Lab 03 Getting Started with Kubernetes

Contents

This lab is intended primarily for system administrator/infrastructure professionals who manage the Kubernetes cluster. This lab explains Kubernetes principles and the internal working of the cluster.

3.1 Introduction

This is "**Lab 03 Getting Started with Kubernetes**" from an IBM Cloud Pak for Applications & App Modernization Proof of technology (PoT). The labs are not required to be executed in order. And, you may skip labs, and only perform the labs that suit your desired learning objectives.

The full set of labs in the PoT are:

Lab01 - Getting started with Docker

Lab02 - Explore RedHat OpenShift Container Platform

Lab03 - Getting started with Kubernetes

Lab04 – Liberty application deployment using Operators

Lab05 – IBM Cloud Pak for Applications - App Modernization using Transformation Advisor

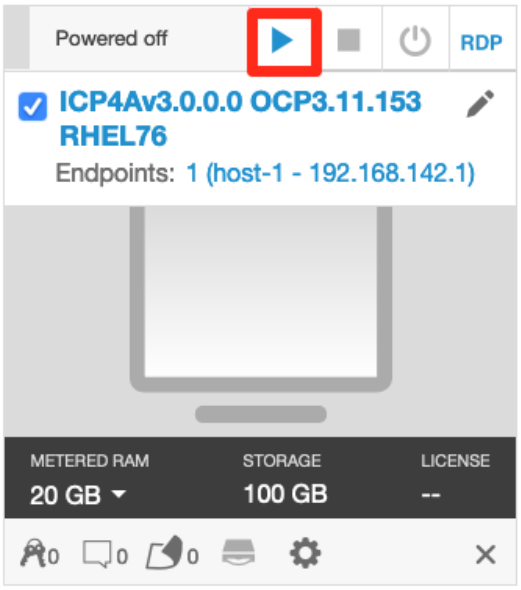
Lab06 – App Modernization with Java EE Microservices and Liberty

Lab07 – Using Tekton pipelines for CI/CD of microservices to RedHat OpenShift Container Platform

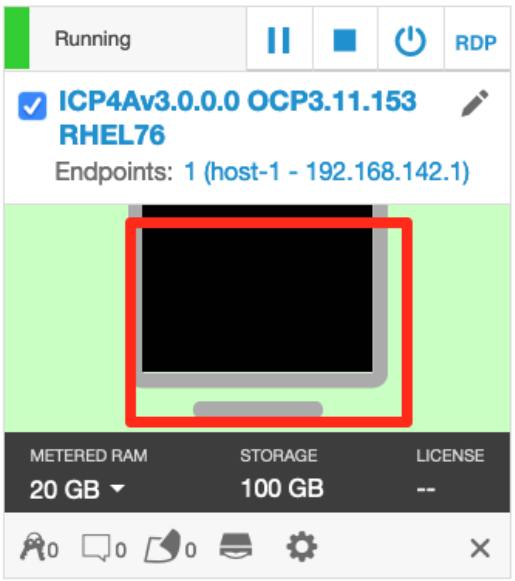
3.2 Let's get started

On your laptop/workstation, locate the `ICP4Av3.0.0.0 OCP3.11.153 RHEL76` virtual machine

__1. The VM should already be running. If not, Launch the Lab environment by clicking the **Run this VM** icon.

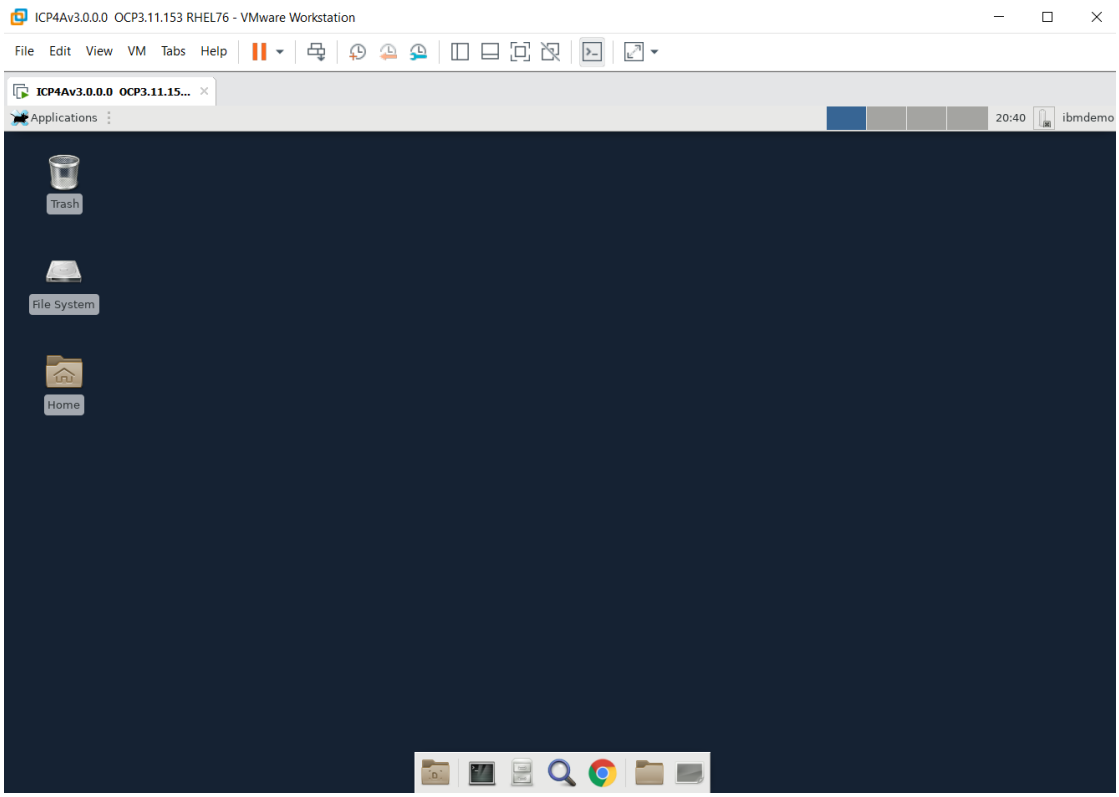


__2. After the VM is running, click its icon to access the VM's desktop.



__3. After the VM machine powers on, log with the `ibmdemo` user using the password `passw0rd`

The `ICP4Av3.0.0.0 OCP3.11.153 RHEL76` virtual machine running and its Desktop is displayed in a web browser window.



	Note: Refer to the **Appendix** in this lab guide for details for using Copy / Paste between the lab guide and the lab environment.

3.2.1 What is Kubernetes?

Kubernetes is an open-source platform for building an ecosystem of components and tools to deploy, scale and manage containerized applications. Kubernetes is often referred to as a *container orchestration framework*.

3.2.2 What is kubectl?

The `kubectl` is a command line tool to communicate with the Kubernetes master node that runs an API server. The API server provides REST API endpoints and `kubectl` internally uses the REST APIs to communicate with the API server, which communicates with the Kubernetes Objects in the cluster.

__1. **kubectl** must be configured for the environment that it will be executing against. Logging into the RHOCP cluster with the **oc** command line will provide the required configuration.

__a. Click `Terminal` from the bottom of the desktop to open a command line terminal.

__b. Type `oc login. Then enter ocpadmin` and `ocpadm1n` for the username and password.

```
[ibmdemo@icp4a ~]$ oc login

Authentication required for https://icp4a.pot.com:8443 (openshift)
Username: ocpadmin
Password:
Login successful.

You have access to the following projects and can switch between them with 'oc project
<projectname>':

  * default
    istio-system
    kabanero
    knative-eventing
    knative-serving
    knative-sources
    kube-public
    kube-service-catalog
    kube-system
    lab3
    management-infra
    openshift
    openshift-console
    openshift-infra
    openshift-logging
    openshift-metrics-server
    openshift-monitoring
    openshift-node
    openshift-node-problem-detector
    openshift-pipelines
    openshift-sdn
    openshift-web-console
    operator-lifecycle-manager
    ta
Using project "default".
```

The **kubectl command line interface** is now configured to communicate with the RHOCP cluster

__2. Run `kubectl version` command to find the Client and Kubernetes version.

```
[ibmdemo@icp4a ~]$ kubectl version

Client Version: version.Info{Major:"1", Minor:"11+", GitVersion:"v1.11.0+d4cacc0", GitCommit:"d4cacc0", GitTreeState:"clean",
        BuildDate:"2019-10-10T16:48:37Z", GoVersion:"go1.10.8", Compiler:"gc", Platform:"linux/amd64"}

Server Version: version.Info{Major:"1", Minor:"11+", GitVersion:"v1.11.0+d4cacc0", GitCommit:"d4cacc0", GitTreeState:"clean",
        BuildDate:"2019-10-10T16:48:37Z", GoVersion:"go1.10.8", Compiler:"gc", Platform:"linux/amd64"}
```

3.3 List Kubernetes Nodes

__3. List all the nodes in our cluster with the command `kubectl get nodes`

```
[ibmdemo@icp4a ~]$ kubectl get nodes
NAME           STATUS    ROLES               AGE       VERSION
icp4a.pot.com  Ready     compute,infra,master  7d        v1.11.0+d4cacc0
```

- This is an “all in one” cluster, with a single node configured for all nodes
- Master node(s) – run the Kubernetes API server, etcd, Controller Manager and HAProxy
- Infrastructure node(s) - run Kubernetes and OpenShift Container Platform control plane services that run on masters, the default router, the container image registry, the cluster metrics collection, or monitoring service, cluster aggregated logging and Service brokers
- Compute node(s) – run application workloads

__4. You can also use `kubectl describe node <nodename>` command to get more information about a specific node, in this case our single node is `icp4a.pot.com`

```
[ibmdemo@icp4a lab3]$ kubectl describe node icp4a.pot.com
Name:               icp4a.pot.com
Roles:              compute,infra,master
Labels:             beta.kubernetes.io/arch=amd64
                    beta.kubernetes.io/os=linux
                    kubernetes.io/hostname=icp4a.pot.com
                    node-role.kubernetes.io/compute=true
                    node-role.kubernetes.io/infra=true
                    node-role.kubernetes.io/master=true
Annotations:        node.openshift.io/md5sum=d59c38bb2c2e6553a869752ba72d3a6c
                    volumes.kubernetes.io/controller-managed-attach-detach=true
CreationTimestamp:  Tue, 29 Oct 2019 20:52:59 -0500
Taints:             <none>
Unschedulable:      false
Conditions:
  Type             Status  LastHeartbeatTime                 LastTransitionTime                Reason                       Message
  ----             ------  -----------------                 ------------------                ------                       -------
  KernelDeadlock   False   Thu, 05 Dec 2019 09:13:08 -0600   Mon, 02 Dec 2019 08:26:11 -0600   KernelHasNoDeadlock          kernel has no deadlock
  OutOfDisk        False   Thu, 05 Dec 2019 09:13:27 -0600   Tue, 29 Oct 2019 20:52:59 -0500   KubeletHasSufficientDisk     kubelet has sufficient disk space available
  MemoryPressure   False   Thu, 05 Dec 2019 09:13:27 -0600   Tue, 29 Oct 2019 20:52:59 -0500   KubeletHasSufficientMemory   kubelet has sufficient memory available
  DiskPressure     False   Thu, 05 Dec 2019 09:13:27 -0600   Tue, 29 Oct 2019 20:52:59 -0500   KubeletHasNoDiskPressure     kubelet has no disk pressure
  PIDPressure      False   Thu, 05 Dec 2019 09:13:27 -0600   Tue, 29 Oct 2019 20:52:59 -0500   KubeletHasSufficientPID      kubelet has sufficient PID available
  Ready            True    Thu, 05 Dec 2019 09:13:27 -0600   Mon, 02 Dec 2019 08:25:31 -0600   KubeletReady                 kubelet is posting ready status
Addresses:
```

```
 InternalIP:  192.168.142.130
 Hostname:    icp4a.pot.com
Capacity:
 cpu:           8
 hugepages-1Gi:  0
 hugepages-2Mi:  0
 memory:        20375412Ki
 pods:          250
Allocatable:
 cpu:           8
 hugepages-1Gi:  0
 hugepages-2Mi:  0
 memory:        20273012Ki
 pods:          250
System Info:
 Machine ID:                 c65e37fd605b43b089d47d8b903e9ace
 System UUID:                E8FB4D56-9CCF-1A83-4549-E3C664EF9DF2
 Boot ID:                    fb77f495-41a7-47b3-bce4-45a8a8dcf562
 Kernel Version:             3.10.0-1062.4.1.el7.x86_64
 OS Image:                   OpenShift Enterprise
 Operating System:           linux
 Architecture:               amd64
 Container Runtime Version:  docker://1.13.1
 Kubelet Version:            v1.11.0+d4cacc0
 Kube-Proxy Version:         v1.11.0+d4cacc0
Non-terminated Pods:         (60 in total)
  Namespace                  Name                                                  CPU Requests  CPU Limits  Memory Requests  Memory Limits
  ---------                  ----                                                  ------------  ----------  ---------------  -------------
  default                    docker-registry-1-m8977                               100m (1%)     0 (0%)      256Mi (1%)       0 (0%)
  default                    router-1-sjbdd                                        100m (1%)     0 (0%)      256Mi (1%)       0 (0%)
  istio-system               istio-ingressgateway-d897d9676-xzl42                  100m (1%)     2 (25%)     128Mi (0%)       1Gi (5%)
  istio-system               istio-pilot-dcf4bd85c-vlxxp                           500m (6%)     0 (0%)      2Gi (10%)        0 (0%)
  kabanero                   appsody-operator-549fd759c8-28gtx                     0 (0%)        0 (0%)      0 (0%)           0 (0%)
  kabanero                   controller-manager-0                                  100m (1%)     1 (12%)     100Mi (0%)       1000Mi (5%)
  kabanero                   icpa-landing-768684bb7c-hv2sw                         0 (0%)        0 (0%)      0 (0%)           0 (0%)
  kabanero                   kabanero-cli-654564cb49-tzz25                         0 (0%)        0 (0%)      0 (0%)           0 (0%)
  kabanero                   kabanero-landing-fcf8788cc-b5gdw                      0 (0%)        0 (0%)      0 (0%)           0 (0%)
  kabanero                   kabanero-operator-8667c666bc-glpfg                    0 (0%)        0 (0%)      0 (0%)           0 (0%)
  kabanero                   knative-eventing-operator-67cdf5dc9f-ljg6x            0 (0%)        0 (0%)      0 (0%)           0 (0%)
  kabanero                   knative-serving-operator-b64558bbc-4tvzh              0 (0%)        0 (0%)      0 (0%)           0 (0%)
  kabanero                   openshift-pipelines-operator-66c4d787cf-dhbnm         0 (0%)        0 (0%)      0 (0%)           0 (0%)
  kabanero                   tekton-dashboard-55fd66fbff-drcff                     0 (0%)        0 (0%)      0 (0%)           0 (0%)
  kabanero                   webhooks-extension-65d44777-hxlj6                     0 (0%)        0 (0%)      0 (0%)           0 (0%)
  knative-eventing           eventing-controller-57dc75c787-lf52q                  0 (0%)        0 (0%)      1000Mi (5%)      1000Mi (5%)
  knative-eventing           in-memory-channel-controller-d65c9bbdd-jd28s          0 (0%)        0 (0%)      0 (0%)           0 (0%)
  knative-eventing           in-memory-channel-dispatcher-5dcf5d557b-w44t7         0 (0%)        0 (0%)      0 (0%)           0 (0%)
```

```
  knative-eventing              sources-controller-5c65c58cfd-x7pqt                    100m (1%)     1 (12%)     100Mi (0%)      1000Mi (5%)
  knative-eventing              webhook-6cc6bbf964-trs89                               0 (0%)        0 (0%)      1000Mi (5%)     1000Mi (5%)
  knative-serving               activator-5647c87477-x7qrz                             20m (0%)      200m (2%)   60Mi (0%)       600Mi (3%)
  knative-serving               autoscaler-58746d858-69z88                             30m (0%)      300m (3%)   40Mi (0%)       400Mi (2%)
  knative-serving               controller-5f6d69cd4b-mlhhw                            100m (1%)     1 (12%)     100Mi (0%)      1000Mi (5%)
  knative-serving               knative-openshift-ingress-6464b574f4-rvpr8             0 (0%)        0 (0%)      0 (0%)          0 (0%)
  knative-serving               networking-certmanager-677bf4f846-nrxqd                100m (1%)     1 (12%)     100Mi (0%)      1000Mi (5%)
  knative-serving               networking-istio-6c885577bb-g2dt9                      100m (1%)     1 (12%)     100Mi (0%)      1000Mi (5%)
  knative-serving               webhook-7645bc789f-vtrdw                               20m (0%)      200m (2%)   20Mi (0%)       200Mi (1%)
  kube-service-catalog          apiserver-jghnh                                        0 (0%)        0 (0%)      0 (0%)          0 (0%)
  kube-service-catalog          controller-manager-n98hx                               0 (0%)        0 (0%)      0 (0%)          0 (0%)
  kube-system                   master-api-icp4a.pot.com                               0 (0%)        0 (0%)      0 (0%)          0 (0%)
  kube-system                   master-controllers-icp4a.pot.com                       0 (0%)        0 (0%)      0 (0%)          0 (0%)
  kube-system                   master-etcd-icp4a.pot.com                              0 (0%)        0 (0%)      0 (0%)          0 (0%)
  openshift-console             console-dd447dd5c-h2kqr                                100m (1%)     100m (1%)   100Mi (0%)      100Mi (0%)
  openshift-infra               hawkular-cassandra-1-2k2gr                             800m (10%)    800m (10%)  1G (4%)         2Gi (10%)
  openshift-infra               hawkular-metrics-tgpbr                                 800m (10%)    800m (10%)  1500M (7%)      2Gi (10%)
  openshift-infra               heapster-m2td8                                         800m (10%)    800m (10%)  937500k (4%)    2Gi (10%)
  openshift-metrics-server      metrics-server-56ff69fdff-snk4t                        0 (0%)        0 (0%)      0 (0%)          0 (0%)
  openshift-monitoring          alertmanager-main-0                                    5m (0%)       5m (0%)     210Mi (1%)      10Mi (0%)
  openshift-monitoring          alertmanager-main-1                                    5m (0%)       5m (0%)     210Mi (1%)      10Mi (0%)
  openshift-monitoring          alertmanager-main-2                                    5m (0%)       5m (0%)     210Mi (1%)      10Mi (0%)
  openshift-monitoring          cluster-monitoring-operator-57647f5877-qpckw           20m (0%)      20m (0%)    50Mi (0%)       50Mi (0%)
  openshift-monitoring          grafana-77cb866df7-dcbws                               100m (1%)     200m (2%)   100Mi (0%)      200Mi (1%)
  openshift-monitoring          kube-state-metrics-59c45bb4f6-b9hz9                    20m (0%)      40m (0%)    40Mi (0%)       80Mi (0%)
  openshift-monitoring          node-exporter-vpsqp                                    10m (0%)      20m (0%)    20Mi (0%)       40Mi (0%)
  openshift-monitoring          prometheus-k8s-0                                       15m (0%)      15m (0%)    60Mi (0%)       60Mi (0%)
  openshift-monitoring          prometheus-k8s-1                                       15m (0%)      15m (0%)    60Mi (0%)       60Mi (0%)
  openshift-monitoring          prometheus-operator-6bc9848445-pdbm6                   0 (0%)        0 (0%)      0 (0%)          0 (0%)
  openshift-node-problem-detector  node-problem-detector-zpkqm                         0 (0%)        0 (0%)      0 (0%)          0 (0%)
  openshift-node                sync-ndr8t                                             0 (0%)        0 (0%)      0 (0%)          0 (0%)
  openshift-pipelines           tekton-pipelines-controller-6c9778c6d4-x78sp           0 (0%)        0 (0%)      0 (0%)          0 (0%)
  openshift-pipelines           tekton-pipelines-webhook-76686fd9c7-bdpnm              0 (0%)        0 (0%)      0 (0%)          0 (0%)
  openshift-sdn                 ovs-dq9xf                                              100m (1%)     0 (0%)      300Mi (1%)      0 (0%)
  openshift-sdn                 sdn-xkrtz                                              100m (1%)     0 (0%)      200Mi (1%)      0 (0%)
  openshift-web-console         webconsole-85777774db-454rv                            100m (1%)     0 (0%)      100Mi (0%)      0 (0%)
  operator-lifecycle-manager    catalog-operator-96d84b9d-8bfzs                        0 (0%)        0 (0%)      0 (0%)          0 (0%)
  operator-lifecycle-manager    olm-operator-89dcb84cf-qvlps                           0 (0%)        0 (0%)      0 (0%)          0 (0%)
  ta                            ta-9mwaqdq2nrskos5kkb12c9ss2-ta-rh-couchdb-5d9f74586c-md9d2  500m (6%)  16 (200%)  1Gi (5%)  8Gi (41%)
  ta                            ta-9mwaqdq2nrskos5kkb12c9ss2-ta-rh-server-6d4c6c6b6c-wsxwh  500m (6%)  16 (200%)  1Gi (5%)  4Gi (20%)
  ta                            ta-9mwaqdq2nrskos5kkb12c9ss2-ta-rh-ui-75bbc6c6d6-x6f8h  500m (6%)  16 (200%)  1Gi (5%)  4Gi (20%)
  ta                            ta-operator-649b6b7c68-qs58f                           0 (0%)        0 (0%)      0 (0%)          0 (0%)
Allocated resources:
  (Total limits may be over 100 percent, i.e., overcommitted.)
  Resource  Requests          Limits
  --------  --------          ------
```

```
 cpu      5865m (73%)       58525m (731%)
 memory   13965203040 (67%)  32372Mi (163%)
Events:   <none>
```

__a. In reviewing the output above we see that this node is running the Linux OS and is a master node. Next, see the information about the operation of node itself.

__b. Following that is the status of the node which indicates that it has sufficient disk and memory, and it is in `Ready` state. The information about capacity displays next.

__c. Next the kernel and OS information for the node. (which is followed by a list of running pods and their resource consumption)

3.4 Check running pods

__1. Run the command `kubectl get pods`. Note: The name space is set when logging into kubectl. When the `-n` switch is not defined, the default namespace is `default` is this case.

```
[ibmdemo@icp4a ~]$ kubectl get pods

NAME                     READY     STATUS    RESTARTS   AGE
docker-registry-1-m8977   1/1       Running   7          36d
router-1-sjbdd            1/1       Running   7          36d
```

__2. Run the commands `kubectl get pods --all-namespaces` to get the list of running pods in all namespaces.

```
[ibmdemo@icp4a lab3]$ kubectl get pods --all-namespaces
NAMESPACE                NAME                                      READY     STATUS    RESTARTS   AGE
default                  docker-registry-1-m8977                   1/1       Running   7          36d
default                  router-1-sjbdd                            1/1       Running   7          36d
istio-system             istio-ingressgateway-d897d9676-xzl42      1/1       Running   5          31d
istio-system             istio-pilot-dcf4bd85c-vlxxp               1/1       Running   5          31d
kabanero                 appsody-operator-549fd759c8-28gtx         1/1       Running   5          31d
kabanero                 controller-manager-0                      1/1       Running   7          31d
kabanero                 icpa-landing-768684bb7c-hv2sw             1/1       Running   5          31d
kabanero                 kabanero-cli-654564cb49-tzz25             1/1       Running   5          31d
kabanero                 kabanero-landing-fcf8788cc-b5gdw          1/1       Running   5          31d
kabanero                 kabanero-operator-8667c666bc-glpfg        1/1       Running   5          31d
kabanero                 knative-eventing-operator-67cdf5dc9f-ljg6x 1/1      Running   5          31d
kabanero                 knative-serving-operator-b64558bbc-4tvzh  1/1       Running   5          31d
kabanero                 openshift-pipelines-operator-66c4d787cf-dhbnm 1/1    Running   5          31d
kabanero                 tekton-dashboard-55fd66fbff-drcff         2/2       Running   10         31d
kabanero                 webhooks-extension-65d44777-hxlj6         1/1       Running   5          31d
knative-eventing         eventing-controller-57dc75c787-lf52q      1/1       Running   5          31d
knative-eventing         in-memory-channel-controller-d65c9bbdd-jd28s 1/1    Running   5          31d
knative-eventing         in-memory-channel-dispatcher-5dcf5d557b-w44t7 1/1   Running   5          31d
knative-eventing         sources-controller-5c65c58cfd-x7pqt       1/1       Running   5          31d
knative-eventing         webhook-6cc6bbf964-trs89                  1/1       Running   5          31d
knative-serving          activator-5647c87477-x7qrz                1/1       Running   17         31d
knative-serving          autoscaler-58746d858-69z88                1/1       Running   5          31d
knative-serving          controller-5f6d69cd4b-mlhhw               1/1       Running   5          31d
knative-serving          knative-openshift-ingress-6464b574f4-rvpr8 1/1      Running   5          31d
knative-serving          networking-certmanager-677bf4f846-nrxqd   1/1       Running   5          31d
knative-serving          networking-istio-6c885577bb-g2dt9         1/1       Running   5          31d
knative-serving          webhook-7645bc789f-vtrdw                  1/1       Running   5          31d
kube-service-catalog     apiserver-jghnh                           1/1       Running   15         36d
kube-service-catalog     controller-manager-n98hx                  1/1       Running   35         36d
kube-system              master-api-icp4a.pot.com                  1/1       Running   7          36d
```

```
kube-system                    master-controllers-icp4a.pot.com                  1/1     Running     7          36d
kube-system                    master-etcd-icp4a.pot.com                         1/1     Running     7          36d
openshift-console              console-dd447dd5c-h2kqr                           1/1     Running     7          36d
openshift-infra                hawkular-cassandra-1-2k2gr                        1/1     Running     7          36d
openshift-infra                hawkular-metrics-schema-jgrk6                     0/1     Completed   0          36d
openshift-infra                hawkular-metrics-tgpbr                            1/1     Running     7          36d
openshift-infra                heapster-m2td8                                    1/1     Running     7          36d
openshift-metrics-server       metrics-server-56ff69fdff-snk4t                   1/1     Running     9          36d
openshift-monitoring           alertmanager-main-0                               3/3     Running     21         36d
openshift-monitoring           alertmanager-main-1                               3/3     Running     21         36d
openshift-monitoring           alertmanager-main-2                               3/3     Running     21         36d
openshift-monitoring           cluster-monitoring-operator-57647f5877-qpckw      1/1     Running     7          36d
openshift-monitoring           grafana-77cb866df7-dcbws                          2/2     Running     14         36d
openshift-monitoring           kube-state-metrics-59c45bb4f6-b9hz9               3/3     Running     21         36d
openshift-monitoring           node-exporter-vpsqp                               2/2     Running     14         36d
openshift-monitoring           prometheus-k8s-0                                  4/4     Running     29         36d
openshift-monitoring           prometheus-k8s-1                                  4/4     Running     29         36d
openshift-monitoring           prometheus-operator-6bc9848445-pdbm6              1/1     Running     7          36d
openshift-node-problem-detector  node-problem-detector-zpkqm                     1/1     Running     7          36d
openshift-node                 sync-ndr8t                                        1/1     Running     7          36d
openshift-pipelines            tekton-pipelines-controller-6c9778c6d4-x78sp      1/1     Running     5          31d
openshift-pipelines            tekton-pipelines-webhook-76686fd9c7-bdpnm         1/1     Running     5          31d
openshift-sdn                  ovs-dq9xf                                         1/1     Running     7          36d
openshift-sdn                  sdn-xkrtz                                         1/1     Running     7          36d
openshift-web-console          webconsole-85777774db-454rv                       1/1     Running     9          36d
operator-lifecycle-manager     catalog-operator-96d84b9d-8bfzs                   1/1     Running     7          36d
operator-lifecycle-manager     olm-operator-89dcb84cf-qvlps                      1/1     Running     7          36d
ta             ta-9mwaqdq2nrskos5kkb12c9ss2-ta-rh-couchdb-5d9f74586c-md9d2       1/1     Running     5          31d
ta             ta-9mwaqdq2nrskos5kkb12c9ss2-ta-rh-server-6d4c6c6b6c-wsxwh         1/1     Running     5          31d
ta             ta-9mwaqdq2nrskos5kkb12c9ss2-ta-rh-ui-75bbc6c6d6-x6f8h            1/1     Running     5          31d
ta             ta-operator-649b6b7c68-qs58f                                      1/1     Running     5          31d
```

__3. Review the function of each *OpenShift Container Platform pod* (the list omits the IBM Cloud Pak for Applications pods in the `istio-system, kababero, knative-eventing, knative-serving, openshift-pipelines` and `ta` inamespaces)

Pod	Description
`docker-registry`	Internal image registry
`router-1`	Directs service requests to the service endpoints
`apiserver`	handles all api requests
`controller-manager`	watches etcd for changes to replication controller objects and then uses the API to enforce the desired state.
`master-controllers`	Scheduler and Replication Controller, responsible for the placement and maintenance of pods
`master-etcd`	kubernetes etcd database to hold state of cluster
`console, web-console`	web console
`hawkular-metrics, hawkular-metrics-schema`	metrics engine and schema
`hawkular-cassandra`	Cassandra database for metrics
`heapster`	scrapes the metrics for CPU, memory and network usage for each node then exports them into Hawkular Metrics
`metrics-server`	Cluster-wide aggregator of resource usage data
`alertmanager-main`	manages incoming alerts; this includes silencing, inhibition, aggregation, and sending out notifications through methods such as email, PagerDuty,
`cluster-monitoring-operator`	watches over the deployed monitoring components and resources, and ensures that they are up to date
`grafana`	cluster monitoring dashboard interface
`kube-state-metrics`	converts Kubernetes objects to metrics consumable by Prometheus
`node-exporter`	agent deployed on every node to collect node metrics
`prometheus-k8s`	Prometheus
`node-problem-detector`	monitors node health and reportsproblems to the API server
`sync`	detects configuration map change, updates the *node-config.yaml* and restarts the appropriate nodes
`ovs, sdn`	OpenShift software-defined networking (SDN) which configures an overlay network using Open vSwitch (OVS).
`catalog-operator`	responsible for resolving and installing ClusterServiceVersions (CSVs) and the Custom Resource Definitions (CRD) specified resources
`olm-operator`	install, update, and manage the lifecycle of all Operators and their associated services

__4. Note that **master node** runs three main components of Kubernetes.

1. *API Server*

2. *Schedule*

3. *Controller Manager*

__5. The fourth component of Kubernetes is the **etcd database** that holds the **state of the cluster**. The `etcd` container runs in a separate pod.

__a. Run the following kubectl -n kube-system get pods | grep -i etcd to see the etcd ods running

```
[ibmdemo@icp4a ~]$ kubectl -n kube-system get pods | grep -i etcd

master-etcd-icp4a.pot.com          1/1       Running   22         36d
```

3.5 Kubernetes Master Node

__1. Run command `kubectl get nodes`

```
[ibmdemo@icp4a ~]$ kubectl get [ibmdemo@icp4a lab3]$ kubectl get nodes

NAME           STATUS    ROLES               AGE       VERSION
icp4a.pot.com   Ready     compute,infra,master   36d       v1.11.0+d4cacc0
```

__2. Run the following command `kubectl -n kube-system get pods | grep master` to find the Kubernetes pods associated with the master.

```
[ibmdemo@icp4a lab3]$ kubectl -n kube-system get pods | grep master

master-api-icp4a.pot.com           1/1       Running   7          36d
master-controllers-icp4a.pot.com   1/1       Running   7          36d
master-etcd-icp4a.pot.com          1/1       Running   7          36d
```

__3. The kubectl command `kubectl -n <namespace> get pod <podname>` can be used to retrieve information about a specific pod Kubernetes pods.

__a. Type `kubectl -n kube-system get pod master-controllers-icp4a.pot.com`

```
[ibmdemo@icp4a lab3]$  kubectl get pod -n kube-system  master-controllers-icp4a.pot.com

NAME                          READY     STATUS    RESTARTS   AGE
master-controllers-icp4a.pot.com   1/1       Running   7          36d
```

`kubectl` is a client that sends REST API to the API server and parses the JSON return output.

__b. Append `-o json` parameter to the above command to see the JSON output from the API server.

```
kubectl -n kube-system get pod master-controllers-icp4a.pot.com -o json
```

```
[ibmdemo@icp4a lab3]$ kubectl -n kube-system get pod master-controllers-icp4a.pot.com -o json
{
    "apiVersion": "v1",
    "kind": "Pod",
    "metadata": {
        "annotations": {
            "kubernetes.io/config.hash": "23d68adcbc24fd01e20c4ea411db1143",
            "kubernetes.io/config.mirror": "23d68adcbc24fd01e20c4ea411db1143",
            "kubernetes.io/config.seen": "2019-10-29T20:52:35.469712434-05:00",
            "kubernetes.io/config.source": "file",
            "scheduler.alpha.kubernetes.io/critical-pod": ""
        },
        "creationTimestamp": "2019-10-30T01:53:40Z",
        "labels": {
            "openshift.io/component": "controllers",
            "openshift.io/control-plane": "true"
        },
        "name": "master-controllers-icp4a.pot.com",
        "namespace": "kube-system",
        "resourceVersion": "364603",
        "selfLink": "/api/v1/namespaces/kube-system/pods/master-controllers-icp4a.pot.com",
        "uid": "189cb711-fab8-11e9-981d-000c29ef9df2"
    },
    "spec": {
        "containers": [
            {
                "args": [
                    "#!/bin/bash\nset -euo pipefail\nif [[ -f /etc/origin/master/master.env ]]; then\n  set -o allexport\n  source
/etc/origin/master/master.env\nfi\nexec openshift start master controllers --config=/etc/origin/master/master-config.yaml --
listen=https://0.0.0.0:8444 --loglevel=${DEBUG_LOGLEVEL:-2}\n"
                ],
                "command": [
                    "/bin/bash",
                    "-c"
                ],
                "image": "registry.redhat.io/openshift3/ose-control-plane:v3.11.153",
                "imagePullPolicy": "IfNotPresent",
                "livenessProbe": {
                    "failureThreshold": 3,
                    "httpGet": {
                        "path": "healthz",
                        "port": 8444,
                        "scheme": "HTTPS"
                    },
                    "periodSeconds": 10,
                    "successThreshold": 1,
                    "timeoutSeconds": 1
                },
                "name": "controllers",
                "resources": {},
                "securityContext": {
                    "privileged": true
                },
                "terminationMessagePath": "/dev/termination-log",
                "terminationMessagePolicy": "File",
                "volumeMounts": [
                    {
                        "mountPath": "/etc/origin/master/",
                        "name": "master-config"
                    },
                    {
                        "mountPath": "/etc/origin/cloudprovider/",
                        "name": "master-cloud-provider"
```

```
            },
            {
              "mountPath": "/etc/containers/registries.d/",
              "name": "signature-import"
            },
            {
              "mountPath": "/usr/libexec/kubernetes/kubelet-plugins",
              "mountPropagation": "HostToContainer",
              "name": "kubelet-plugins"
            },
            {
              "mountPath": "/etc/pki",
              "name": "master-pki"
            },
            {
              "mountPath": "/etc/localtime",
              "name": "host-localtime"
            }
          ]
        }
      ],
      "dnsPolicy": "ClusterFirst",
      "hostNetwork": true,
      "nodeName": "icp4a.pot.com",
      "priority": 2000001000,
      "priorityClassName": "system-node-critical",
      "restartPolicy": "Always",
      "schedulerName": "default-scheduler",
      "securityContext": {},
      "terminationGracePeriodSeconds": 30,
      "tolerations": [
        {
          "effect": "NoExecute",
          "operator": "Exists"
        }
      ],
      "volumes": [
        {
          "hostPath": {
            "path": "/etc/origin/master/",
            "type": ""
          },
          "name": "master-config"
        },
        {
          "hostPath": {
            "path": "/etc/origin/cloudprovider",
            "type": ""
          },
          "name": "master-cloud-provider"
        },
        {
          "hostPath": {
            "path": "/etc/containers/registries.d",
            "type": ""
          },
          "name": "signature-import"
        },
        {
          "hostPath": {
            "path": "/usr/libexec/kubernetes/kubelet-plugins",
            "type": ""
          },
          "name": "kubelet-plugins"
        },
        {
          "hostPath": {
            "path": "/etc/pki",
            "type": ""
          },
          "name": "master-pki"
        },
        {
          "hostPath": {
            "path": "/etc/localtime",
            "type": ""
          },
          "name": "host-localtime"
        }
```

```
        ]
    },
    "status": {
        "conditions": [
            {
                "lastProbeTime": null,
                "lastTransitionTime": "2019-12-02T14:24:42Z",
                "status": "True",
                "type": "Initialized"
            },
            {
                "lastProbeTime": null,
                "lastTransitionTime": "2019-12-02T14:24:43Z",
                "status": "True",
                "type": "Ready"
            },
            {
                "lastProbeTime": null,
                "lastTransitionTime": null,
                "status": "True",
                "type": "ContainersReady"
            },
            {
                "lastProbeTime": null,
                "lastTransitionTime": "2019-12-02T14:24:42Z",
                "status": "True",
                "type": "PodScheduled"
            }
        ],
        "containerStatuses": [
            {
                "containerID": "docker://9c676cffc9a64149f981c0259e99303f551af6393976724bff36e26a3073f770",
                "image": "registry.redhat.io/openshift3/ose-control-plane:v3.11.153",
                "imageID": "docker-pullable://registry.redhat.io/openshift3/ose-control-
plane@sha256:3dd73928e82806dab76e9aa6d2cbc9695b04b42132c41202c5de18031510bab2",
                "lastState": {},
                "name": "controllers",
                "ready": true,
                "restartCount": 7,
                "state": {
                    "running": {
                        "startedAt": "2019-12-02T14:24:43Z"
                    }
                }
            }
        ],
        "hostIP": "192.168.142.130",
        "phase": "Running",
        "podIP": "192.168.142.130",
        "qosClass": "BestEffort",
        "startTime": "2019-12-02T14:24:42Z"
    }
}
[ibmdemo@icp4a lab3]$
```


__4. Alternatively, instead of `-o json`, use `-o yaml` to obtain **yaml** formatted output

```
kubectl -n kube-system get pod master-controllers-icp4a.pot.com -o yaml
```

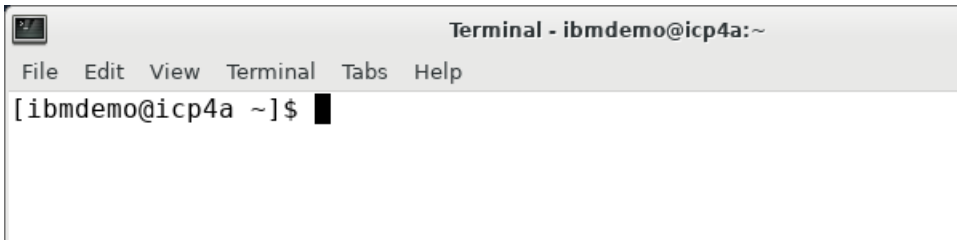
3.6 Let's create our first Pod

__1. Build the Docker Image.

__a. From the `Terminal` window, change to the **student/lab1** directory

```
cd student/lab1
```

__b. You are running as the user `ibmdemo`



__c. Build a Liberty docker image named simpleapp by typing `docker build -t simpleapp .` (note the "." at the end of the command, which will build an image named `simpleapp` using the Dockerfile in the local directory ".”

```
[ibmdemo@icp4a lab1]$ docker build -t simpleapp .

Sending build context to Docker daemon 11.78 kB
Step 1/7 : FROM docker.io/ibmcom/websphere-liberty:19.0.0.6-kernel-ubi-min
 ---> 7810d7fa4666
Step 2/7 : COPY server.xml /config/
 ---> Using cache
 ---> 6edb30c0af77
Step 3/7 : COPY ServletApp.war /config/apps/
 ---> a3723ec150d9
Removing intermediate container 07c7632f0288
Step 4/7 : USER root
 ---> Running in e13bdf4044d8
 ---> 187f8d0995ed
Removing intermediate container e13bdf4044d8
Step 5/7 : RUN chown default:root -R /opt/ibm/wlp/usr/servers/defaultServer
 ---> Running in 94ee6f6f54b4

 ---> 44b1984ac0c4
Removing intermediate container 94ee6f6f54b4
Step 6/7 : USER 1001
 ---> Running in b8131fa83b3e
 ---> 78589551606b
Removing intermediate container b8131fa83b3e
Step 7/7 : RUN configure.sh
 ---> Running in 5cb8b3f4a436

+ WLP_INSTALL_DIR=/opt/ibm/wlp
+ SHARED_CONFIG_DIR=/opt/ibm/wlp/usr/shared/config
+ SHARED_RESOURCE_DIR=/opt/ibm/wlp/usr/shared/resources
+ SNIPPETS_SOURCE=/opt/ibm/helpers/build/configuration_snippets
+ SNIPPETS_TARGET=/config/configDropins/overrides
+ mkdir -p /config/configDropins/overrides
+ '[' '' == true ']'
```

```
+ '[' '' == true ']'
+ '[' '' == true ']'
+ '[' '' == true ']'
+ '[' '' == true ']'
+ '[' '' == true ']'
+ '[' '' == true ']'
+ '[' '' == client ']'
+ '[' '' == embedded ']'
+ '[' '' == true ']'
+ '[' '' == true ']'
+ installUtility install --acceptLicense defaultServer
Checking for missing features required by the server ...
The server requires the following additional features: servlet-3.1.  Installing features from the repository ...
Establishing a connection to the configured repositories  . . .
This process might take several minutes to complete.

Successfully connected to all configured repositories.

Preparing assets for installation. This process might take several minutes to complete.

Additional Liberty features must be installed for this server.

To install the additional features, review and accept the feature license agreement:
The --acceptLicense argument was found. This indicates that you have
accepted the terms of the license agreement.


Step 1 of 4: Downloading servlet-3.1 ...
Step 2 of 4: Installing servlet-3.1 ...
Step 3 of 4: Validating installed fixes ...
Step 4 of 4: Cleaning up temporary files ...


All assets were successfully installed.

Start product validation...
Product validation completed successfully.
+ find /opt/ibm/fixes -type f -name '*.jar' -print0
+ sort -z
+ xargs -0 -n 1 -r -I '{}' java -jar '{}' --installLocation /opt/ibm/wlp
+ find /opt/ibm/wlp -perm -g=w -print0
+ xargs -0 -r chmod -R g+rw
+ /opt/ibm/wlp/bin/server start

Starting server defaultServer.
Server defaultServer started with process ID 103.
+ /opt/ibm/wlp/bin/server stop

Stopping server defaultServer.
Server defaultServer stopped.
+ rm -rf /output/resources/security/ /output/messaging /logs/console.log /logs/messages.log
/logs/messages_19.12.04_20.44.34.0.log /opt/ibm/wlp/output/.classCache
+ chmod -R g+rwx /opt/ibm/wlp/output/defaultServer
+ find /opt/ibm/wlp -type d -perm -g=x -print0
+ xargs -0 -r chmod -R g+rwx
 ---> 59258a04abcf
Removing intermediate container 5cb8b3f4a436
Successfully built 59258a04abcf
```

__2. Type `cd ~/student/lab3` to switch to the lab3 directory.

__3. Create a new OSCP project named lab3, using the command below:

oc new-project lab3

__4. Type `oc project lab3` to ensure your are using the "**lab3**" project and namespace

```
[ibmdemo@icp4a lab3]$ oc project lab3

Now using project "lab3" on server "https://icp4a.pot.com:8443"
```

__5. We're going to use the simpleapp docker image that you built earlier in this lab. Now you need to tag the image and place the image in the RHOCP image registry using the following commands:

docker tag simpleapp:latest docker-registry.default.svc:5000/lab3/simpleapp:latest

docker login -u $(oc whoami) -p $(oc whoami -t) docker-registry.default.svc:5000

docker push docker-registry.default.svc:5000/lab3/simpleapp:latest

```
 [ibmdemo@icp4a lab3]$ docker tag simpleapp:latest docker-registry.default.svc:5000/lab3/simpleapp:latest

[ibmdemo@icp4a lab3]$ docker login -u $(oc whoami) -p $(oc whoami -t) docker-registry.default.svc:5000
Login Succeeded

[ibmdemo@icp4a lab3]$ docker push docker-registry.default.svc:5000/lab3/simpleapp:latest
The push refers to a repository [docker-registry.default.svc:5000/lab3/simpleapp]
0dc8daca8b18: Pushed
33dc8e0893d4: Pushed
fb45bcc4e8b2: Pushed
7777f66dc097: Pushed
892390baf45b: Pushed
3d83e899392d: Pushed
f08ff9c9da21: Pushed
22420617a754: Layer already exists
630e588850a9: Layer already exists
8b556f7654d6: Layer already exists
0882c6107de6: Layer already exists
4b8cddade548: Layer already exists
7ef66f795d0f: Layer already exists
bc9213f684de: Layer already exists
25631575241e: Layer already exists
481324a7ba6d: Layer already exists
26429bebe019: Layer already exists
latest: digest: sha256:b9a462bbaccbc10fef10bd422c12d680cde4ae7ecd96852cb98b8fd0d4530023 size: 3874
```

The docker tag command tags the docker image for the RHOCP registry

The docker login command logs you into the OpenShift internal registry, using the OpenShift username and password that you are currently logged in OpenShift.

The docker push command pushes the docker image to the OpenShift internal registry

__6. Run `cat kube01.yaml` to review the manifest file that will be used to deploy a simpleapp pod to OpenShift

```
[ibmdemo@icp4a lab3]$ cat kube01.yaml
# A simple yaml file to create a pod for simpleapp
apiVersion: v1
kind: Pod
metadata:
  name: simpleapp
  labels:
    app: simpleapp
spec:
  containers:
    - image: docker-registry.default.svc:5000/lab3/simpleapp1:latest
      name: simpleapp
      ports:
      - containerPort: 9080
        protocol: TCP
```

3.6.1 The structure of manifest file (YAML or JSON)

- The manifest file can be written either in YAML or JSON.
- The `yaml` is not new and has been around for 17 years. It started as `Yet Another Markup Language` (YAML) but now it is described as `YAML Ain't Markup Language`.
- The YAML/JSON manifest files are used extensively in Kubernetes.

Note that the YAML and JSON are interchangeable in Kubernetes. The YAML is preferred as it is human readable and can contain comments (not possible in JSON).

A basic understanding of the organization of the manifest file is needed to create resources in Kubernetes. Review the `kube01.yaml` manifest file and notice that it has the following structure:

- We need to define which Kubernetes API is to be used. The name is `apiVersion.`
- The possible values of `apiVersion` are `v1`, `apps/v1`, `v1beta1`, `v1beta2`, `batch/v1`, `extension/v1beta1` and several others. Refer to API documentation at `http://kubernetes.io`.
- The second name-value pair is `kind`, which can be `Pod`, `PodList`, `Service`, `Deployment`, `DeamonSet`, `ReplicaSet`, `Job` and many others.
- The third name-value pair is `metadata`, which describes information such as `name`, `annotations`, `labels`, `namespace` etc.
- The fourth name-value pair is `spec`, which defines containers, their name, image name, and the commands to run with start-up options.

We have used a basic YAML file to create a Pod which uses Docker image `simpleapp` and names the Pod `simpleapp`.

3.6.2 Create Pod

__1. Run `kubectl apply -f kube01.yaml` to create the Pod.

```
[ibmdemo@icp4a lab3]$ kubectl apply -f kube01.yaml

pod/simpleapp created
```

__2. Run `kubectl get pods`

```
[ibmdemo@icp4a lab3]$ kubectl get pods

NAME       READY    STATUS    RESTARTS   AGE
simpleapp  1/1      Running   0          1m
```

__3. Since we're using an OpenShift project, **lab3**, we're already scoped to the **lab3 namespace** so the --namespace or -n argument isn't needed.

__4. Check the pod again. Run `kubectl get pods -o wide`

```
[ibmdemo@icp4a lab3]$ kubectl get pods -o wide

NAME       READY    STATUS    RESTARTS   AGE      IP          NODE          NOMINATED NODE
simpleapp  1/1      Running   0          2m       10.128.0.45 icp4a.pot.com <none>
```

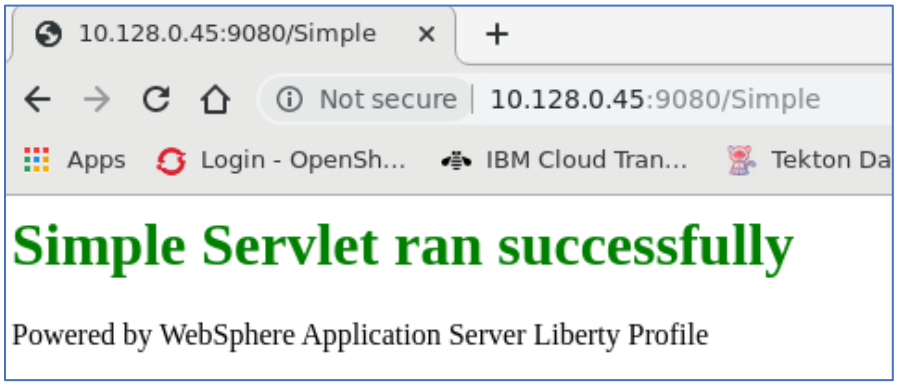
Notice that the pod is created on our only compute node. If there were multiple compute nodes, it could be deployed on any of them.

Also notice that the cluster assigned IP address is `10.128.0.45` (This will likely be different in your case and use your IP address to open the web page.)

__5. To access the simpleapp application, you have to use your cluster assigned IP address for this Pod which in the case shown above is `10.128.045` (This will likely be different in your case and use your IP address to open the web page.)

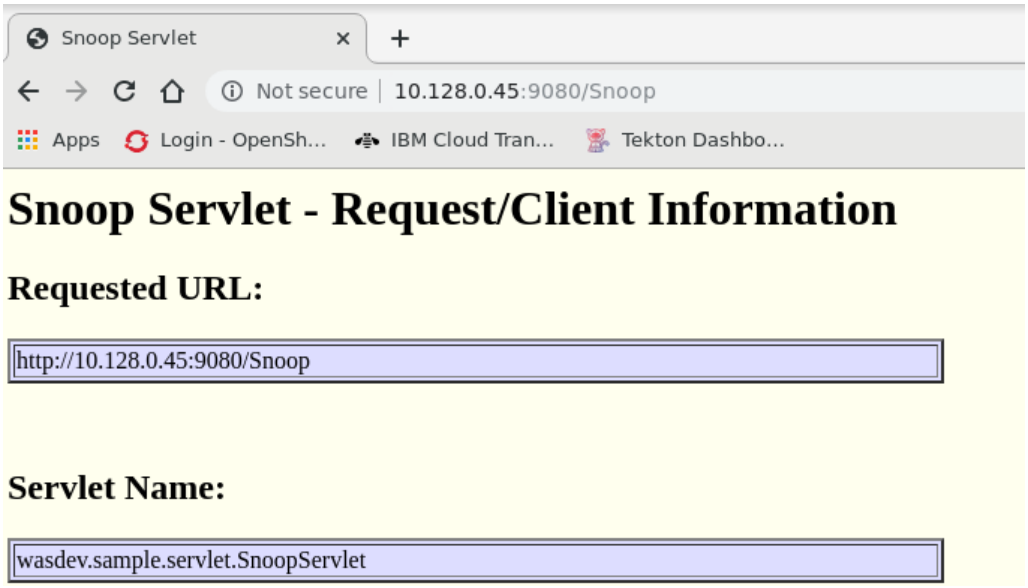
__6. Click `Chrome Web Browser` from bottom of the desktop.

__7. In the address bar, type: http://<yourIP>:9080/Simple Enter the IP address from the previous step.



	Note: If you try to use the host address such as http://192.168.142.130 it won't work, as there is no route defined between host and the container.

__8. In the address bar, type: http://<yourIP>:9080/Snoop

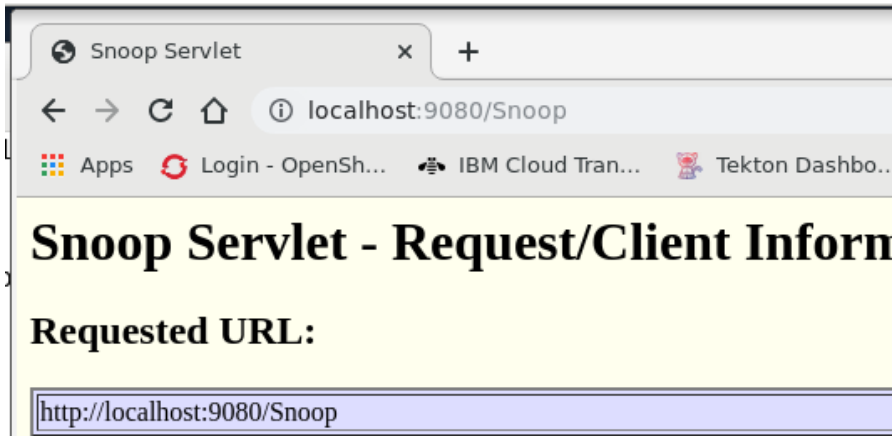


__9. Later, we will see how to expose a service so it can be accessed from the outside world. Often, you simply want to access a specific pod, even if it is not serving traffic on the internet. This can be done through port-forwarding support built into the Kubernetes API.

__a. Run command `kubectl port-forward simpleapp 9080:9080`

```
ibmdemo@icpa:~/student/lab3$ kubectl port-forward simpleapp 9080:9080

Forwarding from 127.0.0.1:9080 -> 9080
Forwarding from [::1]:9080 -> 9080 8
```

__10. From the browser, open page http://localhost:9080/Snoop and the page is displayed on the local host, if the port forward command is still running.



__11. Press CTRL-C (Hold Ctrl and C together) in the command line window to stop port forwarding.

3.7 Delete Pod

__1. Switch to the command window and run kubectl delete pod simpleapp

```
ibmdemo@icpa:~/student/lab3$ kubectl delete pod simpleapp

pod "simpleapp" deleted
```

__2. Run command to check kubectl get pods

```
ibmdemo@icpa:~/student/lab3$ kubectl get pods
No resources found
ibmdemo@icpa:~/student/lab3$
```

Notice that the pod is no longer displayed, and it was not restarted automatically.

We did not define higher level of abstraction that would have restarted the pod automatically such as DaemonSet or ReplicaSet which looks after a pod and restarts it automatically.

3.8 Create Replica Set

We will create a ReplicaSet to manage the number of pods automatically. Kubernetes Deployments provide the same replication functions as a ReplicaSet, as well as the ability to rollout changes and roll them back if necessary and are more typically used for production deployments.

__1. Review `kube02.yaml` which creates a replica set. Run command `cat kube02.yaml`

```
[ibmdemo@icp4a lab3]$ cat kube02.yaml

# A simple yaml file to create a replica set
apiVersion: extensions/v1beta1
kind: ReplicaSet
metadata:
  name: simpleapp
spec:
  replicas: 3
  selector:
    matchLabels:
      app: simpleapp
  template:
    metadata:
      name: simpleapp
      labels:
        app: simpleapp
    spec:
      containers:
      - name: simpleapp
        image: docker-registry.default.svc:5000/lab3/simpleapp:latest
```

Note that the `ReplicaSet` is covered in `apiVersion` of `extensions/v2beta1`. Also note that the Kubernetes documentation is available at `http://kubernetes.io`

We use `kind` as `ReplicaSet` and the `metadata` "`name: simpleapp`".

The metadata also needs to know how to select an application which is defined by the `matchLabels` selector `app: simpleapp`.

__2. For the replica set to start an application, it must have a `template` to start a container. The template is analogous to the way we created the simpleapp pod except that it also has labels set to `app: simpleapp`, which is used by the `ReplicaSet` to manage the container.

__3. Run `kubectl apply -f kube02.yaml` to create the replica set.

```
[ibmdemo@icp4a lab3]$ kubectl apply -f kube02.yaml

replicaset.extensions/simpleapp created
```

__4. Run `kubectl get pods -o wide`

```
[ibmdemo@icp4a lab3]$ kubectl get pods -o wide

NAME           READY    STATUS    RESTARTS   AGE      IP          NODE          NOMINATED NODE
simpleapp-mrnj5   1/1      Running   0          1m       10.128.0.46   icp4a.pot.com   <none>
simpleapp-s9q6p   1/1      Running   0          1m       10.128.0.47   icp4a.pot.com   <none>
simpleapp-tgmrt   1/1      Running   0          1m       10.128.0.48   icp4a.pot.com   <none>
```

Note that the application is running in **three** pods and that the **IP addresse**s and the **nodes** on which they are deployed by the Kubernetes scheduler. **The IP addresses and pod name suffixes will likely differ for you.**

__5. Now delete the first pod. Copy and paste the name of your first container (this is `simpleapp-mrnj5` in the output above) and run the following commands in succession to see the process of terminating, container creating and running the container again.

```
kubectl get pods

kubectl delete pod <your first pod name>

kubectl get pods
```

```
[ibmdemo@icp4a lab3]$ kubectl get pods
NAME              READY     STATUS    RESTARTS   AGE
simpleapp-mrnj5   1/1       Running   0          8m
simpleapp-s9q6p   1/1       Running   0          8m
simpleapp-tgmrt   1/1       Running   0          8m
[ibmdemo@icp4a lab3]$ kubectl delete pod simpleapp-mrnj5
pod "simpleapp-mrnj5" deleted
[ibmdemo@icp4a lab3]$ kubectl get pods
NAME              READY     STATUS    RESTARTS   AGE
simpleapp-qg6n4   1/1       Running   0          10s
simpleapp-s9q6p   1/1       Running   0          8m
simpleapp-tgmrt   1/1       Running   0          8m
```

Since we defined a **replica set** of **3**, Kubernetes keeps an eye on the number of running **pods** and if any pod stops or disappears, it will start another pod automatically. The scheduler balances the creation of pods in worker nodes based on the resource utilization of the workers.

In this case, a new pd was started to replace the pod you deleted. Take note that the new pod that was started has a different pod name and ip address than the one you deleted.

3.9 Scale the application up and down

__1. Run the command `kubectl get replicaset`

```
[ibmdemo@icp4a lab3]$ kubectl get replicaset

NAME      DESIRED   CURRENT   READY    AGE

simpleapp  3        3         3        20m
```

Note that the is Desired number of pods is **3**, Current and Ready are also **3**

__2. You can change the number of pods in the replicaset to **5** by running the following command `kubectl scale --replicas=5 replicaset/simpleapp`

Run the `kubectl get pods` command to see the new pods being created and started.You will see 5 pods now running.

```
[ibmdemo@icp4a lab3]$ kubectl get pods
NAME              READY     STATUS    RESTARTS   AGE
simpleapp-qg6n4   1/1       Running   0          18m
simpleapp-s9q6p   1/1       Running   0          26m
simpleapp-tgmrt   1/1       Running   0          26m
[ibmdemo@icp4a lab3]$ kubectl scale --replicas=5 rs/simpleapp
replicaset.extensions/simpleapp scaled
[ibmdemo@icp4a lab3]$ kubectl get pods
NAME              READY     STATUS              RESTARTS   AGE
simpleapp-9h4cq   0/1       ContainerCreating   0          2s
simpleapp-q84mv   0/1       ContainerCreating   0          2s
simpleapp-qg6n4   1/1       Running             0          18m
simpleapp-s9q6p   1/1       Running             0          26m
simpleapp-tgmrt   1/1       Running             0          26m
[ibmdemo@icp4a lab3]$ kubectl get pods
NAME              READY     STATUS    RESTARTS   AGE
simpleapp-9h4cq   1/1       Running   0          4s
simpleapp-q84mv   1/1       Running   0          4s
simpleapp-qg6n4   1/1       Running   0          18m
simpleapp-s9q6p   1/1       Running   0          26m
simpleapp-tgmrt   1/1       Running   0          26m
```

__3. Change the number of pods in the replicaset to **2** by running the following commands. As before, run the following commands in succession quickly to check the lifecycle of the pods

```
kubectl get pods

kubectl scale --replicas=2 rs/simpleapp          (rs is short for ReplicaSet)

kubectl get pods
```

__a. Notice that Kubernetes is terinating 3 of the 5 pods to match the desired state of 2 replicas. Eventually you will see only 2 pods running.

```
[ibmdemo@icp4a lab3]$ kubectl get pods

NAME              READY     STATUS    RESTARTS   AGE
simpleapp-9h4cq   1/1       Running   0          7m
simpleapp-q84mv   1/1       Running   0          7m
simpleapp-qg6n4   1/1       Running   0          25m
simpleapp-s9q6p   1/1       Running   0          34m
simpleapp-tgmrt   1/1       Running   0          34m
[ibmdemo@icp4a lab3]$ kubectl scale --replicas=2 rs/simpleapp
replicaset.extensions/simpleapp scaled
[ibmdemo@icp4a lab3]$ kubectl get pods
NAME              READY     STATUS        RESTARTS   AGE
simpleapp-9h4cq   0/1       Terminating   0          7m
simpleapp-q84mv   1/1       Terminating   0          7m
simpleapp-qg6n4   1/1       Terminating   0          25m
simpleapp-s9q6p   1/1       Running       0          34m
simpleapp-tgmrt   1/1       Running       0          34m
[ibmdemo@icp4a lab3]$ kubectl get pods
NAME              READY     STATUS    RESTARTS   AGE
simpleapp-s9q6p   1/1       Running   0          26m
simpleapp-tgmrt   1/1       Running   0          26m
```

3.10 Create Service to expose the application

Without a service to expose the application, this application is isolated in the IP address assigned to a pod on a host with no connection between the outside world and the application residing in the pod.

Next, create a service to route the traffic to the application running inside a pod.

The service should be able to find the pod where application is running. It does this by using labels.

__1. Review `kube03.yaml` to create the service. cat kube03.yaml

```
[ibmdemo@icp4a lab3]$ cat kube03.yaml

# A simple yaml file to create a service
apiVersion: v1
kind: Service
metadata:
  labels:
    app: simpleapp
  name: simpleapp
spec:
  selector:
    app: simpleapp
  ports:
  - port: 9080
  type: NodePort
```

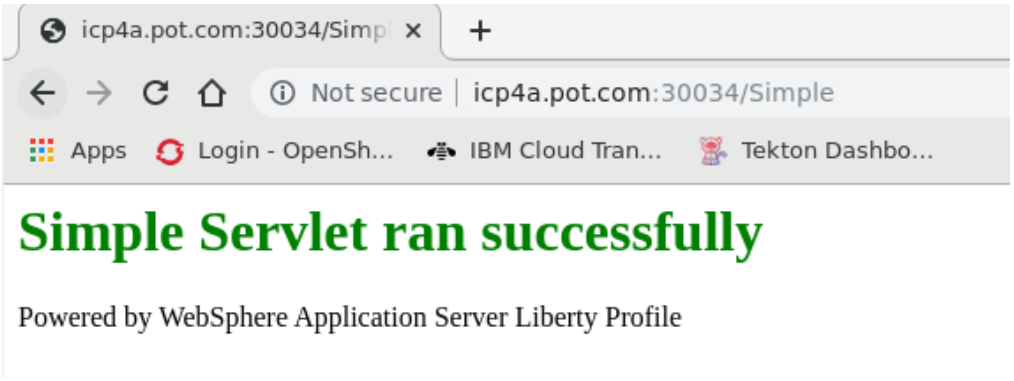
__2. Note that the service `simpleapp` is assigned through a label to `app: simpleapp` and port `9080` is automatically assigned a higher port on the host using `type: NodePort`.

__3. Create the service using `kubectl apply -f kube03.yaml` then check the service via `kubectl get svc` (svc is shorthand for service)

```
[ibmdemo@icp4a lab3]$ kubectl apply -f kube03.yaml

service/simpleapp created

[ibmdemo@icp4a lab3]$ kubectl get svc

NAME         TYPE       CLUSTER-IP     EXTERNAL-IP   PORT(S)          AGE
simpleapp    NodePort   172.30.3.179   <none>        9080:30034/TCP   9s
```

__4. Note the NodePort (`30034` in this case), as it will be different in your case.

__5. Switch to the browser and try the URL `http://icp4a.pot.com:<NodePort>/Simple` and substitute the Node Port with the number from the `kubectl get svc` command.



If we had an environment with multiple nodes, the proxy server would be running on all nodes, you could use any of the hosts to reach to the proper ghost pod.

	**Note:** The `service` routes the traffic from external world to the pod in the cluster through iptables.

Note that we used manifests to create a pod (no high availability), replica set (for number of replicas and high availability) and a service (for routing the traffic).

__6. Close the browser.

	**Note:** Question: If every node is acting like a proxy server, why do we need a dedicated proxy server? Answer: Normally, all compute nodes are shielded from the outside world. In such cases, the only way to reach compute nodes is through the designated proxy server.

3.11 Delete Service and Replica Set

__1. Run the following commands to delete the service and then delete the replica set. Note that the pods are deleted automatically.

```
kubectl get svc

kubectl delete svc simpleapp

kubectl get rs

kubectl delete rs simpleapp

kubectl get pods
```

```
[ibmdemo@icp4a lab3]$ kubectl get svc
NAME        TYPE       CLUSTER-IP     EXTERNAL-IP   PORT(S)          AGE
simpleapp   NodePort   172.30.3.179   <none>        9080:30034/TCP   9s

[ibmdemo@icp4a lab3]$ kubectl delete svc simpleapp
service "simpleapp" deleted

[ibmdemo@icp4a lab3]$ kubectl get rs
NAME        DESIRED   CURRENT   READY     AGE
simpleapp   2         2         2         1h

[ibmdemo@icp4a lab3]$ kubectl delete rs simpleapp
replicaset.extensions "simpleapp" deleted

[ibmdemo@icp4a lab3]$ kubectl get pods
No resources found.
```

End of Lab 03: Getting Started with Kubernetes

Appendix: SkyTap Tips for labs

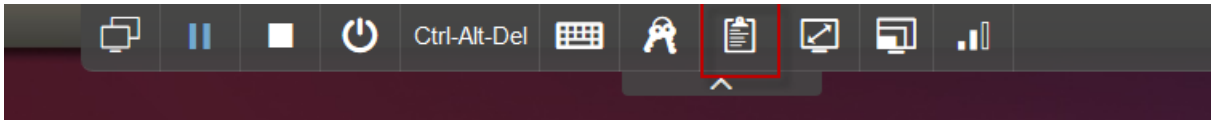
3.1 How to use Copy / Paste between local desktop and Skytap VM?

Using copy / Paste capabilities between the lab document (PDF) on your local workstation to the VM is a good approach to more efficiently work through a lab, while reducing the typing errors that often occur when manually entering data.

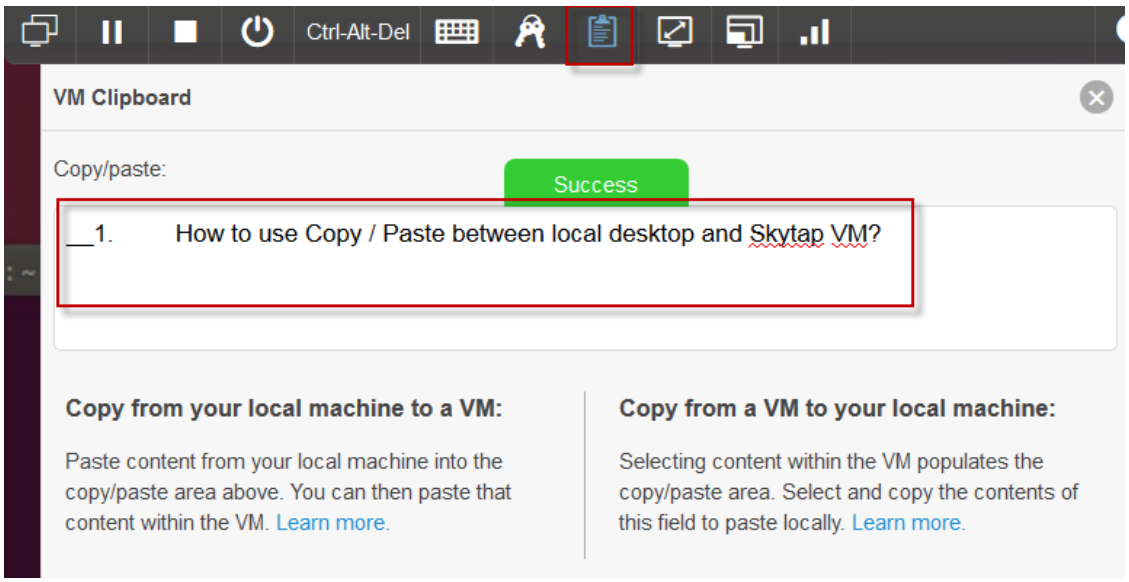
__1. In SkyTap, you will find that any text copied to the clipboard on your local workstation is not available to be pasted into the VM on SkyTap. So how can you easily accomplish this?

__a. First copy the text you intend to paste, from the lab document, to the clipboard on your local workstation, as you always have (CTRL-C)

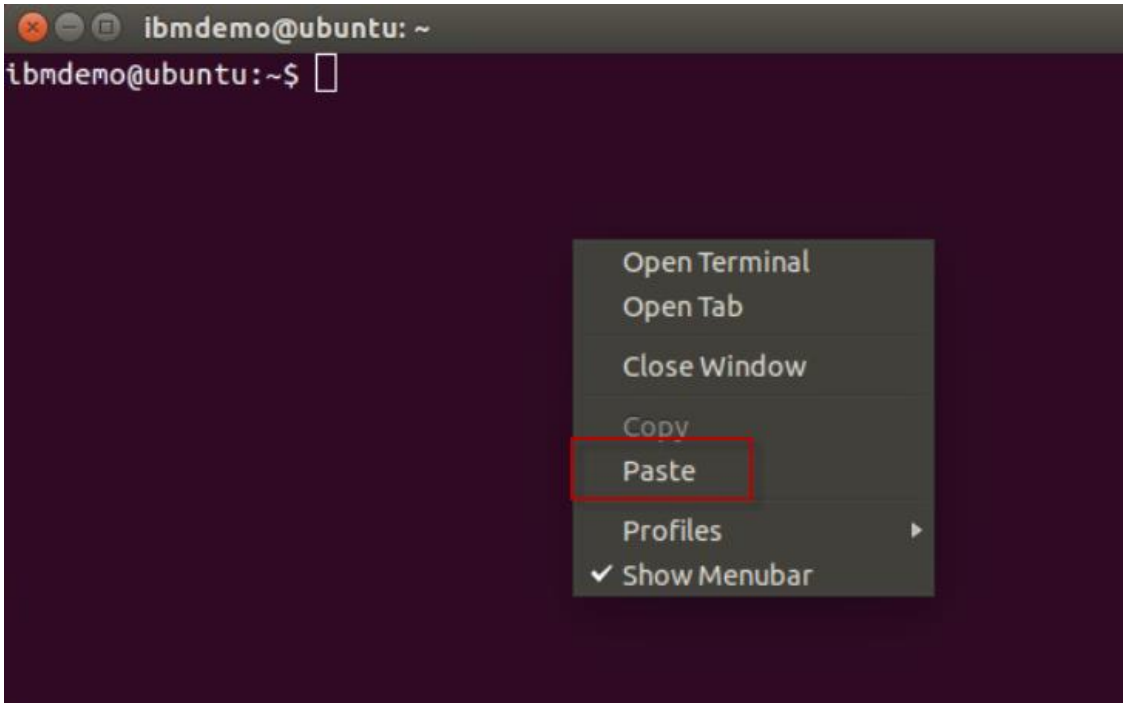
__b. Return to the SkyTap environment and click on the Clipboard at the top of the SkyTap session window.



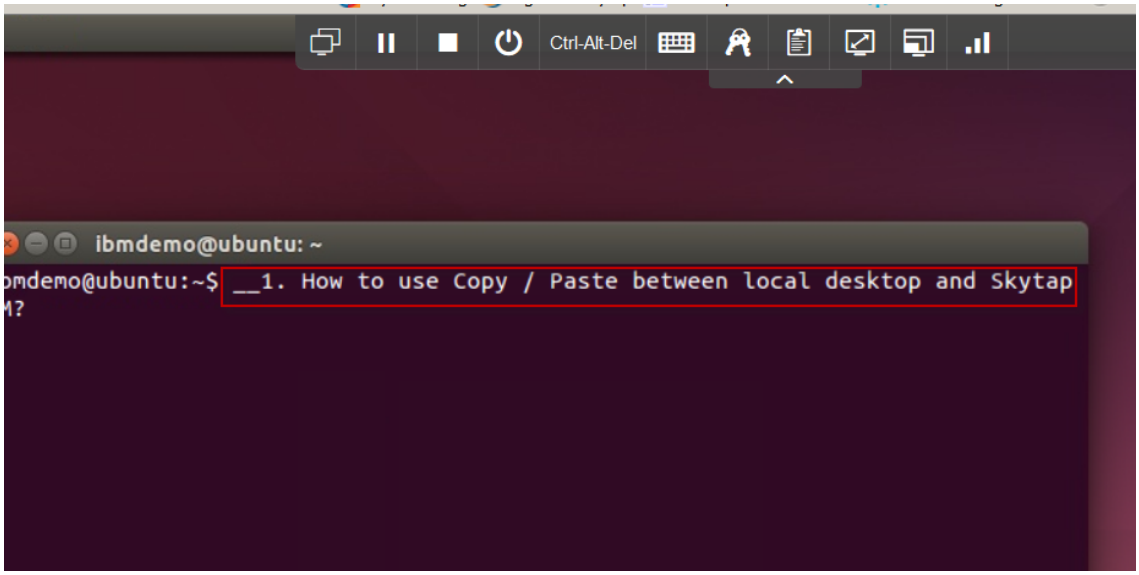
__c. Use **CTRL-V** to paste the content into the Copy/paste VM clipboard. Or use the **paste** menu item that is available in the dialog, when you right mouse click in the clipboard text area.



__d. Once the text is pasted, just navigate away to the VM window where you want to paste the content. Then, use **CTRL-C**, or right mouse click & us the **paste menu item** to paste the content.



__e. The text is pasted into the VM



Note: The very first time you do this, if the text does not paste, you may have to paste the contents into the Skytap clipboard twice. This is a known Skytap issue. It only happens on the 1st attempt to copy / paste into Skytap.