HOL-1289 : Introduction to Kubernetes

HOL-1289 : Introduction to Kubernetes

Author: Bryan Gartner, SUSE < Bryan.Gartner@SUSE.com >
Publication Date: 03/09/2020

List of Figures

Preface

Ramping up to the Fastlane: A Beginner's Guide to Learning and Using Kubernetes

Emerging as the defacto standard of orchestrating containerized, microservice workloads, Kubernetes is quickly becoming a fundamental training objective. In this hands-on session, you will learn how to effectively utilize the Kubernetes technology across the various Dev, Security, Data, and Ops perspectives to leverage the scalable, resilient, manageable environment for the workloads.

- In the first lab session, we will start with a single, yet complete containerized service deployment to give you the attraction of using Kubernetes. In essence, you will learn how to start and destroy the deployment with ease.
- In the next lab section, you will learn how to interact with several of these components to address scale, upgrades, and manage the running deployment through its lifecycle stages.
- In the final lab section you will discover many of the traditional IT pillar aspects of Kubernete to help organize, monitor and effectively operate your workload deployments.

Document Conventions

The following notices and typographical conventions are used in this documentation:

- */etc/passwd* : directory names and file names
- **<PLACEHOLDER>** : replace <PLACEHOLDER> with the actual value
- "PATH" : the environment variable PATH
- `ls`, `--help` : commands, options, and parameters
- *user* : users or groups

- *package name* : name of a package
- ***File › Save As*** : menu items, buttons
- FollowMe (https://www.google.com/)↗ : link to an external resource
- Preface : link to an internal resource in this document
- Commands that can be run by non-privileged users.

```
command
```

  Commands that must be run with root privileges. Often you can also prefix these commands with the sudo command to run them as non-privileged user.

```
sudo command
```

- In-line, self-study quiz questions

```
**QUIZ**
What is the Answer to the Ultimate Question of Life, the Universe, and Eve
```

- Notices:

Warning

Vital information you must be aware of before proceeding. Warns you about security issues, potential loss of data, damage to hardware, or physical hazards.

Important

Important information you should be aware of before proceeding.

Note

Additional information, for example about differences in software versions.

Tip

Helpful information, like a guideline or a piece of practical advice.

1 Start Your Engines

Deploy a complete, microservices-based application providing an end-to-end service to show the value of Kubernetes orchestration.

Goals / Objectives (Estimated time 10-15 minutes)

Easily launch, query, use and shutdown a complete set of containerized services — implementing a full Wordpress instantiation.

Assumptions

The example commands provided in this exercise use the full syntax and option names. In later exercises we will begin to show alternatives and other ways to launch, interact with and view what is running.

Environment

The following diagram shows the basic infrastructure components of the lab environment which you will interact with.

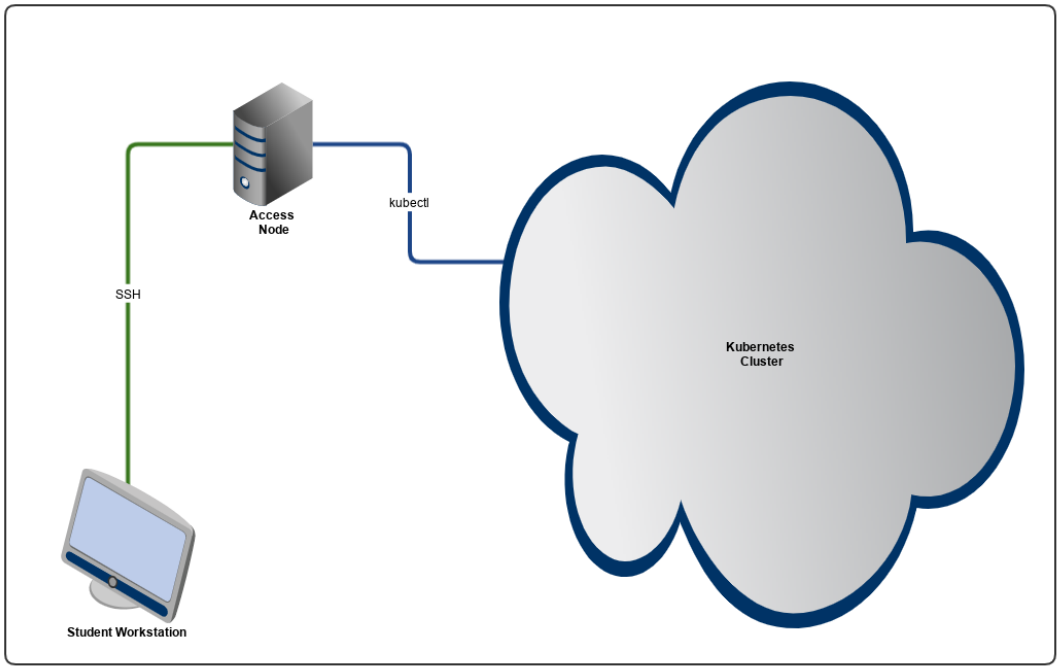

FIGURE 1.1: APPLICATION LAUNCH ENVIRONMENT

Process

Login to your student workstation (**<user>** : *tux* , **<password>** : *linux*)

- From the graphical user interface application dock at the bottom of your screen, launch a Terminal

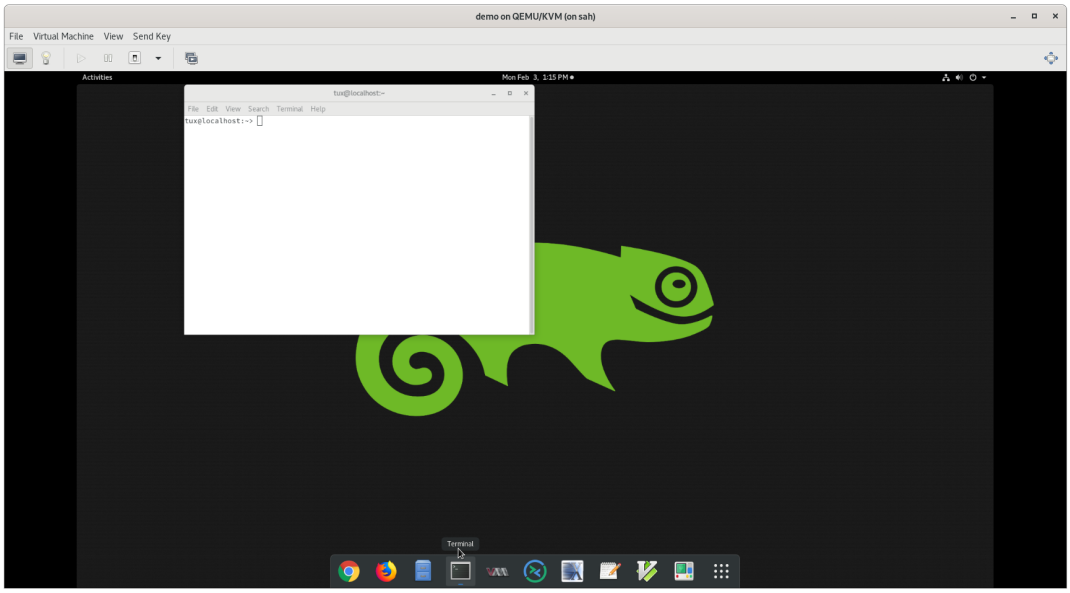

FIGURE 1.2: ENVIRONMENT TERMINAL LAUNCH

- In the terminal window, login to the lab environment's "Access Node" using secure shell (`ssh`) by typing

```
ssh sles@10.110.1.10
```

using the **<password>** *linux*

> **Note**
>
> All of the following process steps should be completed while logged into this lab environment's "Access Node" as mentioned above!

- Exercise - Launch a complete application set

1. From the "Access Node" command prompt, launch the application deployment by entering the following command

```
kubectl apply -k ./Lab1 (1)
```

① Overiew of kubectl (https://kubernetes.io/docs/reference/kubectl/overview/)↗

> **QUIZ**
> How many lines of output resulted from this command?

2. Let's do a quick discovery of each of the component types within your application deployment that you just launched

> **Tip**
>
> As a reference for what you just deployed, refer to Deploying WordPress and MySQL with Persistent Volumes (https://kubernetes.io/docs/tutorials/stateful-application/mysql-wordpress-persistent-volume/)↗.

```
kubectl get secrets  ①
kubectl get services  ②
kubectl get deployments  ③
kubectl get persistentvolumeclaims  ④
kubectl get persistentvolumes
kubectl get pods  ⑤
```

1. Kubernetes Secrets (https://kubernetes.io/docs/concepts/configuration/secret/)↗
2. Kubernetes Services (https://kubernetes.io/docs/concepts/services-networking/service/)↗
3. Kubernetes Deployments (https://kubernetes.io/docs/concepts/workloads/controllers/deployment/)↗
4. Kubernetes Volumes (https://kubernetes.io/docs/concepts/storage/persistent-volumes/)↗
5. Kubernetes Pods (https://kubernetes.io/docs/concepts/workloads/pods/pod/)↗

> **QUIZ**
> a) How many "services" are running (besides kubernetes)?
> b) How many "pods" are running ?

- At this point you should begin to realize that you have deployed several components, with a single command, across most of the traditional IT pillars:
  - Network - a front-end loadbalancer
  - Compute - a web application (WordPress (https://wordpress.org/)↗) and a backing data-base (MySQL (https://dev.mysql.com/)↗)
  - Storage - a backing volume store for each of your compute components

> **Note**
>
> These components are summarized in the figure below:

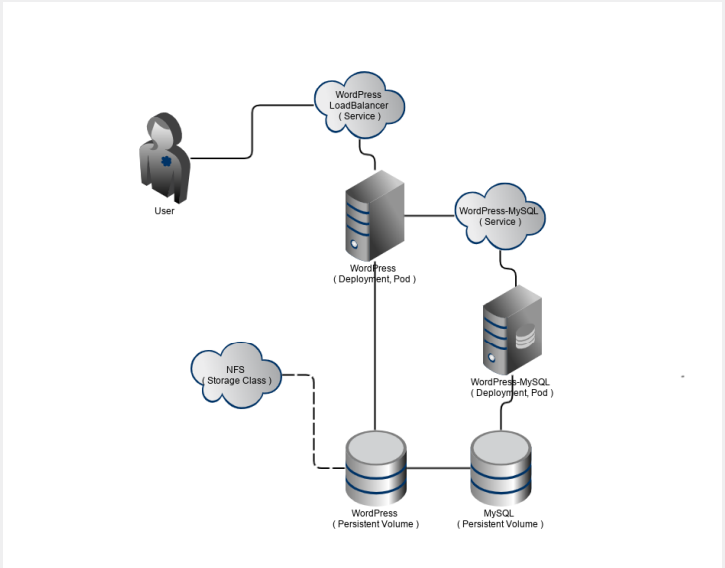

FIGURE 1.3: APPLICATION DEPLOYMENT

3. Next, let's ensure that the necessary components are available and running

```
FixMe
```

Then you can configure access to the deployment from the "Access Node" with a simple networking port forward command (and leave it running in the background)

```
kubectl port-forward --address 0.0.0.0 service/wordpress 8080:
Ctrl-z
bg
```

And you can now access and interact with your deployed service

```
w3m http://127.0.0.1:8080/
```

Which should land you on the WordPress setup page. Feel free to walk through any number of steps.

> **Note**
>
> You should also be able to access the deployed service, from a web-browser on your "Student Workstation" with the URL http://10.110.1.10:8080 (http://10.110.1.10:8080)↗

4. Now you can now quit "q" the text-based `w3m` browser, shutdown the port forward, and delete/stop all of these deployed components

```
fg
Ctrl-c
kubectl delete -k ./Lab1
```

Then validate that the main application deployment components are gone (beyond the core Kubernetes service)

```
kubectl get services
```

> **Important**
>
> ***Congratulations !*** Now you can see the initial appeal of what Kubernetes offers, by simply leveraging the predominant command-line interface (CLI) tool, `kubectl` (which interfaces with the Kubernetes cluster manager) to do what you want to do in a very easy fashion.

- ***Optional / Advanced Exercises*** - If you have some spare time, or are curious about the commands used or what was deployed, try:
  - Explore the available functionality of the `kubectl` command, many of which you will be using in later exercises, for almost all the relevant interactions with Kubernetes

```
kubectl help | more
```

  - Review the manifest content and format used to descriptively manage the resources you need deployed in Kubernetes

```
more ./Lab1/*.yaml
```

- Learn more about the `-k` argument to leveraging Kubernetes kustomize (https://kustomize.io/)↗

Knowledge Check

Initial introduction to `kubectl` and leveraging it to

- deploy multiple microservices with a single command
- get the status of those microservices, using a variation of that single command
- destroy the running set of microservices, again using a variation of that single command

2 Turn Signal On, Ready to Merge

Understand the basics of how to manually interact with Kubernetes and adjustement to an application deployment over its lifecycle.

Goals / Objectives (Estimated time 25 - 35 minutes)

Learn how to manually inspect, launch, scale and update a Kubernetes Deployment object. See Kubernetes Basics (https://kubernetes.io/docs/tutorials/kubernetes-basics/)↗ for more examples/details.

Warning

To reduce confusion, the assumption is that the respective Kubernetes cluster being used does not have any extraneous, non-core microservices running. So please cleanup previous or other lab exercise attempts before proceeding.

Assumptions

The example commands provided in this exercise still use the full syntax and option names. However, alternative ways with abbreviated syntax options are listed below the main command line entries, if you'd like to try those.

Environment

The following diagram shows the infrastructure components of the lab environment which you will interact with.

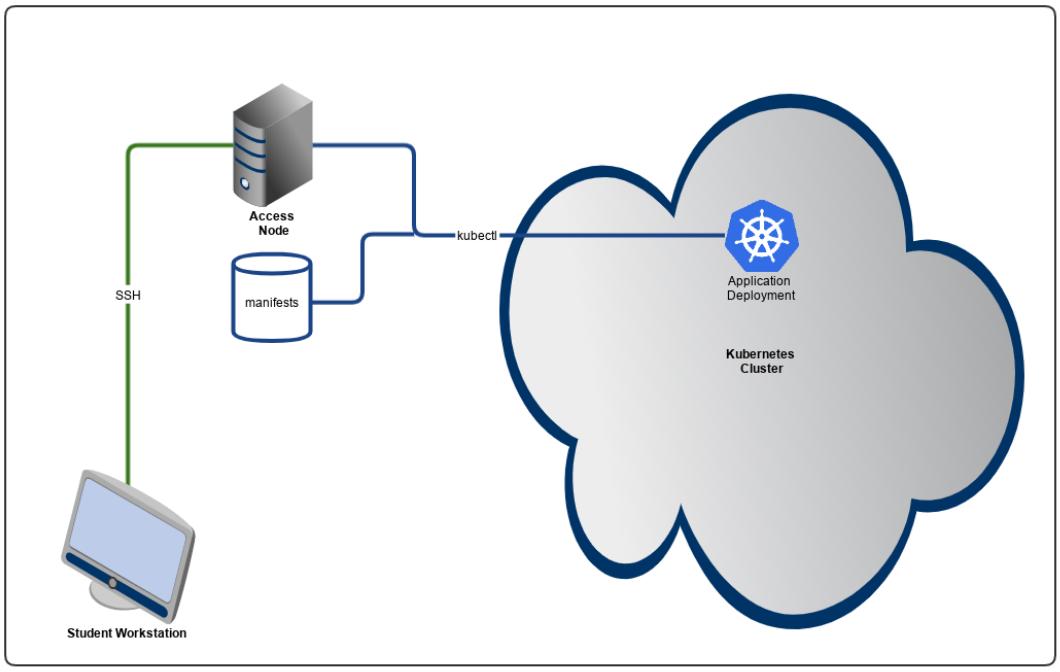

FIGURE 2.1: APPLICATION LAUNCH ENVIRONMENT

Process

Login to your student workstation (**<user>** : *tux* , **<password>** : *linux*)

- From the graphical user interface application dock at the bottom of your screen, launch a Terminal.
- In the terminal window, login to the lab environment's "Access Node" using secure shell (`ssh`) by typing

```
ssh sles@10.110.1.10
```

using the **<password>** *linux*

> **Note**
>
> All of the following process steps should be completed while logged into this lab environment's "Access Node" as mentioned above!

- Exercise - Preview and launch the application deployment manifest (https://github.com/bwgartner/suse-doc/raw/master/SUSECon/2020/HOL-1289/./Lab2/deployment.yaml)↗

  1. From the shell prompt, view manifest for the application that is going to be deployed

     ```
     cat ./Lab2/deployment.yaml
     ```

     QUIZ
     a) What is the name of the deployment?
     b) How many replica sets will be running?
     c) Which container image:version will be used for the pods?

  2. Create the application deployment from the designated manifest

     ```
     kubectl create --filename=./Lab2/deployment.yaml
     ```

     > **Tip**
     >
     > An abbreviated, general version of this command would be `kubectl create -f <manifestFilename>`

  3. Check your answers to the **QUIZ** above, for your active deployment

     ```
     kubectl get deployments nginx-deployment (1)
     kubectl get replicasets (2)
     kubectl get pods (3)
     ```

1. Kubernetes Deployments (https://kubernetes.io/docs/concepts/workloads/controllers/deployment/)↗
2. Kubernetes ReplicaSet (https://kubernetes.io/docs/concepts/workloads/controllers/replicaset/)↗. For now you can ignore the "nfs-client-provisioner" one.
3. Kubernetes Pods (https://kubernetes.io/docs/concepts/workloads/pods/pod/)↗. For now you can ignore the "nfs-client-provisioner" one.

> **Tip**
>
> An abbreviated version of the *replicasets* commands would be `kubectl get rs`

At this point, you can beging to mentally map the previous command outputs to a graphical representation, shown below, of the specific deployment.

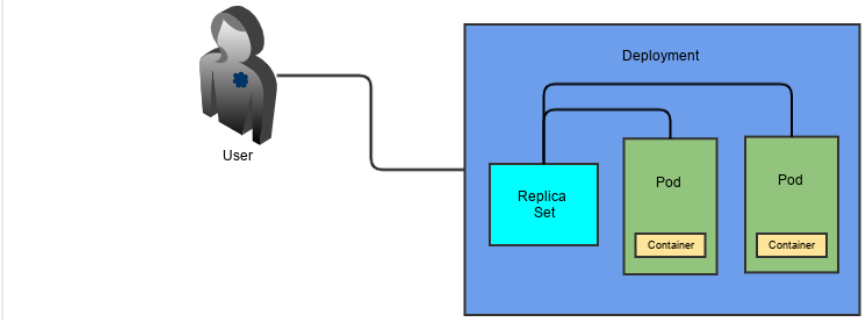

FIGURE 2.2: APPLICATION DEPLOYMENT

4. Continuing with the previous application deployment, obtain a more detailed view of each of the components

```
kubectl describe deployments nginx-deployment
```

QUIZ
a) Are all the desired "Replicas" created and running?
b) What events have happened during the deployment?

- Exercise - Show how Kubernetes maintains the specified number of replica set pods to provide the desired resiliency functionality

  1. Get a view of the currently running pods that are part of the replica set

     ```
     kubectl get pods
     ```

  2. Let's target the last "nginx" one on the previous commands output. Cut and past the entry from the "NAME" first column, and substituting that string for **<nginx-deployment-ID>** in the command below

     ```
     kubectl delete pod <nginx-deployment-ID>
     ```

     The resulting output should convey that the pod was deleted. . Now verify that Kubernetes dealt with that issue, ensure that two pods are always running for your specified replica set

     ```
     kubectl get pods
     kubectl get replicasets
     kubectl describe replicasets
     ```

     As you should see, a new pod **<ID>** has been created, and recorded as an event in the replicaset, to keep the desired count always running (and note that the one of the original pair that you did not delete is still running).

- Exercise - Manually scale your application deployment to perhaps respond to an increased amount of usage

  1. View changes in proposed deployment (https://github.com/bwgartner/suse-doc/raw/master/SUSECon/2020/HOL-1289/./Lab2/deployment-scale.yaml)↗ manifest, compared to the original one

     ```
     diff ./Lab2/deployment.yaml ./Lab2/deployment-scale.yaml
     ```

QUIZ
a) Which attribute is being modified?

2. Check the current state

```
kubectl get replicasets
kubectl get pods
```

3. Scale the application deployment's replicasets by applying the updated manifest

```
kubectl apply --filename=./Lab2/deployment-scale.yaml
```

4. Confirm the expected results

```
kubectl get replicasets
kubectl get pods
```

Bonus QUIZ
a) How might you scale down the deployment back to the original cc

- Exercise - Update your application using a new container image version, as if you need to take advantage of security updates or improved functionality

1. View changes in proposed deployment (https://github.com/bwgartner/suse-doc/raw/master/SUSECon/2020/HOL-1289/./Lab2/deployment-update.yaml)↗ manifest, compared to the previous, scaling one

```
diff ./Lab2/deployment-scale.yaml ./Lab2/deployment-update.yam
```

QUIZ
a) Which attributes are different from the original deployment?
b) Given the previous exercise, which single change do you expect t

2. Check the current state

```
kubectl describe deployments nginx-deployment
```

> **Note**
>
> In `kubectl get deployments` output, you will notice that the previous scaling events has been recorded and logged.

3. Update the application container image with a newer version

```
kubectl apply --filename=./Lab2/deployment-update.yaml
```

4. Confirm the expected results

```
kubectl describe deployments
kubectl describe replicasets
kubectl describe pods
```

Bonus QUIZ
a) How could you downgrade the container image version back dow

> **Note**
>
> One can also accomplish such updates with `kubectl set` or with `kubectl edit`. For those approaches, refer to Kubernetes-Workloads-Controllers-Deployments (https://kubernetes.io/docs/concepts/workloads/controllers/deployment/#updating-a-deployment)↗ documentation.

- ***Optional / Advanced Exercises*** - If you have some spare time, or are curious about the commands used or what was deployed, try:
  - While `kubectl` is a comprehensive command-line interface to interact with Kubernetes, a recent terminal-based user interface, K9s (https://github.com/derailed/k9s)↗ has also become available. Its goal is to make it easier to navigate, observe and manage your applications. If you'd like to take this for a test drive, use the following steps, in the terminal, while logged into the "Access Node":
    1. Install this package

```
sudo zypper install ./Lab2/k9s-0.11.2-bp151.4.1.x86_64.
```

using the **<password>** *linux*

> **Note**
>
> You will need to answer "y" to the *Continue* question and "i" to the *Signature verification* question.

2. Then beginning learning about the K9s command and launch it

```
k9s help
k9s info
k9s
```

You should see an interface like that shown in the following figure

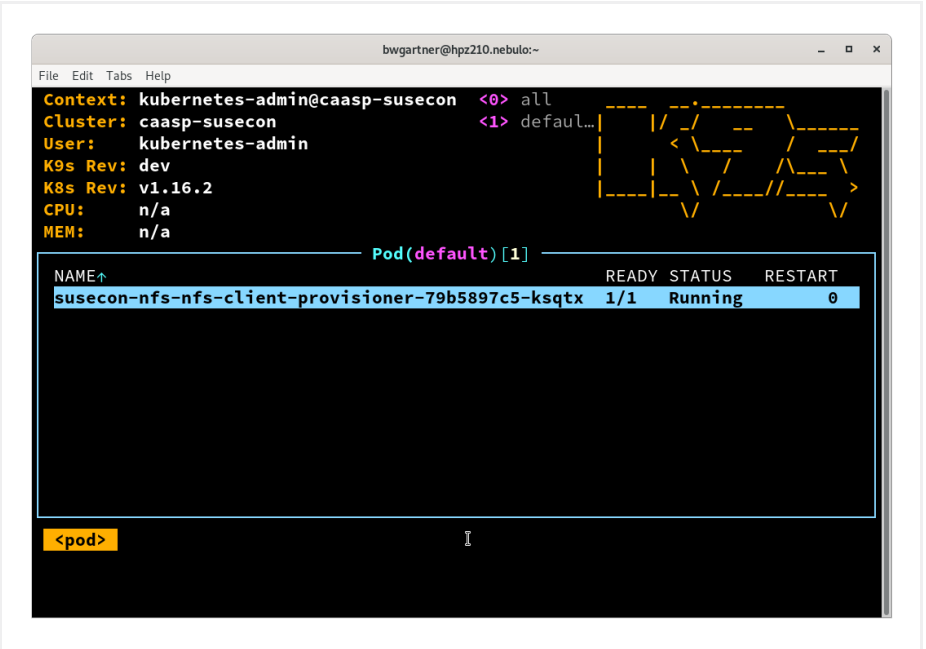

FIGURE 2.3: KUBERNETS CLI TO MANAGE YOUR CLUSTERS IN STYLE

3. At this point you can explore using the interface by looking over the K9s (https://github.com/derailed/k9s)↗ project GitHub site to determine which keystroke leads you to the desired output.

> **Tip**
>
> If you would like to continue running this for the remainder of this hands-on session, simply open another Terminal and log into the "Access Node" to complete those exercises.

- Cleanup

> **Important**
>
> Ensure that you clean up all your application deployments to return to a known, clean state. Delete the running deployment and check that all the other objects are gone as well.

```
kubectl delete --filename=./Lab2/deployment-update.yaml
kubectl get deployments
kubectl get replicasets
kubectl get pods
```

Knowledge Check

Initial introduction to the Kubernetes Deployment object

- launching and exploring the deployment components
- scaling and updating selected deployment components
- deleting the deployment

3 Cruising Down the Highway

- Become aware of more Kubernetes aspects

Goals / Objectives (Estimated time 40 - 50 minutes)

Drill down through more layers of Kubernetes functionality to understand sharing of resources, operational interfaces and aspects of workloads, and what the underlying cluster contains and provides.

> **Warning**
>
> Again, to reduce confusion, the assumption is that the respective Kubernetes cluster being used does not have any extraneous, non-core microservices running. So please cleanup previous or other lab exercise attempts before proceeding.

Assumptions

The example commands provided in this exercise still use the full syntax and option names. However, alternative ways with abbreviated syntax options are listed below the main command line entries, if you'd like to try those.

Environment

Coverage of the respective environment you are interacting with will be provided towards the end of this lab section.

Process

Login to your student workstation (**<user>** : *tux* , **<password>** : *linux*)

- From the graphical user interface application dock at the bottom of your screen, launch a Terminal.
- In the terminal window, login to the lab environment's "Access Node" using secure shell (`ssh`) by typing

```
ssh sles@10.110.1.10
```

using the **<password>** *linux*

> **Note**
>
> Most all of the following process steps should be completed while logged into this lab environment's "Access Node" as mentioned above! Exceptions will be noted and called out.

- Exercise - A simplistic way to provide "virtual cluster spaces" to divide cluster resources via Kubernetes Namespaces (https://kubernetes.io/docs/concepts/overview/working-with-objects/namespaces/)↗

1. From the shell prompt, view the current state of namespaces on your target cluster

```
kubectl get namespace
```

QUIZ
a) In all the previous exercises, what namespace did your workloads lan
b) Any guesses on what might exist in the "kube-system" namespace?

2. View the manifest for the application that is going to be deployed

```
cat ./Lab3/deployment.yaml
```

QUIZ
a) What namespace is being targeted for this deployment?

3. Create a new namespace and launch the application deployment to specifically reside there

```
kubectl create namespace hol1289
kubectl apply --filename=./Lab3/deployment.yaml --namespace=hol128
```

> **Tip**
>
> An abbreviated, general version of the latter command would be `kubectl create -f <manifestFilename> -n<nameSpace>`

```
kubectl get namespaces (1)
kubectl get deployment --namespace=hol1289 (2)
```

```
kubectl get replicasets --namespace=hol1289 (3)
kubectl get pods --namespace=hol1289 (4)
```

(1) Kubernetes Namespaces (https://kubernetes.io/docs/concepts/overview/working-with-objects/namespaces/)↗
(2) Kubernetes Deployments (https://kubernetes.io/docs/concepts/workloads/controllers/deployment/)↗
(3) Kubernetes ReplicaSet (https://kubernetes.io/docs/concepts/workloads/controllers/replicaset/)↗.
(4) Kubernetes Pods (https://kubernetes.io/docs/concepts/workloads/pods/pod/)↗

4. And just to validate your new deployment is only part of this new, parallel world (aka namespace you just created), "hol1289", try to look for the same deployments without the namespace designation (which implies just looking in the "default" namespace"

Note

Expecting a return from the command below of "Error from server (NotFound): deployments.apps "nginx-deployment" not found", assuming you had cleaned up your previous deployments from the last lab.

```
kubectl get deployment nginx-deployment
```

Tip

In case you weren't aware, for all of the exercises in this hands-on lab, you have been given the "admin" privileges for the Kubernetes cluster. Kubernetes provides a very granular set of role-based access control (https://kubernetes.io/docs/reference/access-authn-authz/rbac/)↗ (RBAC) methods to permit who can do what and with how much (quota extent). A namespace is just one of typical attribute resources given to individuals or groups to confine their view of the world and potential impact to other users and workloads.

Next, cleanup this deployment and namespace, then validate they are gone

```
kubectl delete --filename=./Lab3/deployment.yaml --namespace=hol12
kubectl get deployment --namespace=hol1289
kubectl delete namespace hol1289
kubectl get namespaces
```

- Exercise - Leverage storage volumes for sharing data across containerized workloads

> **Note**
>
> While some of the initial attraction and focus for containerized workloads was their stateless, ephemeral approach, many applications want to start from a known state and rely upon existing data or even populate more content for other workloads. Kubernetes Container Storage Interface (https://kubernetes-csi.github.io/docs/drivers.html)↗ (CSI) provides many options to provide such Volumes (https://kubernetes.io/docs/concepts/storage/persistent-volumes/)↗.

1. View the Kubernetes Storage Class (https://kubernetes.io/docs/concepts/storage/storage-classes/)↗ configured on this implementation

```
kubectl get storageclasses
```

QUIZ
a) Based on the naming, what provisioner protocol do you expect provic

> **Tip**
>
> This "Storage Class" was pre-deployed for you, via a Helm (https://helm.sh/)↗ chart (https://github.com/helm/charts/tree/master/stable/nfs-client-provisioner)↗ (which can be explored in some of the "Optional / Advanced" exercises later. In this very self-contained Kubernetes instance, you can actaully find the directory on the "Access Node" at */public* where the storage location is offered from and any resulting data resides.

2. Launch a set of components and workloads that leverage this backing store and each other's actions

```
kubectl apply -k ./Lab3/volume
```

3. Verify each of the component types, starting with Kubernetes Persistent Volumes (https://kubernetes.io/docs/concepts/storage/persistent-volumes/)↗

```
kubectl get persistentvolumeclaims
```

> **Tip**
>
> An abbreviated version of this command would be `kubectl get pvc`

Plus check to see that a new directory "default-*" has appeared in the underlying filesystem directory on the "Access Node" providing the storage class

```
ls /public
```

4. Now find the pods "nfs-busybox-*" which are utilizing the volumes and responsible for writing data entries there

```
kubectl get pods
```

You can either review the data writing function by looking at the manifest

```
cat  ./Lab3/volume/nfs-busybox*
```

or by querying the running pod directly (pick either of the launched pod "NAME" from the previous `kubectl` command, substituting the **<nfs-busybox-ID>** in the `kubectl` command below

```
kubectl describe <nfs-busybox-ID>
```

QUIZ
a) Which "Command:" is being run in this pod?
a) What "Mounts:" point inside the pod is associated with the persistent

Of course, another application could have been launched to utilize the ever-changing content, yet a simple way to watch the changes (given the mostly self-contained instance used in this lab) is

```
watch cat /public/default*/index.html
```

After you see some updates (a timestamp change and possibly even the pod name responsible for writing the data change), you can exit the `watch` with *Ctrl-c*. You can also enter the running pod (briefly) to see the actual mount (again substituting the respective pod **<nfs-busybox-ID>**

```
kubectl exec -it <nfs-busybox-ID> /bin/df
```

Tip

In general, for applications running in a Kubernetes pod that support some userspace utilities (including this particular pod), you may be able to enter and interact from the internal shell by calling `kubectl exec -it <podID> /bin/sh` (or "/bin/bash")

5. From an operational standpoint, often it is valuable to view the logs from a given pod (beyond the "Events:" returned by `kubectl describe`). Upon review of the `kubectl get pods` command below, substitute the respective name of the NFS storage class pod for **<nfs-client-ID>** in the `kubectl logs <nfs-client>` command

```
kubectl get pods
kubectl logs <nfs-client-ID> | more
```

6. Finally, cleanup these components from this exercise

```
kubectl delete -k ./Lab3/volume
```

- Exercise - Explore the underlying Kubernetes infrastructure that you have been utilizing throughout this hands-on lab session

1. Start with simple listings of the APIs, the access configuration and then nodes plus roles in this Kubernetes instance

```
kubectl cluster-info
kubectl config view
kubectl get nodes
```

QUIZ
a) How many "nodes" are in this Kubernetes cluster instance?
b) Are all the "nodes" in a "Ready" STATUS?
c) What version of Kubernetes is being used?

Tip

A more comprehensive listing can be obtained via
`kubectl get nodes -o wide`

2. Let's find more details about the node providing the Kubernetes "ROLE" of "master"

```
kubectl describe node caasp-master-0
```

QUIZ
a) What "Kernel Version:" does this node have?
b) What "Container Runtime Version:" does this node utilize?
c) How many "Non-terminated Pods:" are running on this node? And in v

3. And review the details for one of the nodes providing the Kubernetes "ROLE" of "worker"

```
kubectl describe node caasp-worker-0
```

QUIZ
a) From a "Capacity:" perspective
- how many "cpu:" cores are available?
- how much "ephemeral-storage:"?
- how much "memory:"?
b) From an "Allocated resources:" standpoint, the current "Requests" fc
- "cpu" and what are the units of measure
- "memory"
- "ephemeral storage"?
c) Are the same number/type of "Non-terminated Pods:" running on this

4. The following diagram shows a general overview of Kubernetes infra-structure components (refer to Kubernetes Components (https://kubernetes.io/docs/concepts/overview/components/)↗)

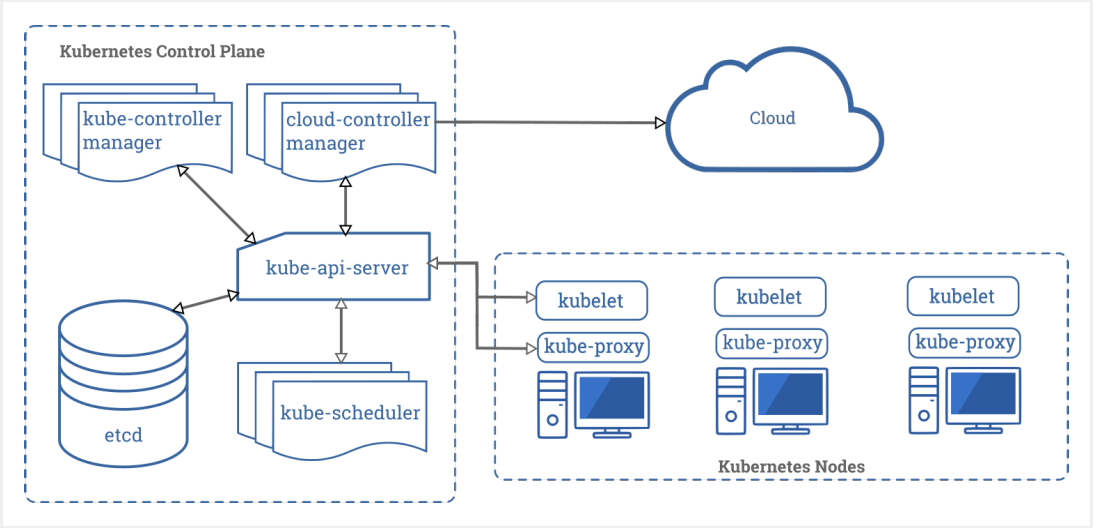

FIGURE 3.1: KUBERNETES COMPONENTS (FIXME)

Throughout this hand-on lab session, you have interacted with many of these components, as denoted in the following diagram

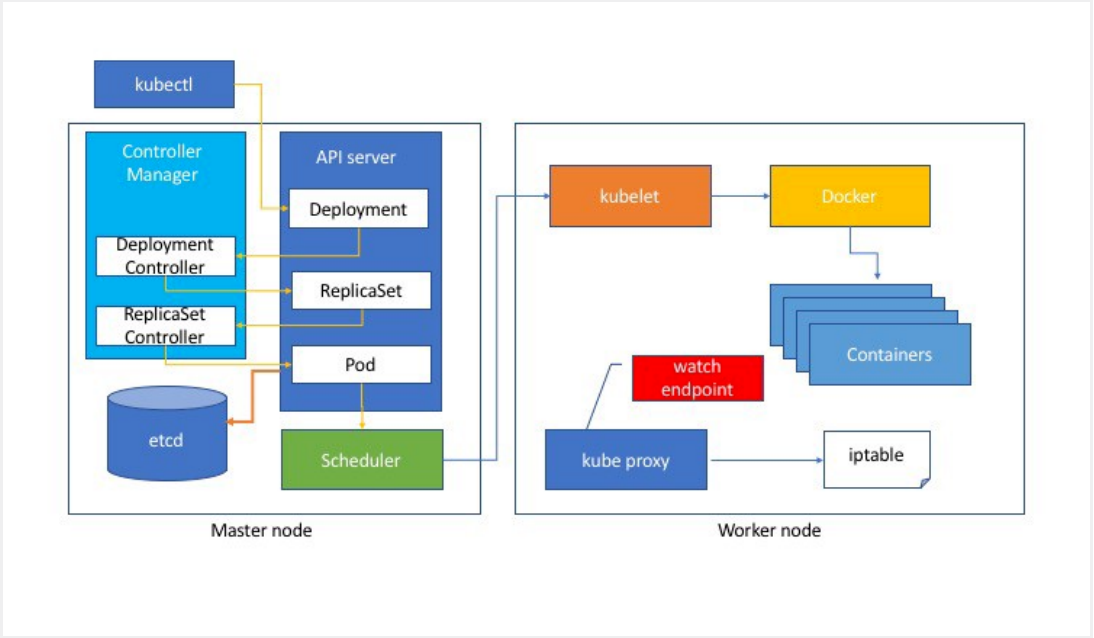

FIGURE 3.2: KUBERNETES ARCHITECTURE (FIXME)

And from your "Student Workstation" graphical user interface dock, if you launch"Virtual Machine Manager" you will see that all of the needed Kubernetes "nodes" providing the "ROLES" are running as virtual ma-chines

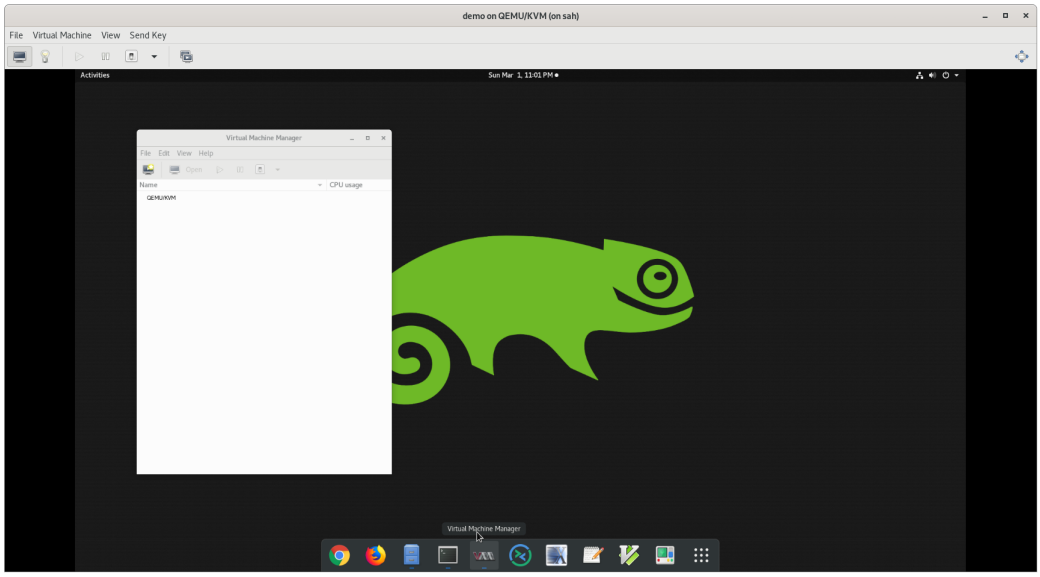

FIGURE 3.3: KUBERNETES LAB VIRTUAL MACHINES (FIXME)

Which yields the following environment you have been exercising in this session

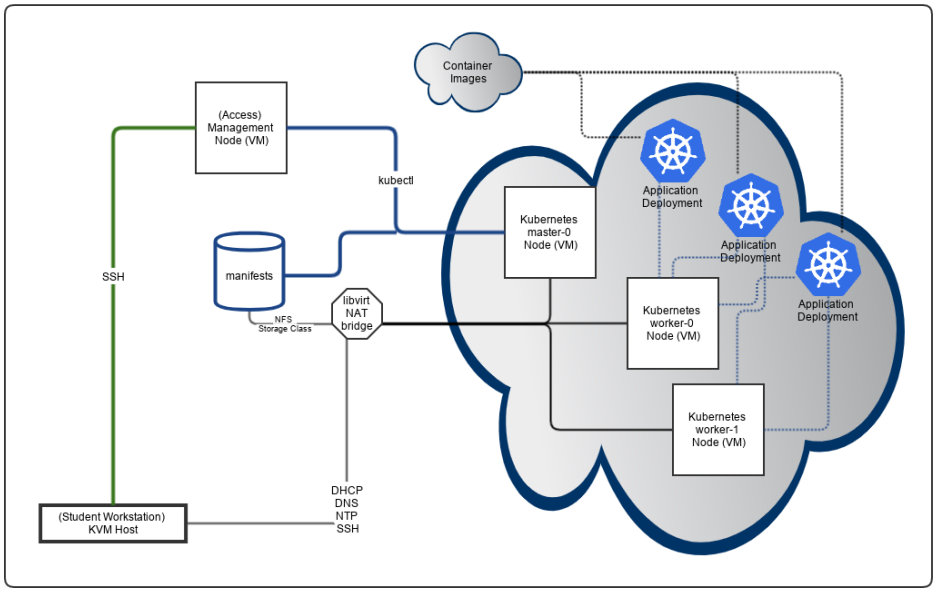

FIGURE 3.4: KUBERNETES LAB ENVIRONMENT INSTANCE (FIXME)

- ***Optional / Advanced Exercises*** - If you have some spare time, or are curious about the commands used or what was deployed, try:
    - FixMe
        - using Helm
        - using Stratos to view multi-cluster
        - reviewing how to deploy a K8s cluster

        Knowledge Check
        Initial introduction to the Kubernetes Deployment object
- FixMe

4 Summary

As you have likely noticed, this hands-on lab provided exercises

- from a top-down perspective, starting with a complete microservice-based application deployment
- then drilling down through managing and adjusting the deployment through life-cycle attributes
- and digging even deeper to take operational advantage of more functional components, including uncovering the underlying resources involved

At this point you should now feel like you have enough experience with Kubernetes to share with your peers, begin using it on-premise, in managed services or in public cloud providers.

Important

help us cleanup the lab environment before you leave

5 Homework

If this enticed you to further investigate this technology, feel free to: * take this lab guide (and even grab the source (https://github.com/bwgartner/suse-doc/tree/master/SUSECon/2020/HOL-1289)↗) including all of the referenced file content * download (https://www.suse.com/download-linux/)↗ the SUSE Linux Enterprise Server (https://www.suse.com/products/server/)↗ operating system and SUSE CaaS Platform (https://www.suse.com/products/caas-platform/)↗ and the associated product documentation (https://documentation.suse.com/)↗ * the technology to deploy (https://github.com/alexarnoldy/new_SUSECon)↗ such a Kubernetes cluster on your own Linux KVM host