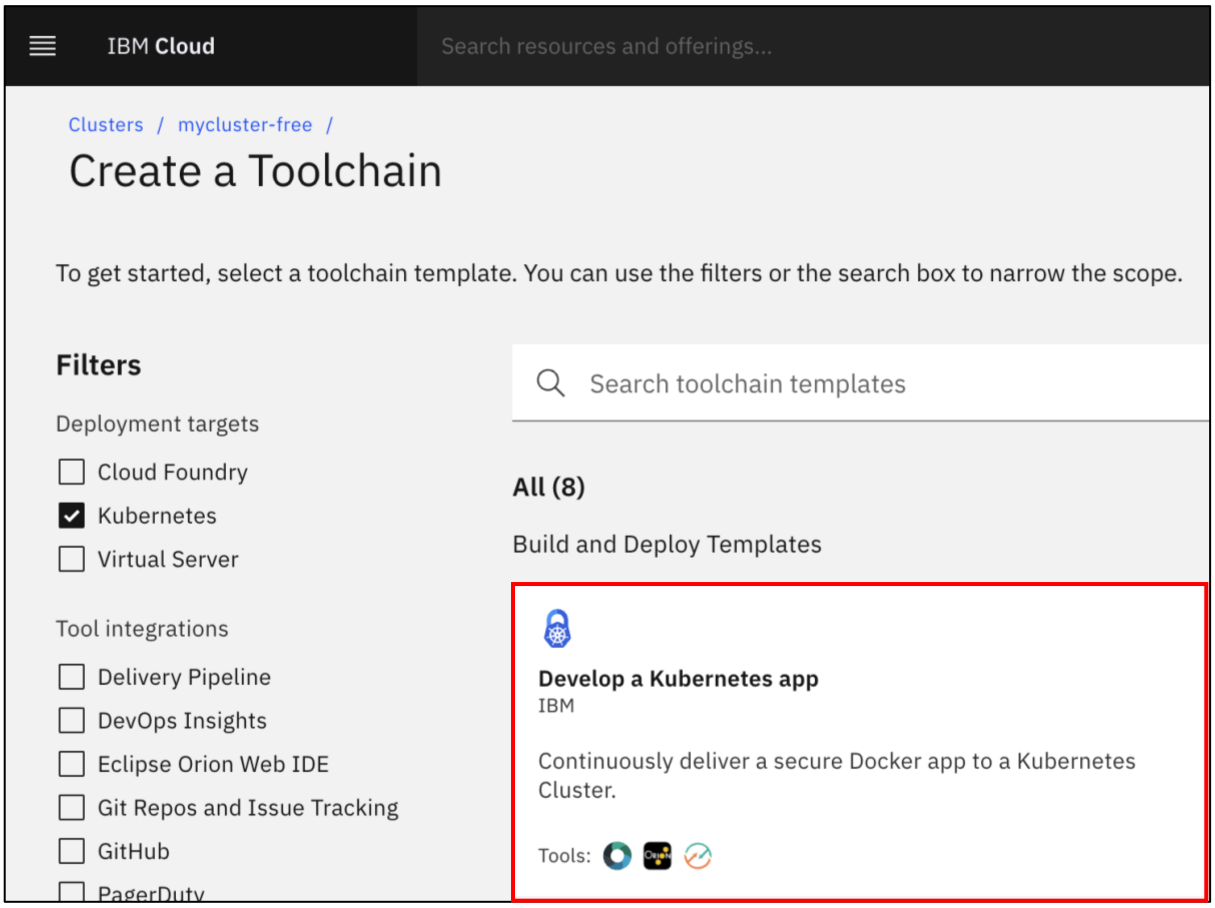
Tip: Plan to use multiple clusters? Repeat these steps for each cluster. Then you can use the `kubectl config use-context` command to switch your context to a different cluster.

It can take a few minutes for your cluster to be ready. While you wait, try creating a registry! When it's ready, you can use the Kubernetes dashboard button to access your cluster information.

Next step: Enable continuous delivery

Enable continuous delivery to automate builds, test, and deployments through the Delivery Pipeline, git repos, issue tracking, and more.

b. Click on **Develop a Kubernetes app** tile.



IBM Cloud

Search resources and offerings...

Clusters / mycluster-free /

Create a Toolchain

To get started, select a toolchain template. You can use the filters or the search box to narrow the scope.

Filters

Deployment targets

- [ ] Cloud Foundry
- [x] Kubernetes
- [ ] Virtual Server

Tool integrations

- [ ] Delivery Pipeline
- [ ] DevOps Insights
- [ ] Eclipse Orion Web IDE
- [ ] Git Repos and Issue Tracking
- [ ] GitHub
- [ ] PagerDuty

Search toolchain templates

All (8)

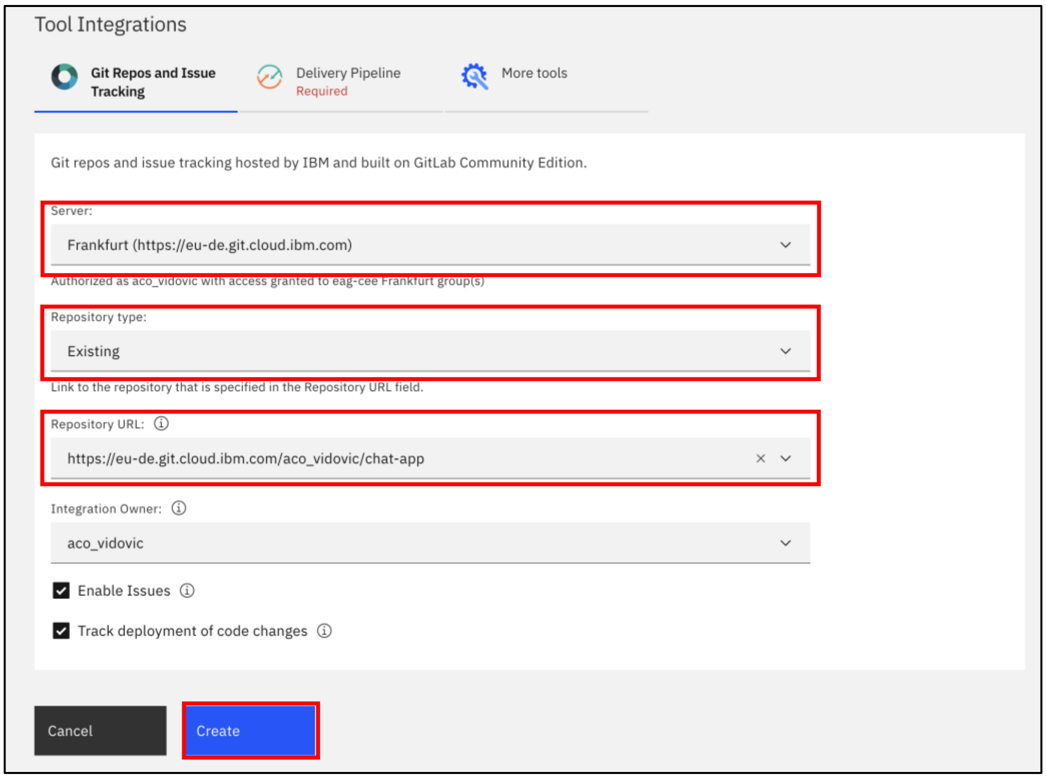
Build and Deploy Templates

Develop a Kubernetes app
IBM

Continuously deliver a secure Docker app to a Kubernetes Cluster.

Tools:

c. Select appropriate values for the **Server** where your git repository is located, **Repository type** (such as Existing or Clone etc), select and the **Repository URL**. Then click **Create**.



Tool Integrations

Git Repos and Issue Tracking | Delivery Pipeline Required | More tools

Git repos and issue tracking hosted by IBM and built on GitLab Community Edition.

Server: Frankfurt (https://eu-de.git.cloud.ibm.com)

Authorized as aco_vidovic with access granted to eag-cee Frankfurt group(s)

Repository type: Existing

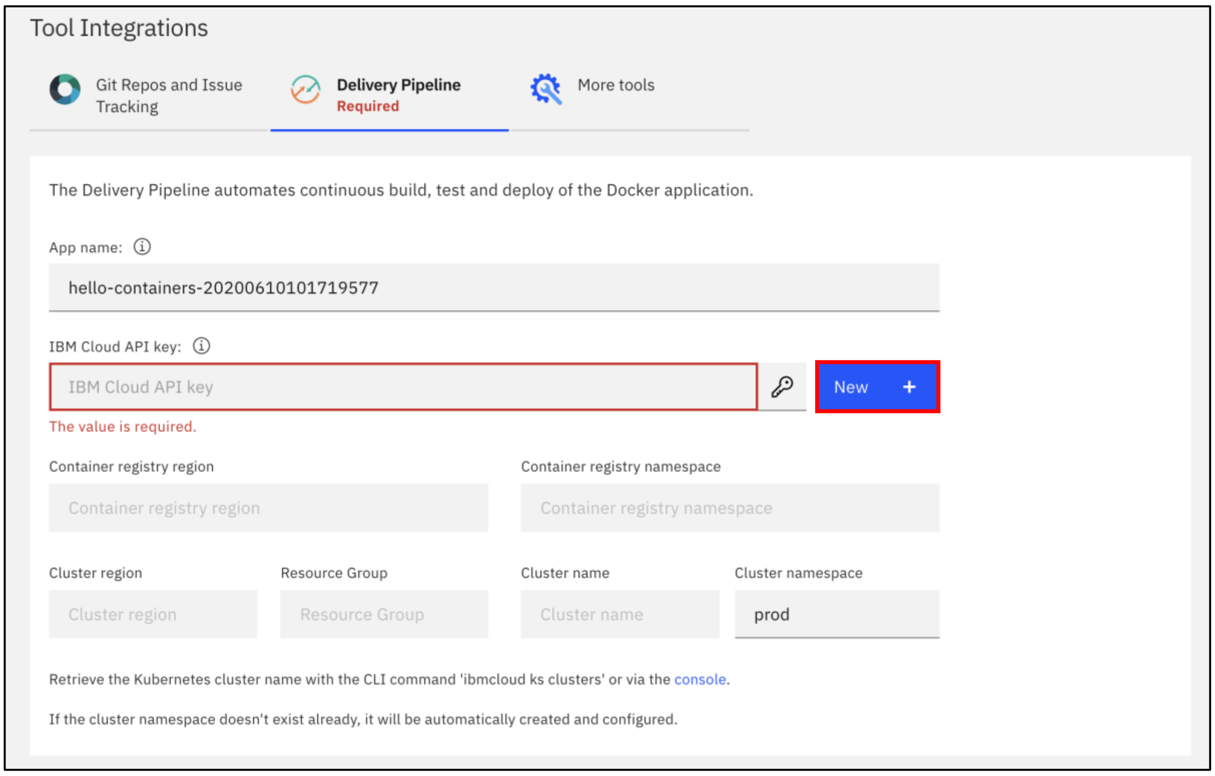
Link to the repository that is specified in the Repository URL field.

Repository URL: https://eu-de.git.cloud.ibm.com/aco_vidovic/chat-app

Integration Owner: aco_vidovic

- [x] Enable Issues
- [x] Track deployment of code changes

d. Next to the **IBM Cloud API key** field, click on **New**.


Tool Integrations

Git Repos and Issue Tracking | Delivery Pipeline Required | More tools

The Delivery Pipeline automates continuous build, test and deploy of the Docker application.

App name: hello-containers-20200610101719577

IBM Cloud API key: IBM Cloud API key

The value is required.

Container registry region: Container registry region

Container registry namespace: Container registry namespace

Cluster region: Cluster region

Resource Group: Resource Group

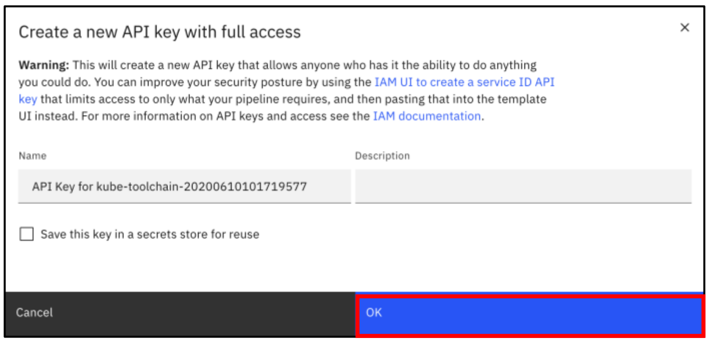
Cluster name: Cluster name

Cluster namespace: prod

Retrieve the Kubernetes cluster name with the CLI command 'ibmcloud ks clusters' or via the console.

If the cluster namespace doesn't exist already, it will be automatically created and configured.

e. Accept default value and click **OK**.


Create a new API key with full access

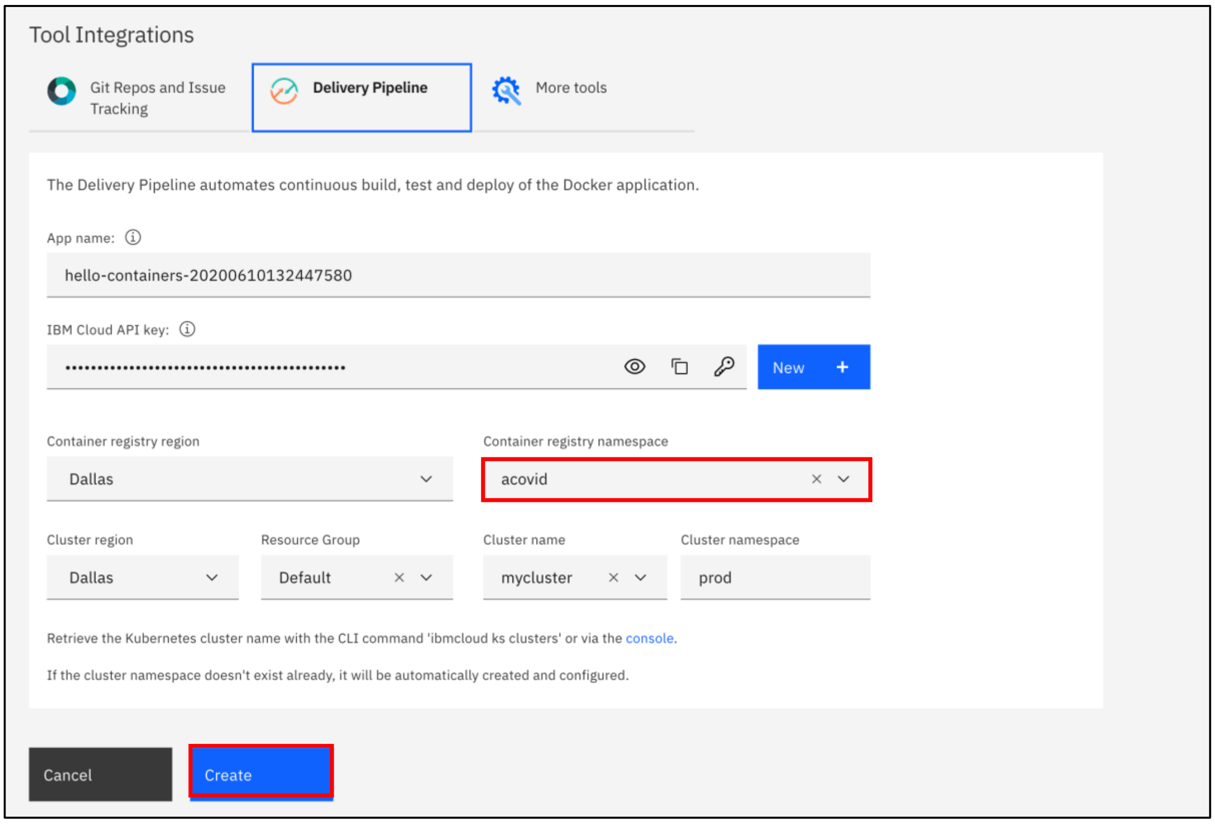
Warning: This will create a new API key that allows anyone who has it the ability to do anything you could do. You can improve your security posture by using the IAM UI to create a service ID API key that limits access to only what your pipeline requires, and then pasting that into the template UI instead. For more information on API keys and access see the IAM documentation.

Name: API Key for kube-toolchain-20200610101719577

Description:

☐ Save this key in a secrets store for reuse

f. Provide **namespace** for your container registry and click **Create**.


Tool Integrations

The Delivery Pipeline automates continuous build, test and deploy of the Docker application.

App name: hello-containers-20200610132447580

IBM Cloud API key: ••

Container registry region: Dallas

Container registry namespace: acovid

Cluster region: Dallas

Resource Group: Default

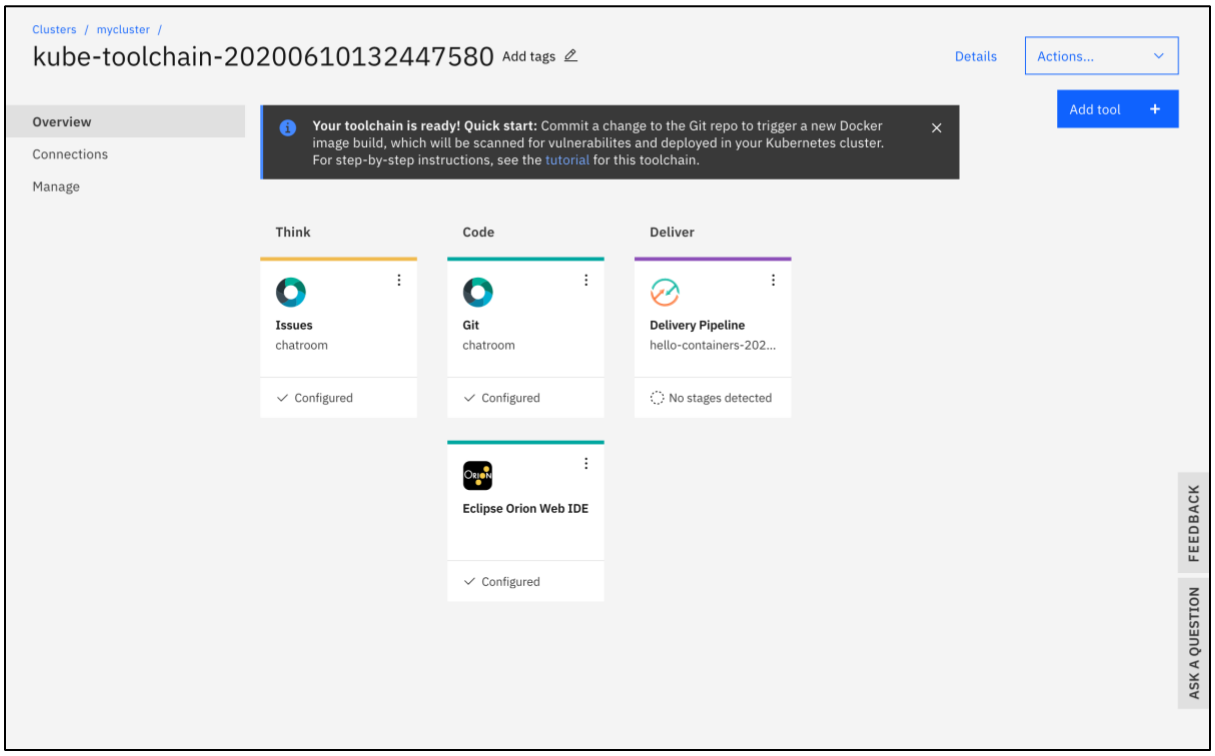
Cluster name: mycluster

Cluster namespace: prod

Retrieve the Kubernetes cluster name with the CLI command 'ibmcloud ks clusters' or via the console.

If the cluster namespace doesn't exist already, it will be automatically created and configured.

14. Leave your toolchain for now, let it be created, you will come back to it later when it is ready.


Clusters / mycluster /

kube-toolchain-20200610132447580

Your toolchain is ready! Quick start: Commit a change to the Git repo to trigger a new Docker image build, which will be scanned for vulnerabilities and deployed in your Kubernetes cluster. For step-by-step instructions, see the tutorial for this toolchain.

Think	Code	Deliver
Issues, chatroom — Configured	Git, chatroom — Configured	Delivery Pipeline, hello-containers-202... — No stages detected
	Eclipse Orion Web IDE — Configured	

15. Back at your laptop's command prompt, push your image to IBM registry:

```
docker push de.icr.io/acovid/chat-app
```

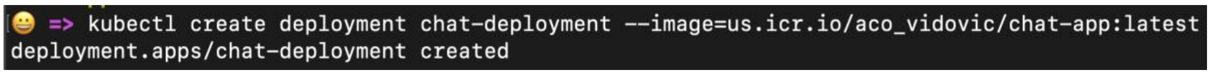
16. List images in your registry:

```
ibmcloud cr images
```

17. Deploy app to Kubernetes:

```
kubectl create deployment chat-deployment --\
image=de.icr.io/acovid/chat-app
```

Attention: do not type in the backslash (\) sign in the above command! It is there just to depict the end of the line.

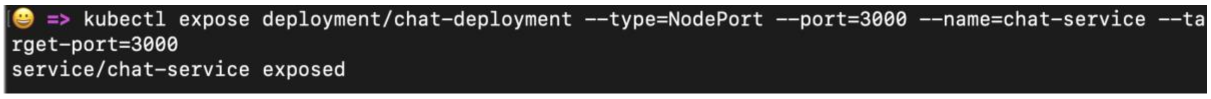


```
😀 => kubectl create deployment chat-deployment --image=us.icr.io/aco_vidovic/chat-app:latest
deployment.apps/chat-deployment created
```

18. Create service:

```
kubectl expose deployment/chat-deployment --type=NodePort --port=3000\
--name=chat-service --target-port=3000
```

Attention: do not type in the backslash (\) sign!

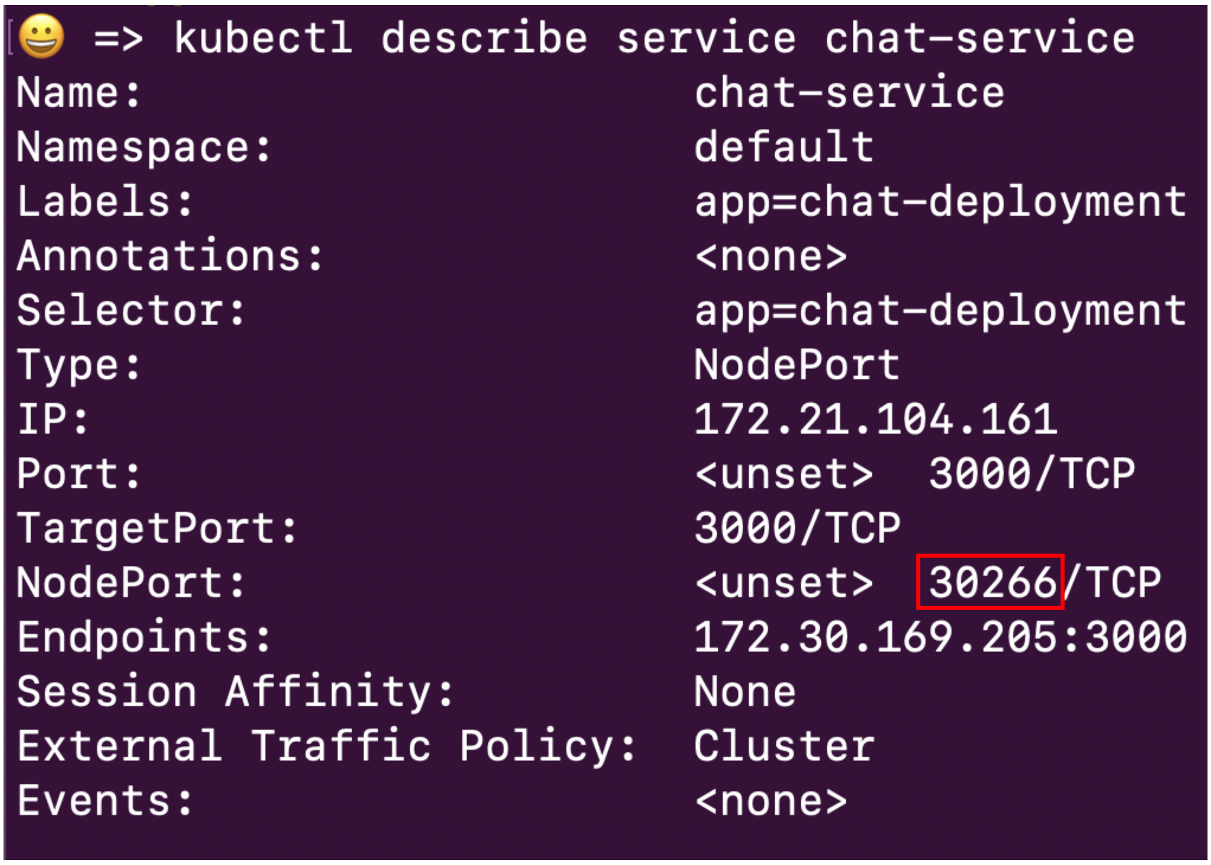


```
😀 => kubectl expose deployment/chat-deployment --type=NodePort --port=3000 --name=chat-service --ta
rget-port=3000
service/chat-service exposed
```

19. Find the external port of your app:

```
kubectl describe service <service-name>
```

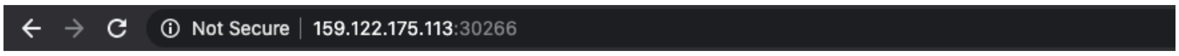
The external port should look like this:



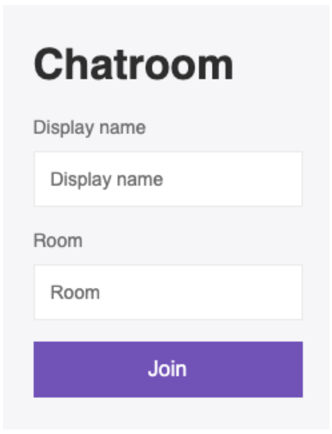
```
😀 => kubectl describe service chat-service
Name:                     chat-service
Namespace:                default
Labels:                   app=chat-deployment
Annotations:              <none>
Selector:                 app=chat-deployment
Type:                     NodePort
IP:                       172.21.104.161
Port:                     <unset>  3000/TCP
TargetPort:               3000/TCP
NodePort:                 <unset>  30266/TCP
Endpoints:                172.30.169.205:3000
Session Affinity:         None
External Traffic Policy:  Cluster
Events:                   <none>
```

In our case the port is **30266**.

20. Test the app from the browser: `<public-ip-address>:<external-port>`



Not Secure | 159.122.175.113:30266



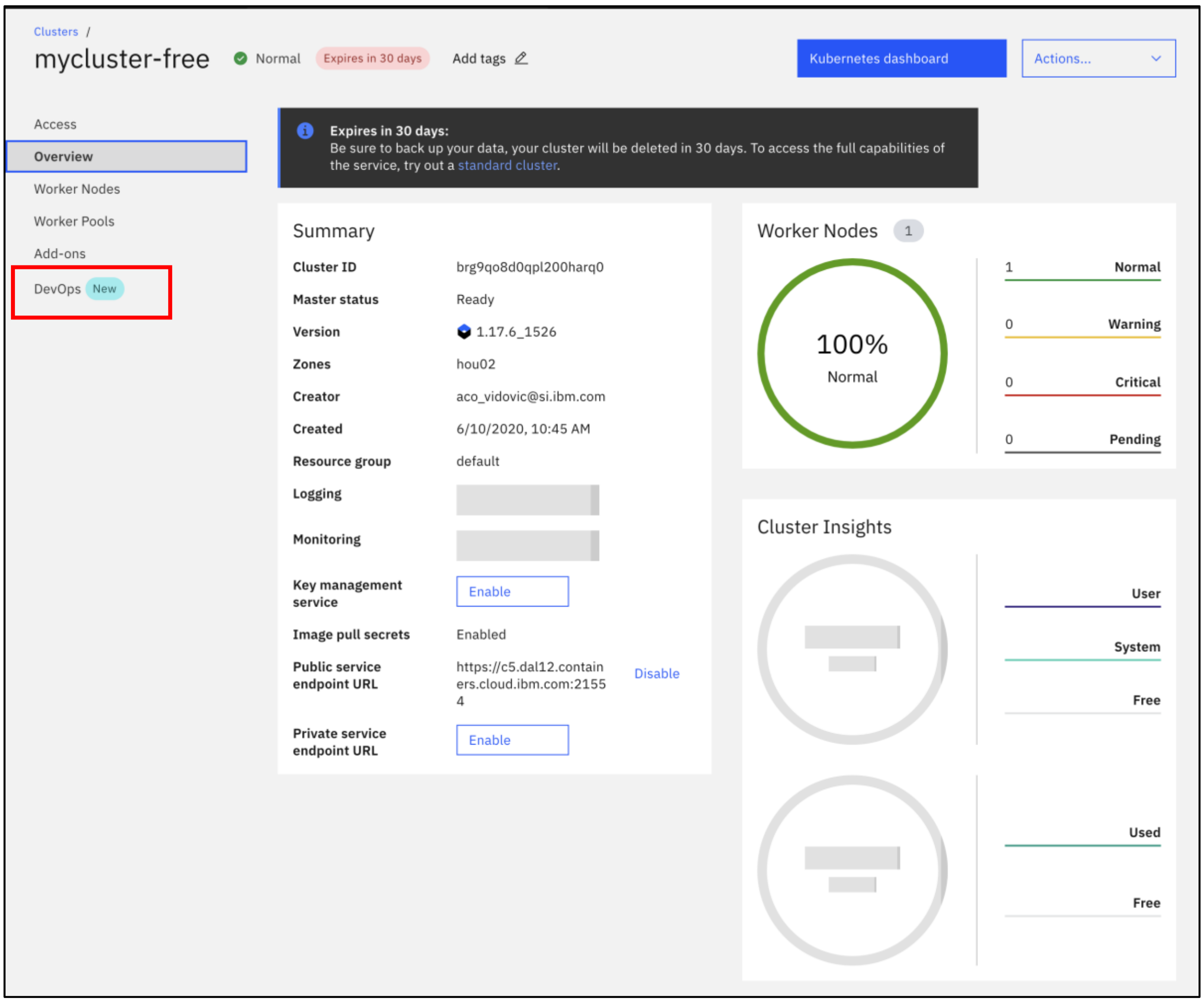
With this, we have manually deployed our chat app to Kubernetes cluster on IBM Cloud.

Automatic deployment using CI/CD toolchain and pipeline

Next, we will automate our deployment by using **Continuous Delivery (CD) service on IBM Cloud**. With CI/CD toolchains, you can build, test, and deliver applications by using DevOps practices and industry-leading tools.

You have already enabled your K8S cluster for CI/CD toolchain. In next steps, you will test how it works.

1. In your web browser, go back to **Clusters** page, open your cluster and click on **DevOps** on the left hand side.

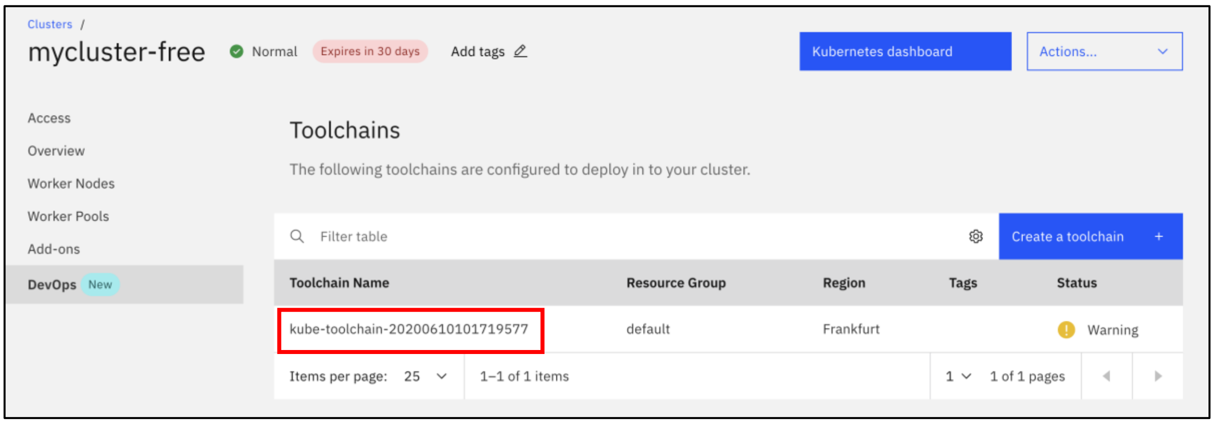


Expires in 30 days:
Be sure to back up your data, your cluster will be deleted in 30 days. To access the full capabilities of the service, try out a standard cluster.

Summary

Cluster ID	brg9qo8d0qpl200harq0	
Master status	Ready	
Version	1.17.6_1526	
Zones	hou02	
Creator	aco_vidovic@si.ibm.com	
Created	6/10/2020, 10:45 AM	
Resource group	default	
Logging		
Monitoring		
Key management service	Enable	
Image pull secrets	Enabled	
Public service endpoint URL	https://c5.dal12.containers.cloud.ibm.com:21554	Disable
Private service endpoint URL	Enable	

2. You will see the Toolchain you previously created. Click on it.


Clusters /

mycluster-free ✓ Normal Expires in 30 days Add tags

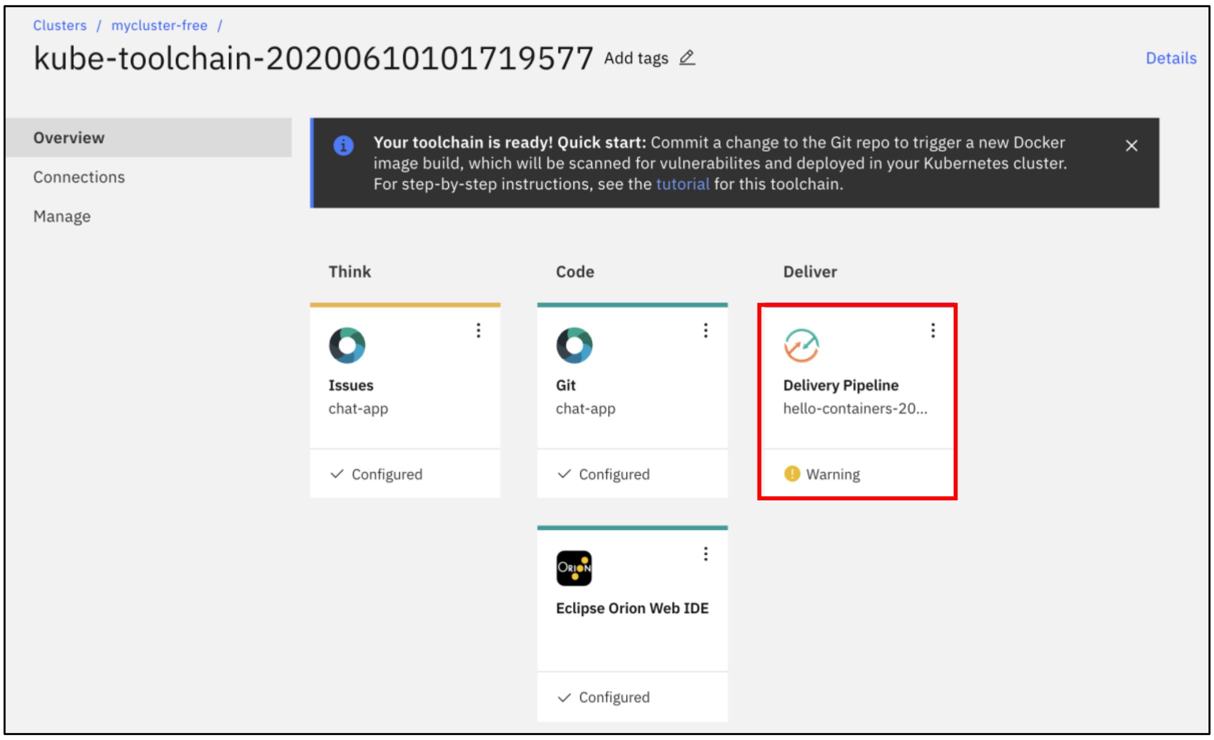
Toolchains

The following toolchains are configured to deploy in to your cluster.

Toolchain Name	Resource Group	Region	Tags	Status
kube-toolchain-20200610101719577	default	Frankfurt		Warning

Items per page: 25 1–1 of 1 items 1 1 of 1 pages

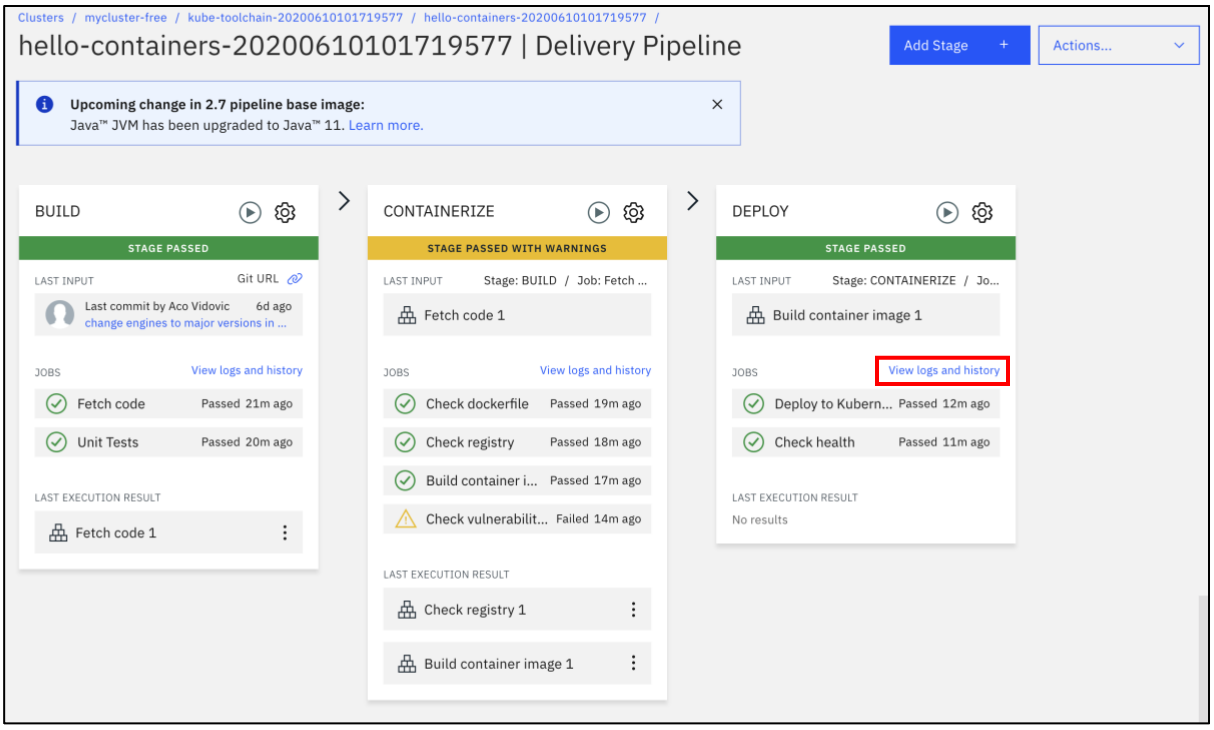
3. Click on the **Delivery Pipeline** tile.


Clusters / mycluster-free /

kube-toolchain-20200610101719577 Add tags

Your toolchain is ready! Quick start: Commit a change to the Git repo to trigger a new Docker image build, which will be scanned for vulnerabilites and deployed in your Kubernetes cluster. For step-by-step instructions, see the tutorial for this toolchain.

4. Notice that your pipeline consists of 3 stages: BUILD, CONTAINERIZE and DEPLOY.
5. In the **DEPLOY** tile, click on **View logs and history**.



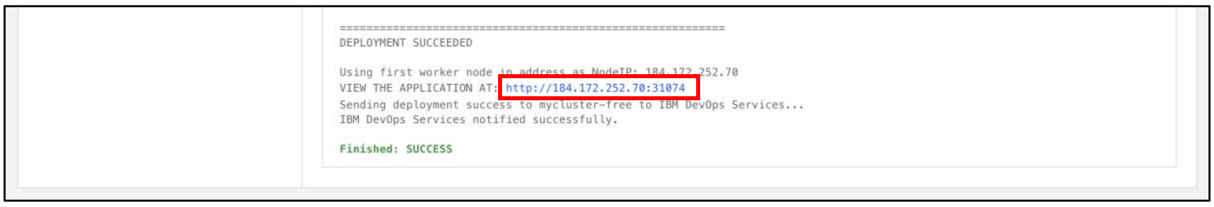
Clusters / mycluster-free / kube-toolchain-20200610101719577 / hello-containers-20200610101719577 /

hello-containers-20200610101719577 | Delivery Pipeline

Upcoming change in 2.7 pipeline base image:
Java™ JVM has been upgraded to Java™ 11. Learn more.

BUILD	CONTAINERIZE	DEPLOY
STAGE PASSED	STAGE PASSED WITH WARNINGS	STAGE PASSED
LAST INPUT: Git URL — Last commit by Aco Vidovic, 6d ago — change engines to major versions in ...	LAST INPUT: Stage: BUILD / Job: Fetch ... — Fetch code 1	LAST INPUT: Stage: CONTAINERIZE / Jo... — Build container image 1
JOBS: Fetch code, Passed 21m ago; Unit Tests, Passed 20m ago	JOBS: Check dockerfile, Passed 19m ago; Check registry, Passed 18m ago; Build container i..., Passed 17m ago; Check vulnerabilit..., Failed 14m ago	JOBS: Deploy to Kubern..., Passed 12m ago; Check health, Passed 11m ago
LAST EXECUTION RESULT: Fetch code 1	LAST EXECUTION RESULT: Check registry 1; Build container image 1	LAST EXECUTION RESULT: No results

6. In the Logs pane, scroll down to the bottom to find the URL of your demo app.



```
============================================================
DEPLOYMENT SUCCEEDED

Using first worker node ip address as NodeIP: 184.172.252.70
VIEW THE APPLICATION AT: http://184.172.252.70:31074
Sending deployment success to mycluster-free to IBM DevOps Services...
IBM DevOps Services notified successfully.

Finished: SUCCESS
```

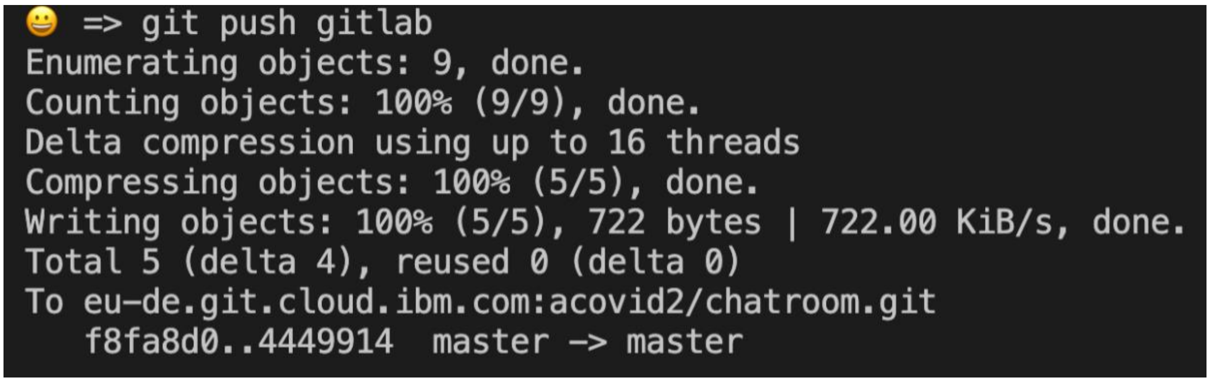
Note: the Toolchain has created a brand new URL for your demo app! It is on the same public IP address as the app you deployed manually, but on a different port.

7. Click on the URL on the above picture to test your automatically deployed app. Your app should look the same as in all previous deployments.

8. Test your Toochain. In next steps you will test if your CD really works. You will make some change in the app socde, commit it to your git repo and expect to see your change in the automatically deployed app.

   a) Using your source code editor, make some visible change in your app source code.
   b) Test your change locally.
   c) In your command line, commit the change to Git repo.

```
git push gitlab
```



```
😀  => git push gitlab
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 16 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 722 bytes | 722.00 KiB/s, done.
Total 5 (delta 4), reused 0 (delta 0)
To eu-de.git.cloud.ibm.com:acovid2/chatroom.git
   f8fa8d0..4449914  master -> master
```

d) At the same time observe your Toolchain in web browser. As the CD kicked in, you should see changes in status in different stages in tehg pipeline

---

End of Demo