Technical Workshop - SUSE CaaS Platform

-Workbook-

Course ID: Technical Workshop - SUSE Container as a Platform
Version: 3.0.0
Date: 2018-7-11

Proprietary Statement

SUSE LLC, has intellectual property rights relating to technology embodied in the product that is described in this document.

SUSE
Maxfeldstrasse 5
90409 Nuremberg
Germany
www.suse.com

If you know of illegal copying of software, contact your local Software Antipiracy Hotline.

Disclaimer

SUSE LLC, makes no representations or warranties with respect to the contents or use of this documentation, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose.

Further, SUSE LLC, reserves the right to revise this publication and to make changes to its content, at any time, without obligation to notify any person or entity of such revisions or changes. Further, SUSE LLC, makes no representations or warranties with respect to any software, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. Further, SUSE LLC, reserves the right to make changes to any and all parts of SUSE software, at any time, without any obligation to notify any person or entity of such changes.

Any products or technical information provided under this Agreement may be subject to U.S. export controls and the trade laws of other countries. You agree to comply with all export control regulations and to obtain any required licenses or classification to export, re-export or import deliverables. You agree not to export or re-export to entities on the current U.S. export exclusion lists or to any embargoed or terrorist countries as specified in the U.S. export laws. You agree to not use deliverables for prohibited nuclear, missile, or chemical biological weaponry end uses. SUSE assumes no responsibility for your failure to obtain any necessary export approvals.

This SUSE Training Manual is published solely to instruct students in the use of SUSE networking software. Although third-party application software packages may be used in SUSE training courses, this is for demonstration purposes only and shall not constitute an endorsement of any of these software applications.

Further, SUSE LLC does not represent itself as having any particular expertise in these application software packages and any use by students of the same shall be done at the student’s own risk.

Table of Contents

Documentation Conventions:....6

Section 1 : Demonstration of SUSE CaaS Platform....9

(No Exercises)....10

Section 2 : Install and Configure SUSE CaaS Platform....11

Exercise 1 : Start the Lab VMs....12

Task 1: Start the Lab VMs....12

Exercise 2 : Install a CaaS Platform Admin Node....15

Task 1: Install the Admin Node....15

Task 2: Configure the Admin Node....17

Exercise 3 : Deploy Cluster Node....22

Task 1: Deploy the Cluster Nodes configuration....22

Task 1: Deploy the Cluster Nodes....22

Exercise 4 : Bootstrap the Kubernetes Cluster....24

Task 1: Bootstrap the Cluster....24

Section 3 : Connecting to the Cluster....28

Exercise 1 : Use the kubectl Command to Display Info About the Cluster....29

Task 1: Download the kubeconfig File....29

Task 2: Use the kubectl Command....30

Exercise 2 : Deploy the Kubernetes Dashboard....33

Task 1: Deploy the Kubernetes Dashboard....33

Task 2: Access the Kubernetes Dashboard....33

Section 4 : Deploying a Workload....35

Exercise 1 : Deploy a Simple Pod on Kubernetes....36

Task 1: View a Manifest for the Deployment....36

Task 2: Deploy a simple application (GUI option)....37

Task 3: Deploy a simple application (Command Line option)....39

Exercise 2 : Delete a Deployment on Kubernetes....41

Task 1: Delete the Deployment (GUI Option)....41

Task 2: Delete the Deployment (Command Line option)....41

Exercise 3 : Scale Out a Deployment....43

Task 1: View the Manifest for the Deployment....43

Task 2: Scale the Deployment (GUI option)....43

Task 3: Scale the Deployment (Command Line Option)....45

Task 4: Scale the Deployment Out....45

Section 5 : Monitoring a Cluster....46

Exercise 1 : Access cAdvisor on the Kubernetes Cluster Nodes....47

Task 1: Access the cAdvisor Dashboard....47

Section 6 : Work With Kubernetes....49

Exercise 1 : Update a Deployment....50

Task 1: Create a New Manifest for the Deployment....50

Task 2: Update the Deployment....50

Exercise 2 : Update a Deployment Via Rolling Updates....52
Task 1: Create a New Manifest for the Deployment....52
Task 2: Update the Deployment....52
Exercise 3 : Expose a Service Running in a Pod....54
Task 1: Create a Manifest for the Service....54
Task 2: Define the Service....54
Task 3: Access the Exposed Service....55
Exercise 4 : Define Limits for Containers and Pods in Kubernetes....56
Task 1: Create a New Namespace in the Cluster....56
Task 2: Create a Manifest for the Limits....56
Task 3: Apply the Limits to the Namespace....57
Exercise 5 : Introduction to Helm....58
Task 1: Initialize Helm....58
Task 2: Use Some Basic Helm Commands....58
Exercise 6 : Deploy an Application with Helm....60
Task 1: Create Helm Chart Config File....60
Task 2: Deploy the Helm Chart....60
Task 3: Access the Application Deployed by the Helm Chart....61
Task 4: Update the Application Release....61
Task 5: Delete a Deployed Helm Chart Release....62
Section 7 : Storage....64
Exercise 1 : Configure NFS Persistent Storage....65
Task 1: Create the Persistent Volume on the NFS Server....65
Task 2: Create the Manifest for the Persistent Volume....65
Task 3: Create the Manifest for the Persistent Volume Claim....65
Task 4: Create the Manifest for the Busybox Instance....66
Task 5: Create the Manifests for the Web Frontend....66
Task 6: Deploy the Objects....67
Task 7: Test the Persistent Data....68
Task 8: Remove the Objects from the Cluster....68
Exercise 2 : Configure Persistent Storage with a NFS StorageClass....70
Task 1: Create the Directory for the Persistent Volumes on the NFS Server....70
Task 2: Define a Service Account, Cluster Role and Cluster Role Binding....70
Task 3: Define a Storage Class....72
Task 4: Create a Persistent Volume Claim and Pod that Uses It....73
Task 5: Create a Persistent Volume Claim and Pod that Uses It....75
Task 6: Remove the Objects from the Cluster....76
Section 8 : Horizontal Scale....77
Exercise 1 : Configure Horizontal Pod Autoscaling....78
Task 1: Create the Manifest for the App to be Scaled....78
Task 2: Open the Manifest for the App’s Service....80
Task 3: Open the Manifest for the Autoscaler....80
Task 4: Open the Manifest for the Load Generator....80
Task 5: Deploy the AutoScaler Objects....81
Task 6: Cause the Deployment To Scale Out....82
Task 7: Cause the Deployment To Scale Back....82

Task 8: Clean Up the Deployments....82
Section 9 : Deploy an Online shop....84
Exercise 1 : Install sock shop microservices demo in a Kubernetes Cluster....85
Task 1: Create a new namespace for our demo....85
Task 2: Watch Pods as the launch....85
Task 3: Install Sock Shop....85
Task 4: Visit sock shop website....86
Section 10 : Appendix: Bonus Labs....88
Exercise 1 : Deploy a Stateful App with Volume Storage on Kubernetes....89
Task 1: Create a Manifest for the Deployment....89
Task 2: Deploy the Pod....90
Exercise 2 : Use Environment Variables in a Pod....91
Task 1: Create a Manifest for the Deployment....91
Task 2: Deploy the Pod....91
Task 3: Display the Environment Variable in the Container....92
Task 4: Delete the Pod....92
Exercise 3 : Define and Access a Secret in Kubernetes....93
Task 1: Create a Manifest for the Deployment....93
Task 2: Define the Secret....93
Task 3: Access the Secret....94
Exercise 4 : Deploy an Invalid Pod on Kubernetes....95
Task 1: Create a Manifest for the Pod....95
Task 2: Deploy the Pod....95
Exercise 5 : Delete a Namespace from a Kubernetes Cluster....97
Task 1: Delete a Namespace from the Cluster....97
Lab Variables:....98

Documentation Conventions:

The following typographical conventions are used in this manual:

Bold	Represents things you should pay attention to or buttons you click, text or options that you should click/select/type in a GUI.
Bold Gray	Represents the name of a Task or in the context of what is seen on the screen, the screen name, a tab name, column name, field name, etc.
Bold Red	Represents warnings or very important information.
Option > Option > Option	Represents a chain of items selected from a menu.
BOLD_UPPERCASE_ITALIC	Represents an “exercise variable” that you replace with another value.
`bold monospace`	Represents text displayed in a terminal or entered in a file.
`bold monospace blue`	Represents commands entered at the command line.
`bold monospace green`	Represents a file name.

1 Demonstration of SUSE CaaS Platform

Description:

How to highlight the Key Benefits of SUSE CaaS Platform

(No Exercises)

2 Install and Configure SUSE CaaS Platform

Description:

In this section, you install and configure a SUSE CaaS Platform cluster.

Note 1:

In the provided lab environment the SMT server is already configured with all of the infrastructure services (NTP / DHCP/ T FTP / PXE / DNS / NFS / Docker Registry etc).

2- 1 Start the Lab VMs

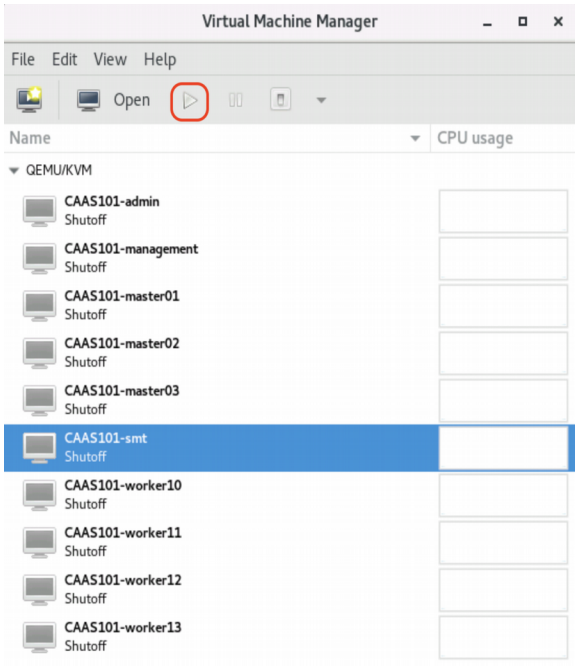
Description:

In this exercise, you use the Virt-Manager utility to start the lab VM(s) in the proper order.

Task 1: Start the Lab VMs

1. On the **VM host**, launch the Virt-Manager utility
   (4th icon from the left on the dock at the bottom of the screen)
   You should see the course VMs listed.

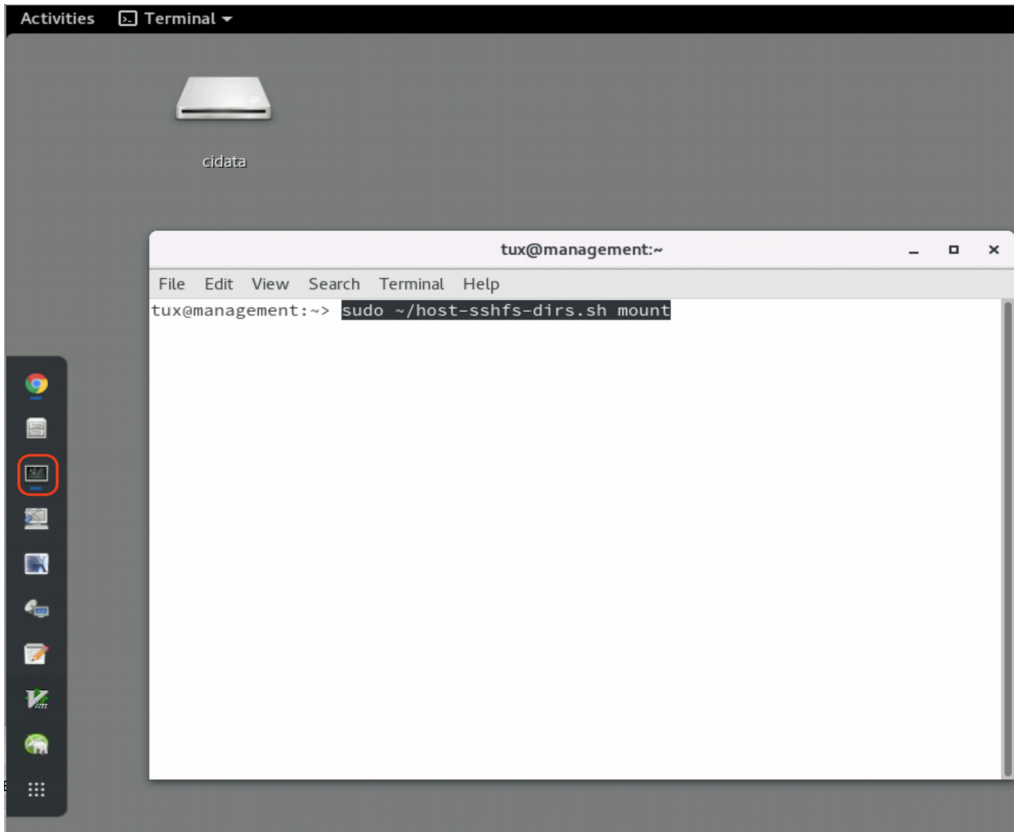
2. Right-click on the **CAAS101-smt** VM and select **Run**



3. Double-click on the **CAAS101-smt** VM to view its console
   When the **CAAS101-smt** VM has finished booting completely, close its console window.
   (You can use the CPU utilization graph in Virt-Manager to see when the node is finished booting and starting services).
4. Repeat the same previous steps for the **CAAS101-management** VM to power it on
   When the **CAAS101-management** VM has finished booting you will use it as your management workstation for all tasks including managing VMs with the Virt-Manager

utility.

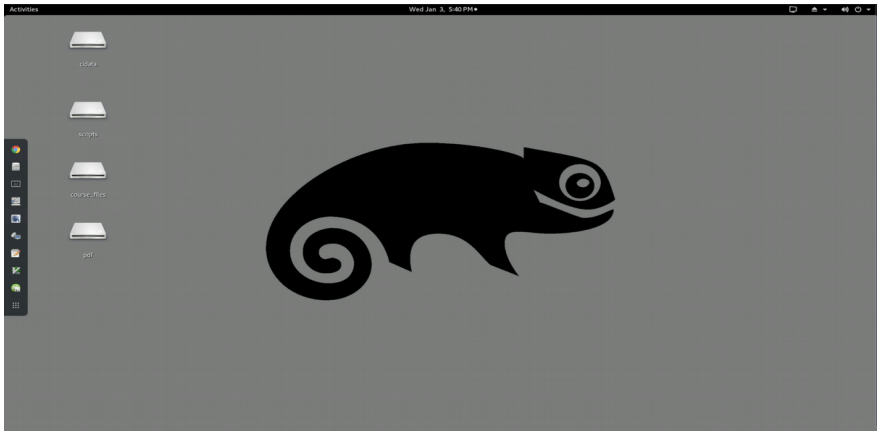
5. Make sure you double click on the **CAAS101-management** so you view it full screen
6. Open a Terminal session by clicking on the 3rd icon down



7. To mount all of the course specific directories that reside on the host machine into the management VM, as the **tux** user, at a command prompt, enter the following command:

   ```
   sudo ~/host-sshfs-dirs.sh mount
   ```

   All directories that contain course specific files such as PDFs, additional course files, etc. should now be mounting onto the same corresponding directories in the VM as they are in the host.
8. When you are done your desktop should look like this:



Summary:

In this exercise, you started the required lab VM(s) in the proper order.

(End of Exercise)

2- 2 Install a CaaS Platform Admin Node

Description:

In this exercise, you install a SUSE CaaS Platform admin node.

Dependencies:

The SMT server must be configured.

DHCP, PXE and DNS must be configured.

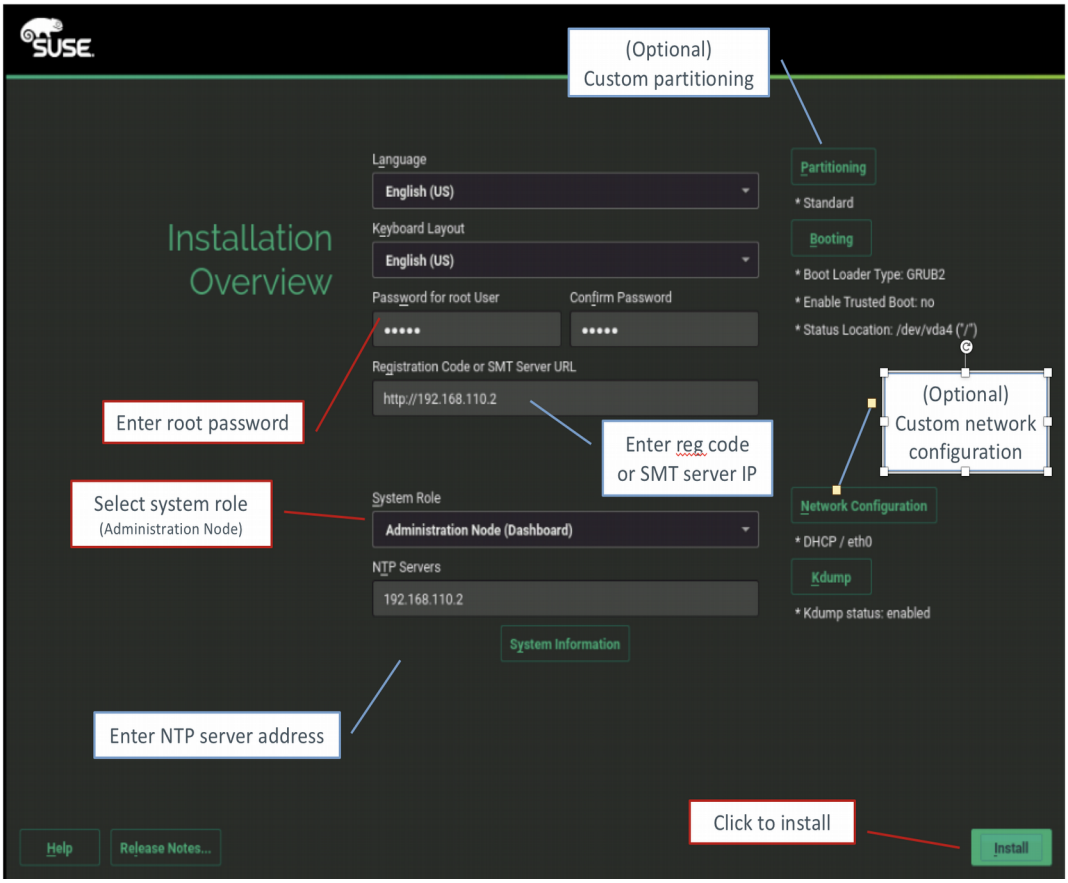
Task 1: Install the Admin Node

1. Open Virt-Manager and double-click on the **CAAS101-admin** VM
   Select the graphical console view
2. Power on the **CAAS101-admin** VM
3. Boot the VM from the DVD (should be the default)
4. On the boot screen, use the arrow keys to select **install**:

Install *CAASP_ADMIN_MAC* (admin)

5. The installation process should begin.
6. On the **Install Overview** screen select/enter the following (leaving all others with their default values):

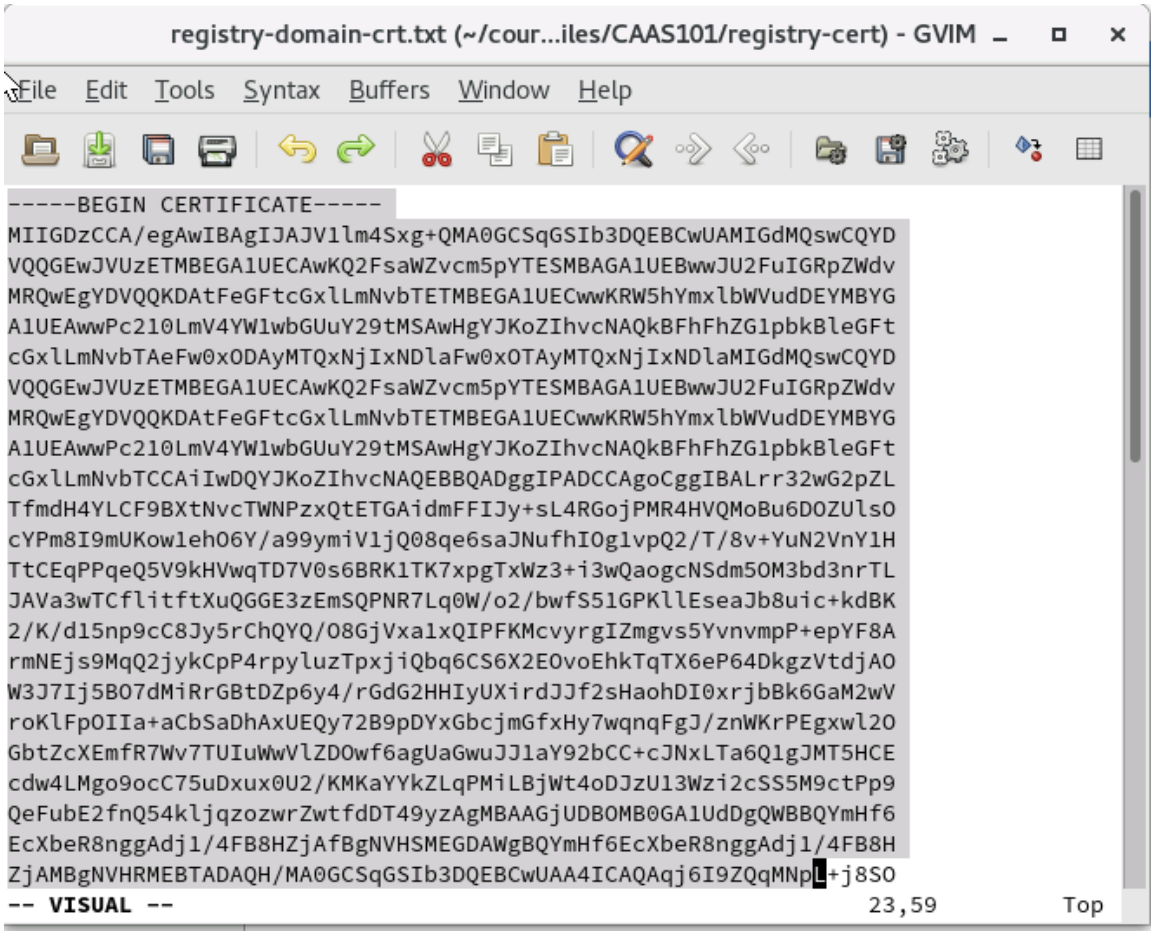
   Password for root user: **linux**
   Registration Code or SMT Server URL: **http://192.168.110.2**
   System Role: **Administration Node (Dashboard)**
   NTP Servers: **192.168.110.2**



7. Click **Install**
   When prompted that the **password is too simple**, click: **Yes**
   When prompted to **enable the updates repository** during installation, click: **Yes**
   When prompted to Confirm Installation, click: **Install**
8. Wait for the admin node to finish installing

Task 2: Configure the Admin Node

1. Open `~/course_files/CAAS101/registry-cert/registry-domain-cert.txt` with a text editor, select all of the text and **Copy** the contents



2. When the admin node is finished installing, has rebooted and is at a login prompt, open a web browser and point to:

   https://192.168.110.99

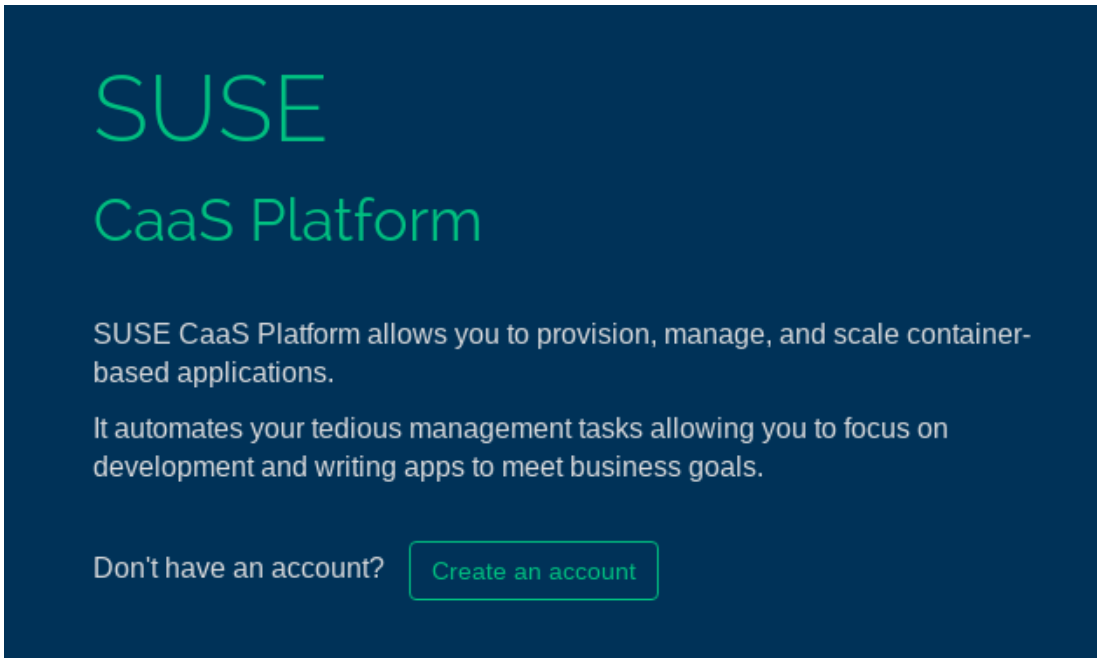
Note:

If you get an error that the site can't be reached or an error stating that the database is not running, wait a minute or two and then refresh the web page.

If presented with an untrusted certificate warning, accept the certificate to proceed.

You should see the **SUSE Container as as Service Platform** login screen.

3. To create the initial admin account, click: **Create an account**

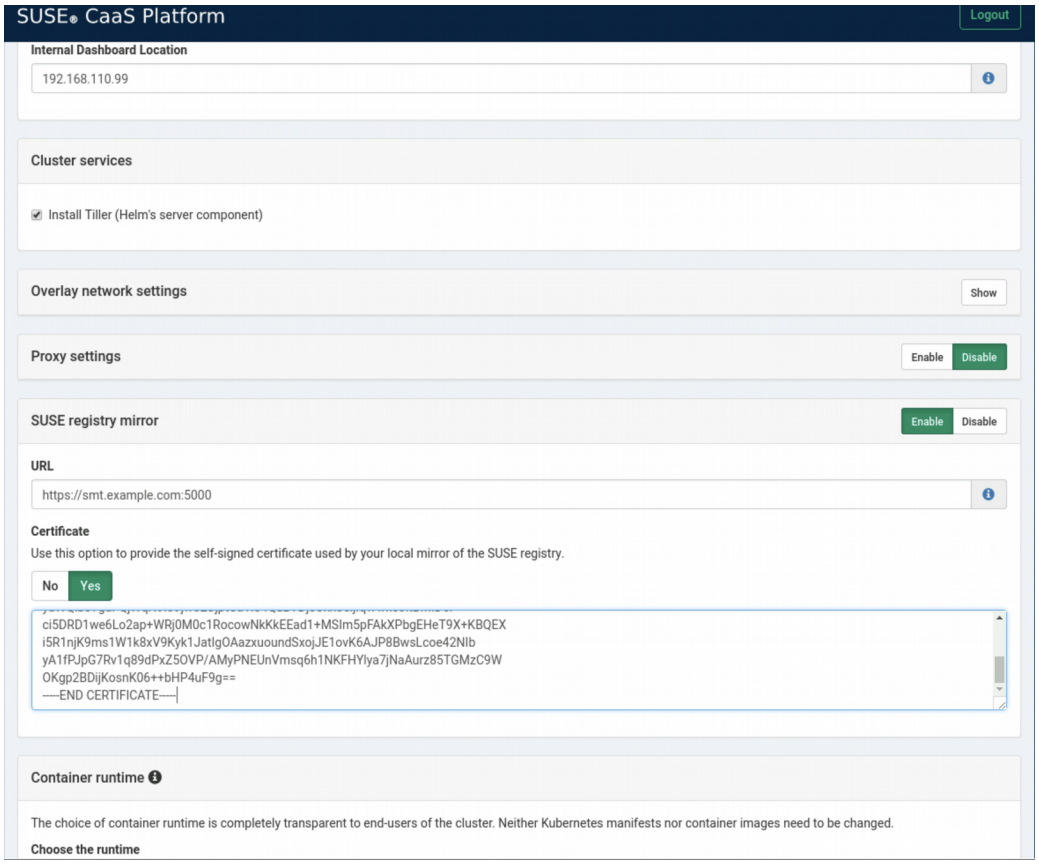


4. Under **Sign Up** enter the following:

   Email: **admin@example.com**
   Password: **susecaasp**

5. Click: **Create Admin**

6. On the **Initial CaaSP Configuration** screen, enter/select the following:



Generic settings:
- **Dashboard location**: **(accept default)**

Cluster services:
- **Install Tiller (Helm's server component)**: **(checked)**

Overlay network settings:
- **Cluster CIDR: (accept default)**
- **Cluster CIDR (lower bound): (accept default)**
- **Cluster CIDR (upper bound): (accept default)**
- **Node allocation size (CIDR length per worker node): (accept default)**
- **Services CIDR: (accept default)**
- **API IP address: (accept default)**
- **DNS IP address: (accept default)**

Proxy settings: **(disable)**

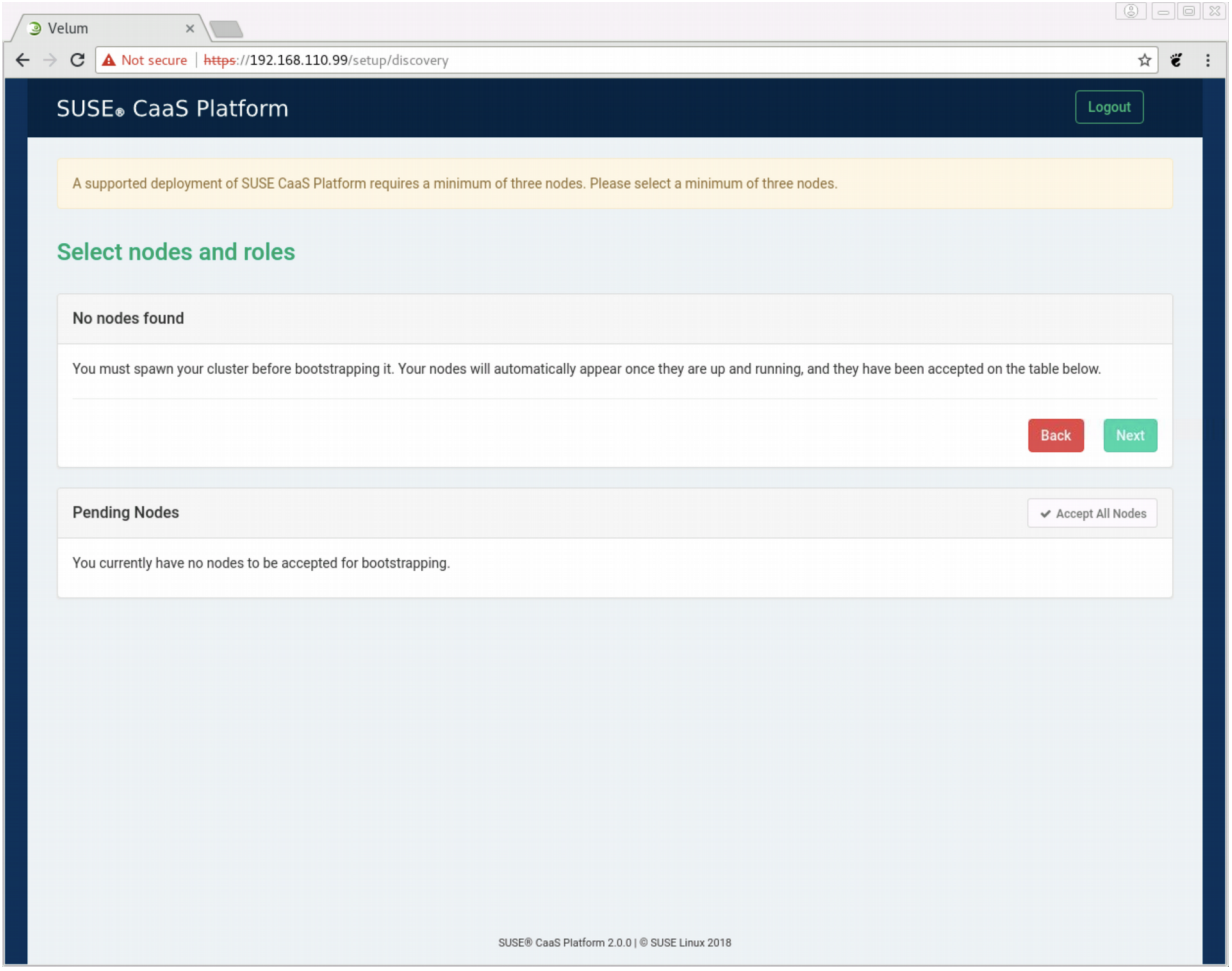
SUSE registry mirror: **(Enable)**
- **URL**: **https://smt.example.com:5000**

Certificate: **(Enable)**

7. Paste the contents of the `~/course_files/CAAS101/registry-cert/registry-domain-cert.txt` (from Step 1)
8. On the **Bootstrap your Container as a Service Platform** screen, verify that the **autoyast=** URL that is displayed matches what you entered in the PXE configuration files in the `/srv/tftpboot/pxelinux.cfg/` directory.
9. Click **Next**

   You should see the **Select nodes and roles** screen.

   This is the end of the admin node installation process. You will now continue with the bootstrap exercises.



10. Click **Next** to Prepare to Nodes to join the cluster

Summary:

In this exercise, you installed the admin node (via PXE or from the installation DVD). You then performed the initial configuration of the Admin node.

(End of Exercise)

2- 3 Deploy Cluster Node

Description:

In this exercise, you select the cluster master nodes and worker nodes and then bootstrap the cluster.

Dependencies:

The SMT server must be configured.

DHCP, PXE and DNS must be configured.

The admin node must be installed and configured.

Task 1: Deploy the Cluster Nodes configuration

This will copy the configuration out to a virtual USB drive on each node

1. From a **Terminal** on the **CAAS101-management** VM
   ```
   cd ~/course_files/CAAS101/scripts
   sh ./update-cloudinit-config.sh
   ```
2. when prompted to continue type **yes**
3. Enter the **linux** when prompted
4. when if finishes you can type **exit** to exit the terminal session

Task 1: Deploy the Cluster Nodes

5. Open Virt-Manager and double-click on the **CAAS101-master01** VM
   Select the graphical console view
6. Power on the **CAAS101-master01, CAAS101-master02**, **CAAS101-master03**, **CAAS101-worker10**, **CAAS101-worker11** and **CAAS101-worker12**
   (DO NOT deploy the **CAAS101-worker13** yet. That will be done in an upcoming exercise.)

Summary:

In this exercise, you selected the cluster master nodes and worker nodes.

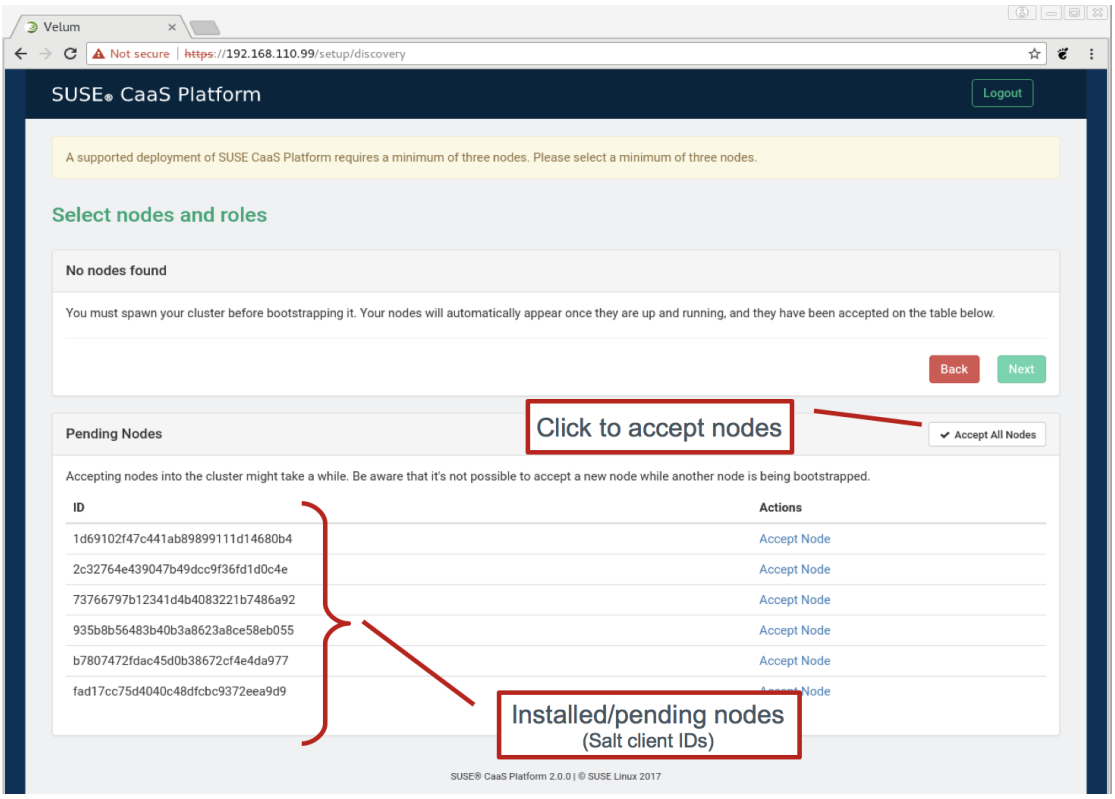
(End of Exercise)

2- 4 Bootstrap the Kubernetes Cluster

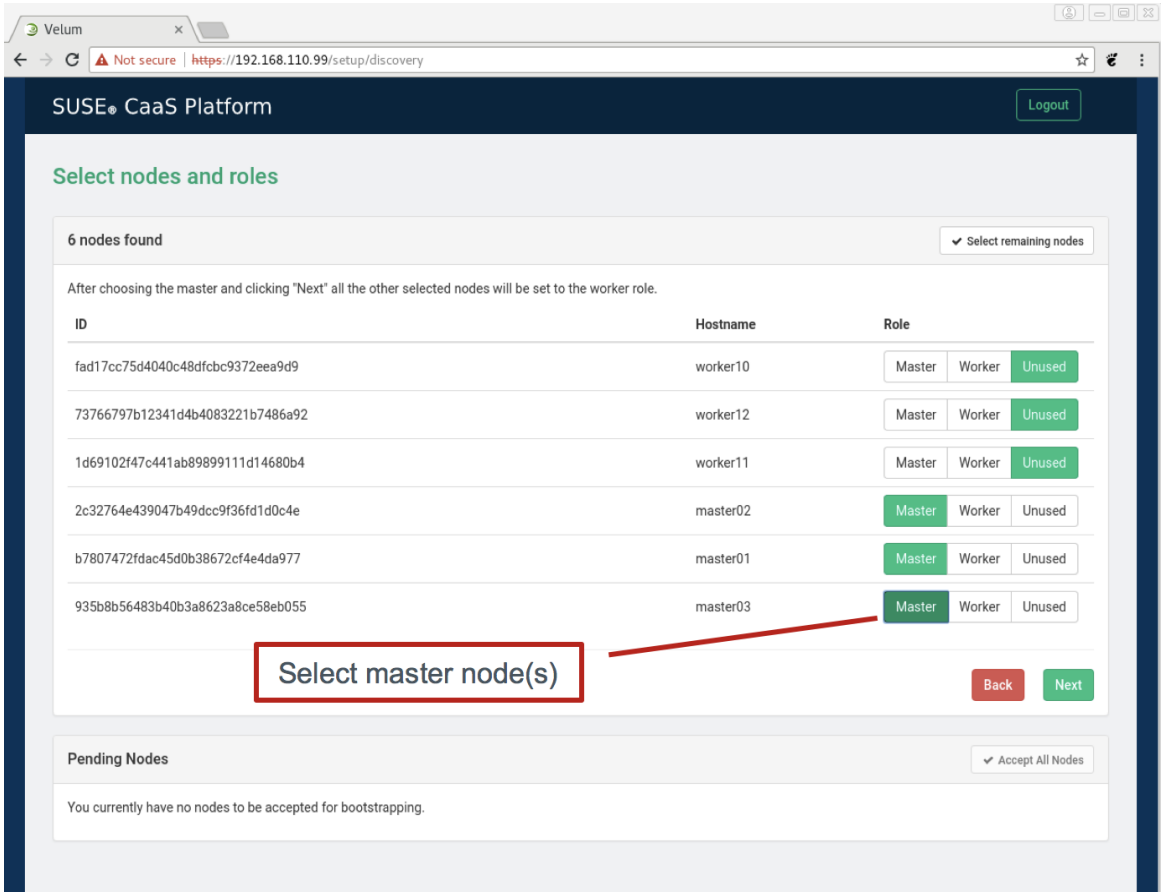
Task 1: Bootstrap the Cluster

1. When all of the nodes have been installed and are displayed in the **Pending Nodes** section, click **Accept All Nodes**

   You should see their actions change to “Acceptance in progress”.

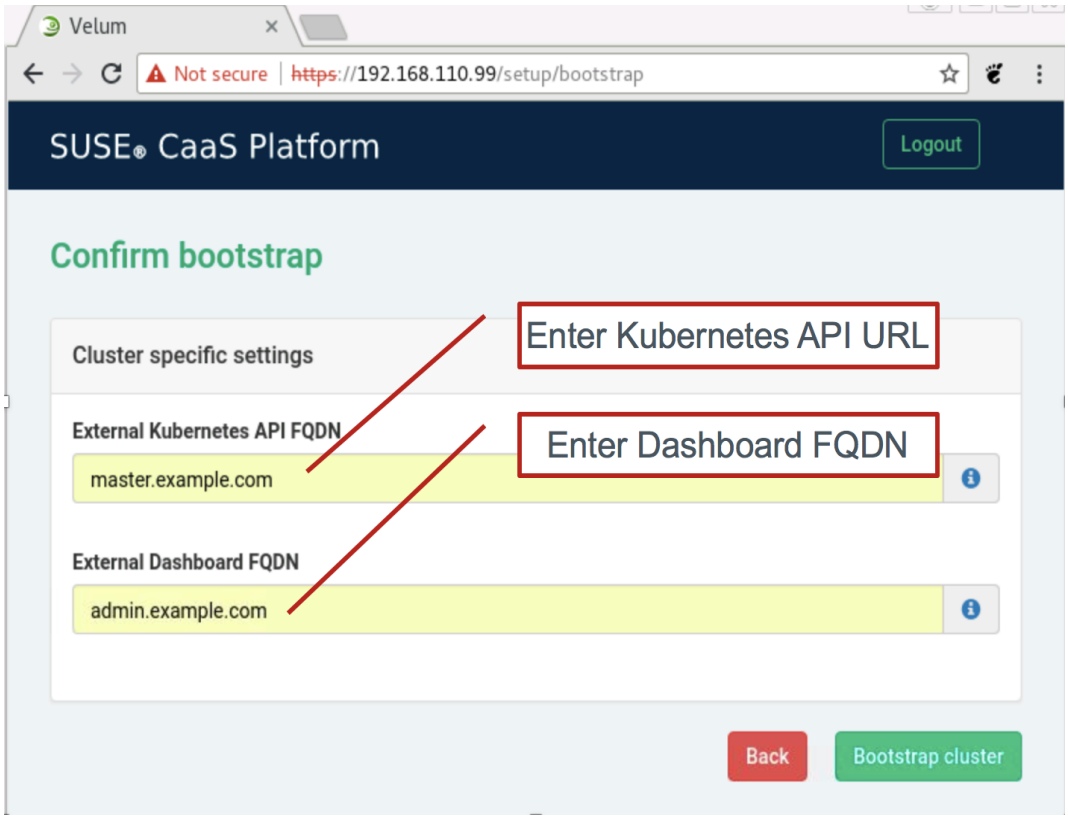


2. When all of the nodes appear in the **nodes found** section, select the **master01** , **master02** and **master03** nodes to be the **master** by selecting **Master** from the **Role** column next to it



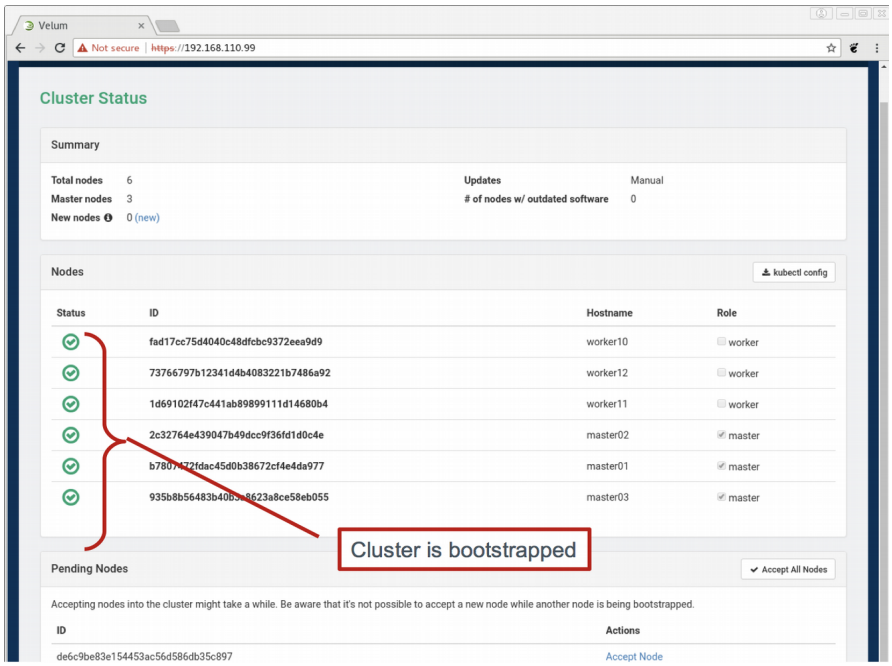
3. Select all other nodes to be bootstrapped as worker nodes by selecting **Worker** from the **Role** column next to it
4. Click **Next**
5. On the **Confirm Bootstrap** screen, enter/select the following:

   External Kubernetes API server FQDN: **master.example.com**
   External Dashboard FQDN: **admin.example.com**



6. Click **Bootstrap cluster**

7. Back on the **Cluster Status** screen, watch the **Status** column as the cluster is bootstrapped. When the bootstrapping is finished the nodes will have a green checkmark in the **Status** column.



If you see a red warning across the top of the Velum screen saying that it “cannot connect with Velum API”, follow the instructions and reload the browser session.

Summary:

In this exercise, you selected the cluster master nodes and worker nodes and then bootstrapped the cluster.

(End of Exercise)

3 Connecting to the Cluster

Description:

In this section you will learn how to connect to the Cluster via both the command line and a GUI interface.

3- 1 Use the kubectl Command to Display Info About the Cluster

Description:

In this exercise, you use the `kubectl` command to display info about the Kubernetes cluster.

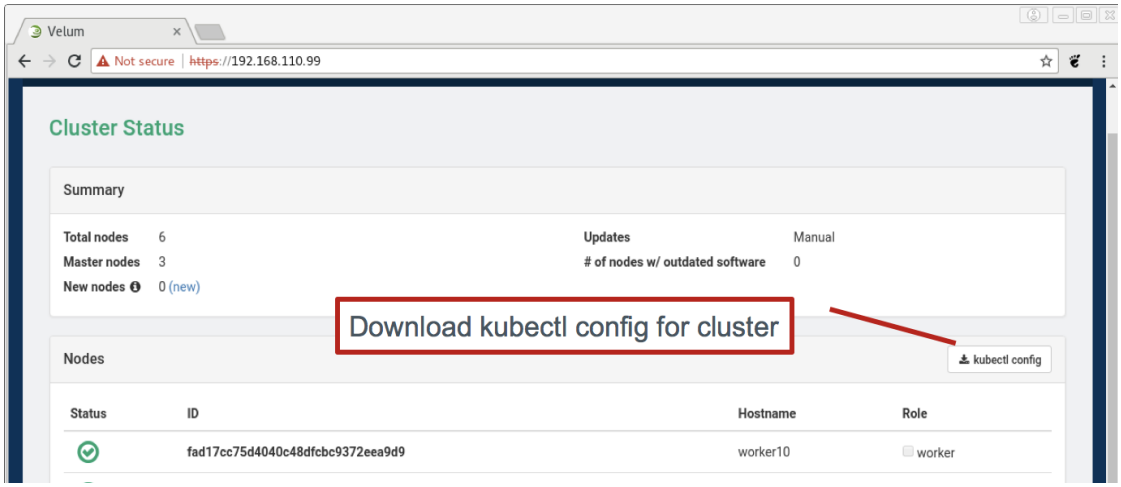
Task 1: Download the kubeconfig File

1. Open a web browser and point to:

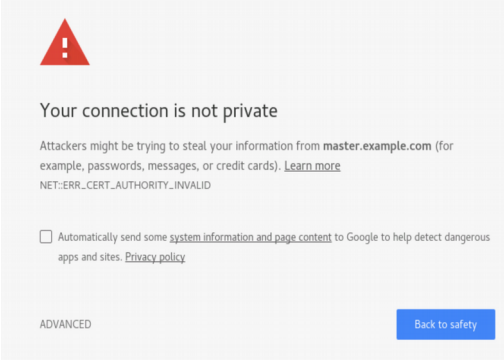
   http://192.168.110.99

   You should see the **SUSE Container as a Service Platform** login page.
2. Log in with the following credentials:

   Username: **admin@example.com**
   Password: **susecaasp**
3. On the **Cluster Status** page, in the **Nodes** section, click the **kubectl config** download button



4. If prompted that the SSL connection is unsafe, use your browser's method to proceed anyway



5. When prompted to log in, enter the following credentials:

   Username: **admin@example.com**
   Password: **susecaasp**

   Save the file in your Downloads directory.

6. Open a **terminal** session and enter the following commands to copy the kubectl config file to its default location:

```
cp ~/Downloads/kubeconfig ~/.kube/config
```

Note:

If you don't want to put the kubectl config file in the default location, or if you have multiple Kubernetes clusters you need to access and therefor multiple corresponding kubectl config files, you can use the `--kubeconfig=<config_file>` option for the `kubectl` command to specify the kubectl config file you want to use.

Task 2: Use the kubectl Command

1. From a terminal, enter the following command to view the URL of the Kubernetes master:

```
kubectl cluster-info
```

   You should see the URL of the Kubernetes master displayed.

2. Enter the following command to see a more detailed output of the status of the cluster:

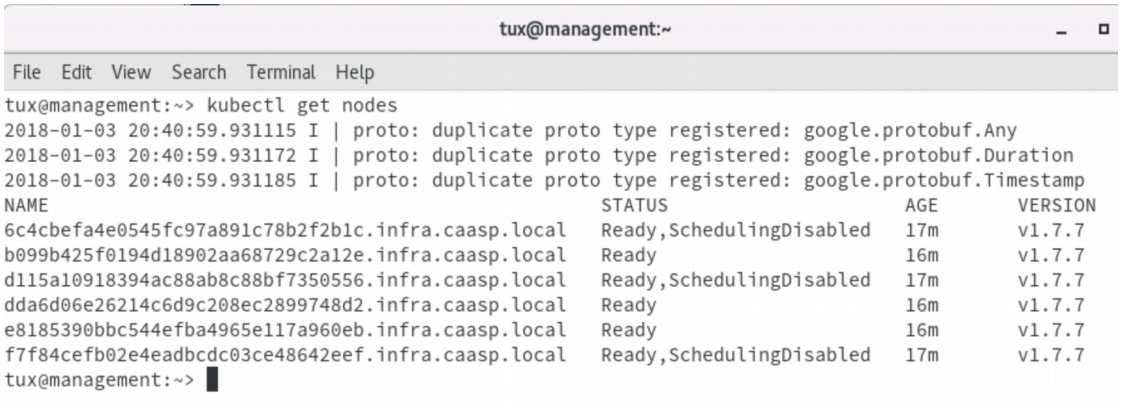
```
kubectl cluster-info dump
```

   You should see a json dump of the status of cluster.

3. Enter the following command to display a list of Kubernetes nodes:

```
kubectl get nodes
```

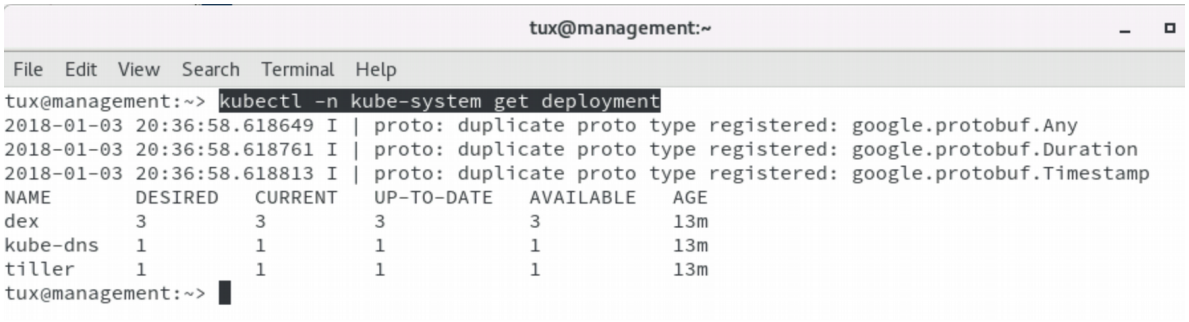
You should see the list of nodes, their node names, status, age and version displayed.



4. Enter the following command to display a list of Kubernetes nodes:

```
kubectl -n kube-system get deployments
```

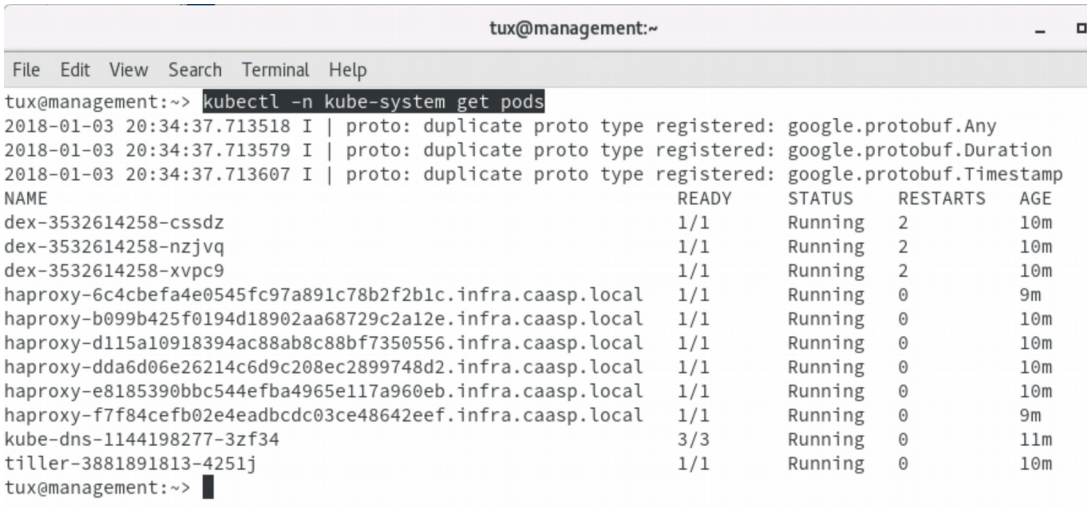
You should see the list of deployments, their nodes names, status, age and version displayed.



5. Enter the following command to display a list of Kubernetes pods:

   ```
   kubectl -n kube-system get pods
   ```

   You should see the list of pods, their names, status, and age displayed.



6. Enter the following command to load Heapster

   ```
   kubectl apply -f ~/course_files/CAAS101/manifests/monitoring/
   ```

Summary:

In this exercise, you downloaded the kubectl config file from the cluster and then ran some kubectl commands to view information about the cluster.

(End of Exercise)

3- 2 Deploy the Kubernetes Dashboard

Description:

In this exercise, you deploy the Kubernetes Dashboard.

Task 1: Deploy the Kubernetes Dashboard

1. On the **CAAS101-management VM**, open a terminal
2. Enter the following command to deploy the Kubernetes Dashboard:

   ```
   kubectl apply -f ~/course_files/CAAS101/manifests/dashboard
   ```

   You should see that a **secret**, **serviceaccount** , **role**, **rolebinding**, **deployment** and **service** were created:
3. Enter the following command to view the pods deployed in the **kube-system** namespace:

   ```
   kubectl -n kube-system get pods
   ```

   You should see the kubernetes-dashboard listed among the pods.
4. Enter the following command to view the deployments in the **kube-system** namespace:

   ```
   kubectl -n kube-system get deployments
   ```

   You should see the kubernetes-dashboard listed among the deployments.
5. Enter the following command to view the services in the **kube-system** namespace:

   ```
   kubectl -n kube-system get services
   ```

   You should see the kubernetes-dashboard listed among the services.

Task 2: Access the Kubernetes Dashboard

1. Enter the following command to start a kubectl proxy in a screen session:

   ```
   screen kubectl proxy
   ```

   You should see the proxy session started. Notice that port 8001 on the local host is the port used.
2. Enter the following keystrokes to detach from the screen session:

   ```
   ctrl+a ctrl+d
   ```

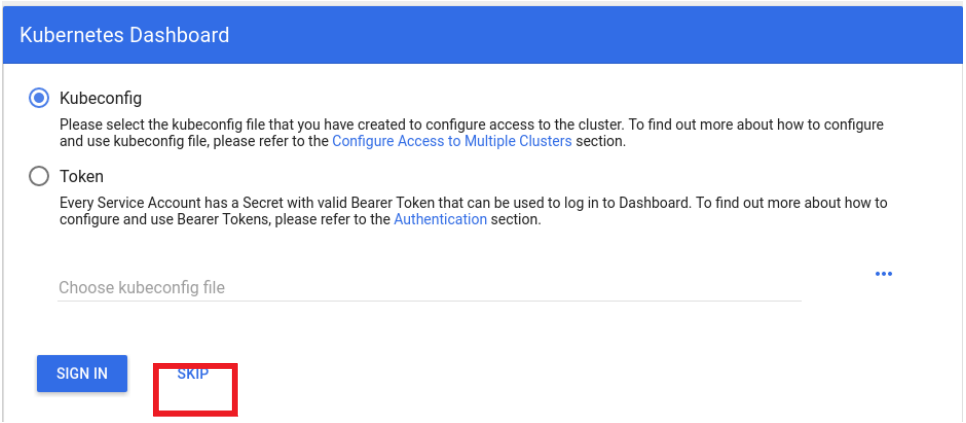
   You should be detached from the screen session and back at a command prompt.

3. Open a web browser and point to:

   `http://localhost:8001/api/v1/namespaces/kube-system/services/https:kubernetes-dashboard:/proxy/#!/login`

   You should see the Kubernetes Dashboard authentication method screen.

4. Click: **Skip** to login without a token



5. From the list on the left, in the **Namespace** section, from the drop-down list (which is probably displaying the **default** namespace), select **kube-system**

   You should see the kubernetes dashboard deployment in the **kube-system** namespace.

6. From that same drop-down menu, select **default**

   You should see that there is nothing deployed in the **default** namespace.

Summary:

In this exercise, you deployed the Kubernetes Dashboard to the cluster. You then started a kubectl proxy in a screen session and accessed the Kubernetes Dashboard.

(End of Exercise)

4 Deploying a Workload

Description:

In this section you will deploy a simple workload in Kubernetes.

4- 1 Deploy a Simple Pod on Kubernetes

Description:

In this exercise, you deploy the Nginx web server as a simple pod on the Kubernetes cluster.

Task 1: View a Manifest for the Deployment

1. On the **management workstation**, in the text editor of your choice, open the file:

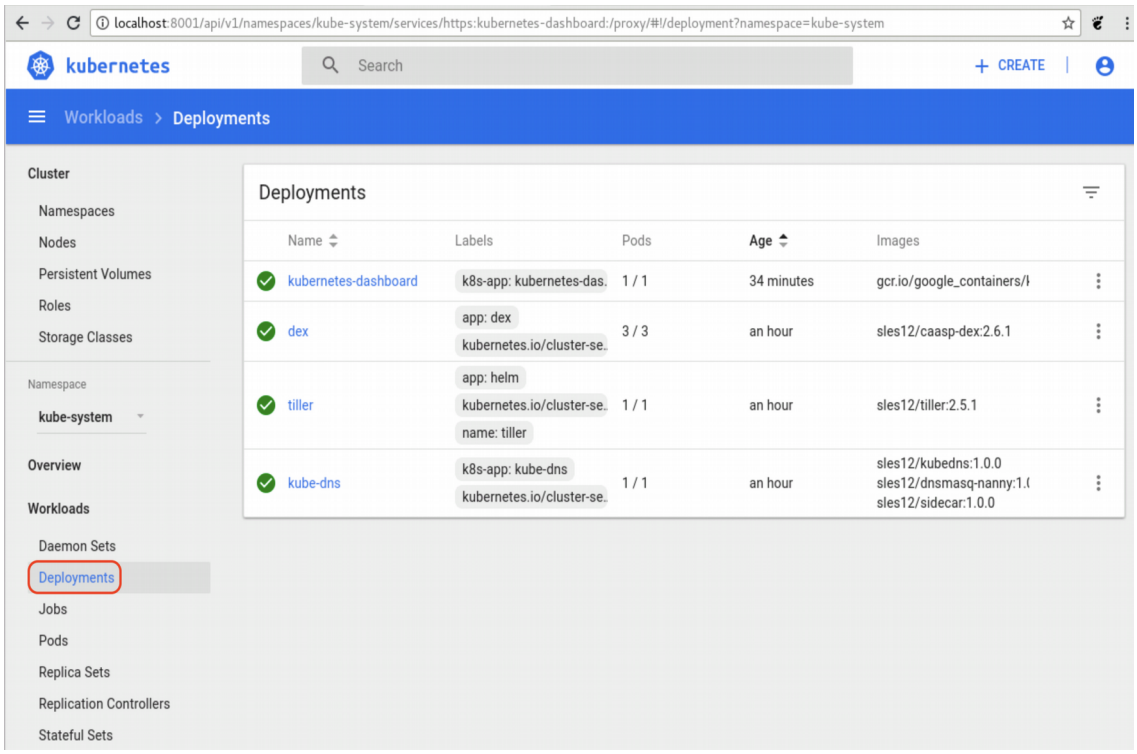
   `~/course_files/CAAS101/manifests/labs/nginx-deployment.yaml`

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  replicas: 1
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: smt.example.com:5000/nginx:1.7.9
        ports:
        - containerPort: 80
```

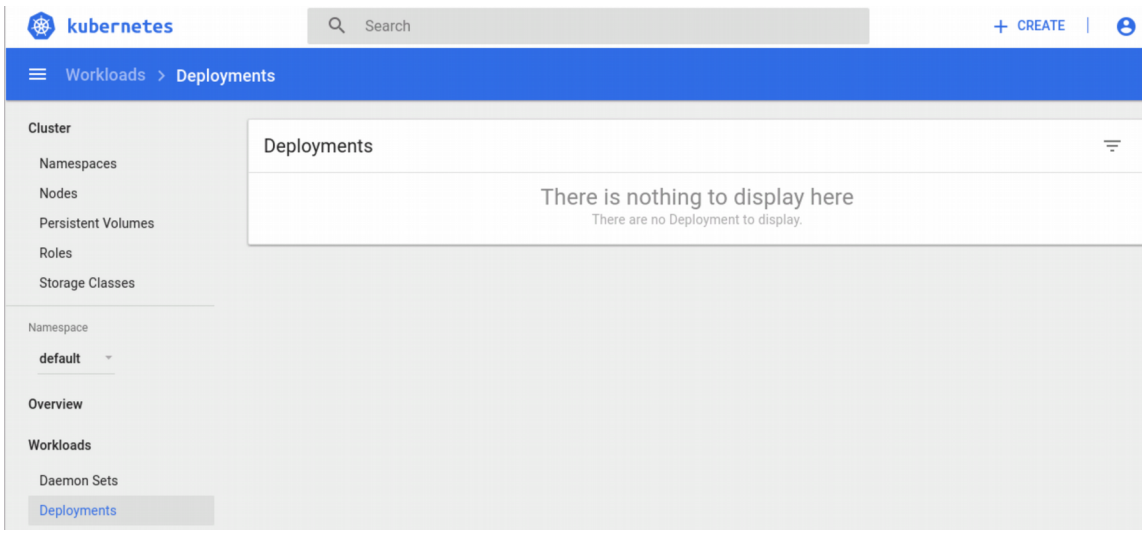
Task 2: Deploy a simple application (GUI option)

To deploy the application, go to the Kubernetes Dashboard

1. Click on **Deployments** on the left hand panel

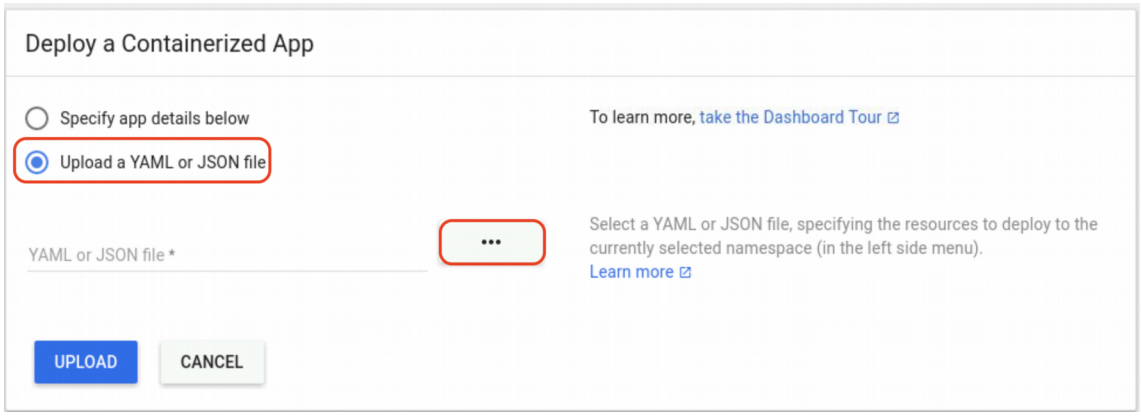


   All of the currently deployed Deployments are in the kube-system namespace
2. Select **Namespace** on the left hand panel
3. Change the **Namespace** to **Default** and select **Deployments** again

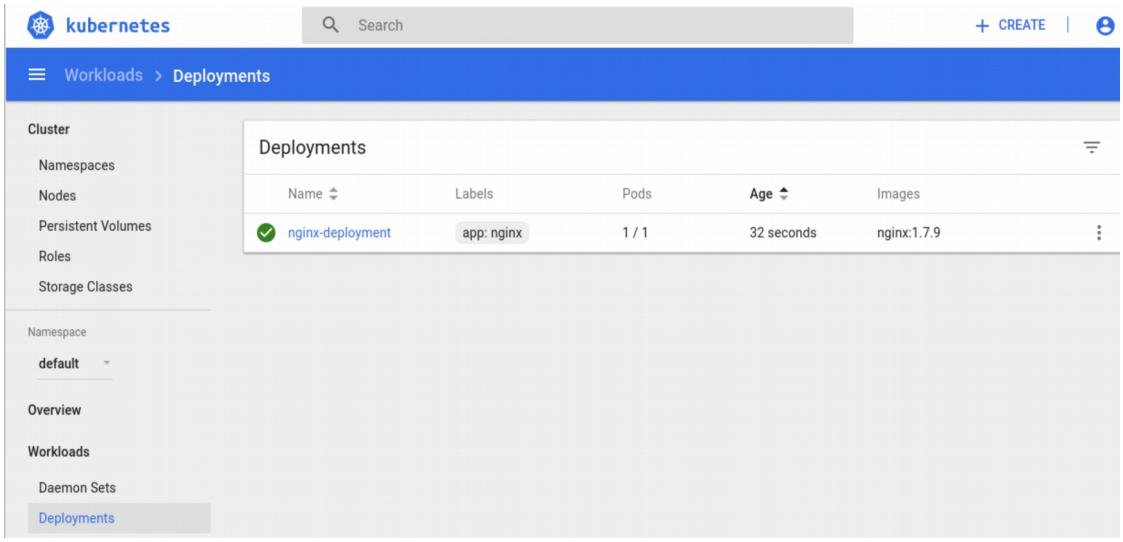


   note there are no current deployments in the **default** namespace

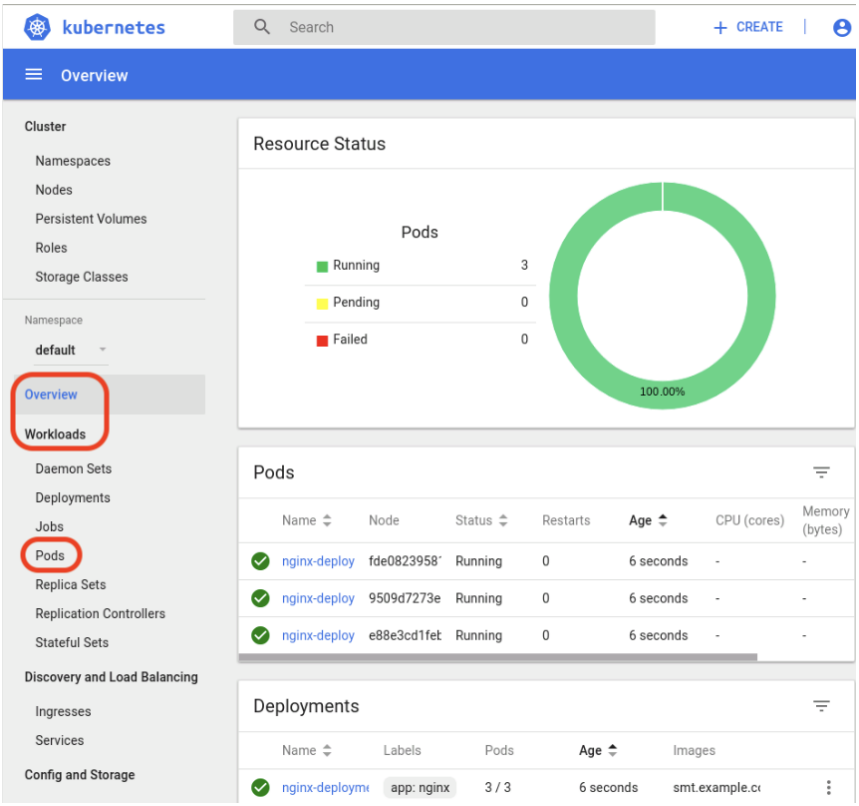
4. Click on **+ CREATE**
5. Select **Upload a YAML or JSON file** and the press '**…' to** select file



6. Browse to `~/course_files/CAAS101/manifests/labs/nginx-deployment.yaml`
7. Press Upload
8. Select **Deployments** view and we should now see the nginx-deployment



9. Explore the Kubernetes Dashboard
   Make sure you look at the **Overview, Workloads, and Pods** views



Task 3: Deploy a simple application (Command Line option)

10. To deploy the pod, open a terminal and enter the following command:

```
kubectl apply -f ~/course_files/CAAS101/manifests/labs/nginx-deployment.yaml
```

```
tux@management:~> kubectl apply -f ~/course_files/CAAS101/manifests/labs/nginx-deployment.yaml
2018-03-13 09:27:42.158048 I | proto: duplicate proto type registered: google.protobuf.Any
2018-03-13 09:27:42.158121 I | proto: duplicate proto type registered: google.protobuf.Duration
2018-03-13 09:27:42.158136 I | proto: duplicate proto type registered: google.protobuf.Timestamp
deployment "nginx-deployment" created
tux@management:~>
```

You should see the deployment “nginx-deployment” was created.

11. Enter the following command to view the deployments:

    ```
    kubectl get deployments
    ```

    You should see that a single instance of the **nginx-deployment** pod is running.
12. Enter the following command to view the deployed pods:

    ```
    kubectl get pods
    ```

    You should see a single instance of the **nginx-deployment** pod running.

Summary:

In this exercise, you launched a single instance of the Nginx web server as a pod in a deployment on the cluster.

(End of Exercise)

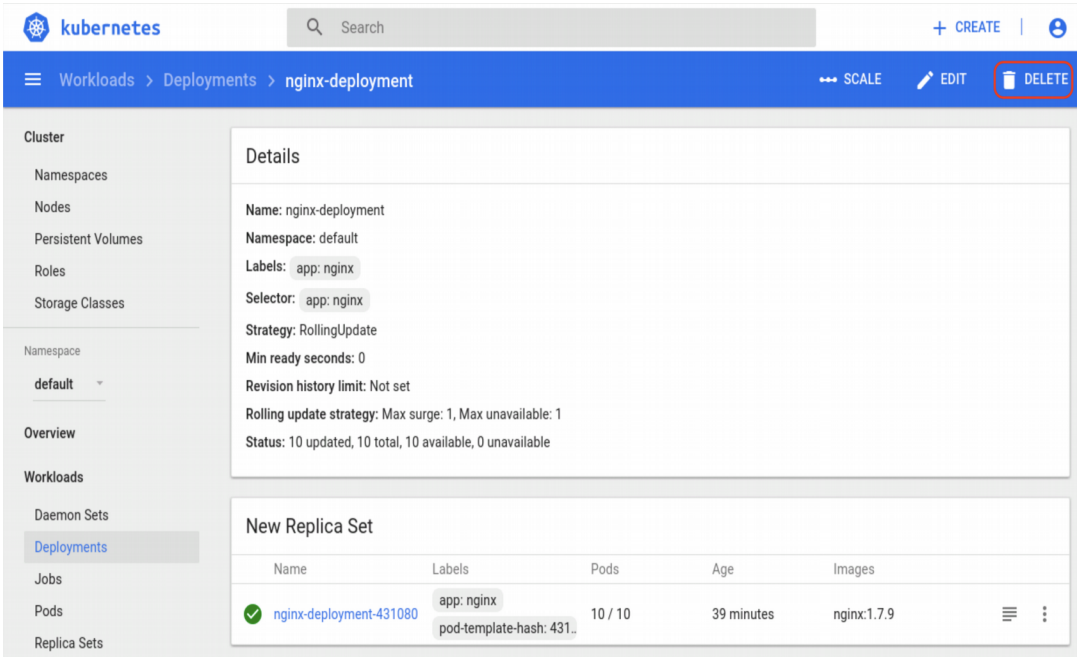
4- 2 Delete a Deployment on Kubernetes

Description:

In this exercise, you delete the Nginx web server deployment that was previously deployed on the Kubernetes cluster.

Task 1: Delete the Deployment (GUI Option)

1. Select **nginx-deployment** under Deployments
2. Select **Delete**



Task 2: Delete the Deployment (Command Line option)

3. To view the deployments, on the **CAAS101-management VM**, enter the following command:

   ```
   kubectl get deployments
   ```

   You should see that one instance of the **nginx-deployment** is running.

4. Enter the following command to delete the deployment:

   ```
   kubectl delete deployment nginx-deployment
   ```

   You should see the **nginx-deployment** was deleted.

5. View the deployments again:

   ```
   kubectl get deployments
   ```

   You should see that the **nginx-deployment** is no longer running.

6. View the pods:

   ```
   kubectl get pods
   ```

   You should see that all of the pods that were part of the **nginx-deployment** are no longer running.

Summary:

In this exercise, you deleted the Nginx web server deployment that was previously deployed on the cluster.

(End of Exercise)

4- 3 Scale Out a Deployment

Description:

In this exercise, you scale out a running Deployment.

Task 1: View the Manifest for the Deployment

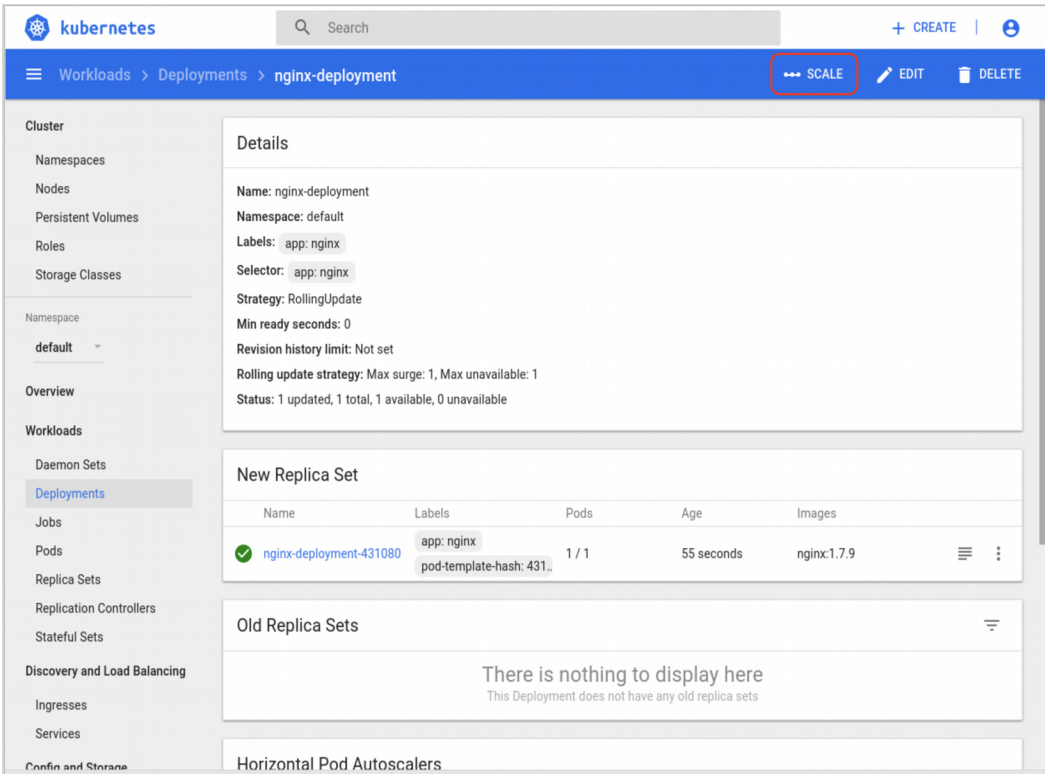
1. In the text editor of your choice, open the `~/course_files/CAAS101/manifests/labs/nginx-scaleout.yaml` file

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  replicas: 4
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: smt.example.com:5000/nginx:1.7.9
        ports:
        - containerPort: 80
```

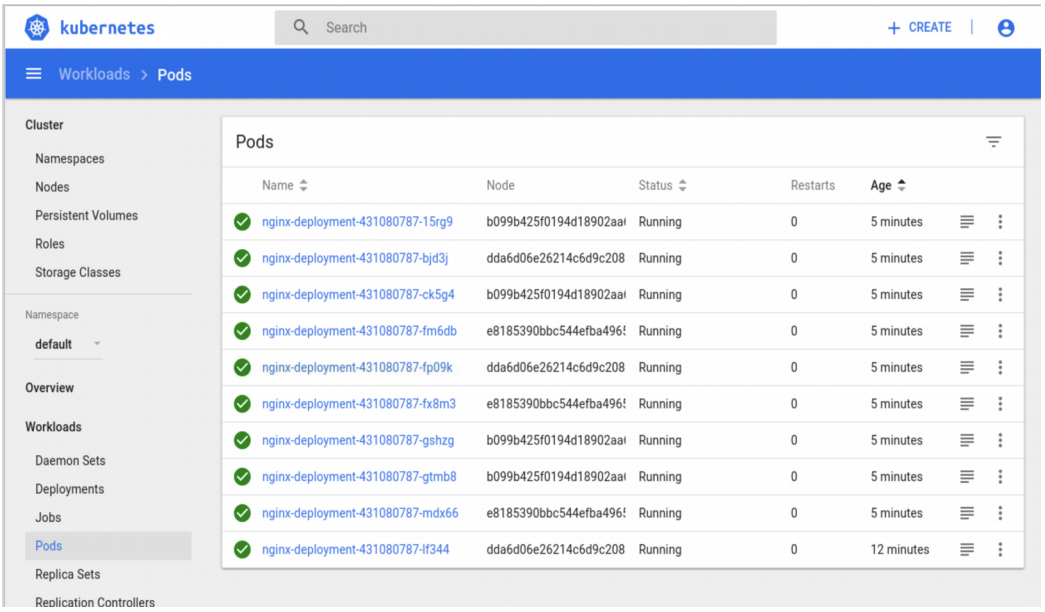
Task 2: Scale the Deployment (GUI option)

1. From **Kubernetes Dashboard** select **Create** and open `~/course_files/CAAS101/manifests/labs/nginx-scaleout.yaml`
2. Press the **Deploy** button
3. Select the **nginx-deployment** under **Deployments**

4. click the **Scale** Button and change the number of **Desired pods** to 10



5. View the deployed pods by selecting **Pods** in the left hand panel



6. Scale the Pods back down to 4

Task 3: Scale the Deployment (Command Line Option)

1. To scale the deployment, open a terminal and enter the following command:

   ```
   kubectl apply -f ~/course_files/CAAS101/manifests/labs/nginx-scaleout.yaml
   ```

2. Enter the following command to view the deployments:

   ```
   kubectl get deployments
   ```

   You should see that one instance of the **nginx-deployment** Deployment is running.

3. Enter the following command to view the deployed pods:

   ```
   kubectl get pods
   ```

   You should see mulitple instances of the **nginx-deployment** pod running.

Task 4: Scale the Deployment Out

1. To scale the deployment out, open a terminal and enter the following command:

   ```
   kubectl apply -f ~/course_files/CAAS101/manifests/labs/nginx-rolling_update.yaml
   ```

   You should see the deployment “nginx-deployment” was configured.
2. Enter the following command to view the deployments:

   ```
   kubectl get deployments
   ```

   You should see that four instances of the **nginx-deployment** Deployment is running again.
3. Enter the following command to view the deployed pods:

   ```
   kubectl get pods
   ```

   You should see four instances of the **nginx-deployment** pod running.

Summary:

In this exercise, you created a new manifest for an existing deployment that specified a smaller number of replicas. You then applied the updated manifest to scale in the deployment. Finally you applied the original manifest to scale the deployment back out.

(End of Exercise)

5 Monitoring a Cluster

Description:

In this section you are introduced to how to install and use the Kubernetes Monitoring utilities.

5- 1 Access cAdvisor on the Kubernetes Cluster Nodes

Description:

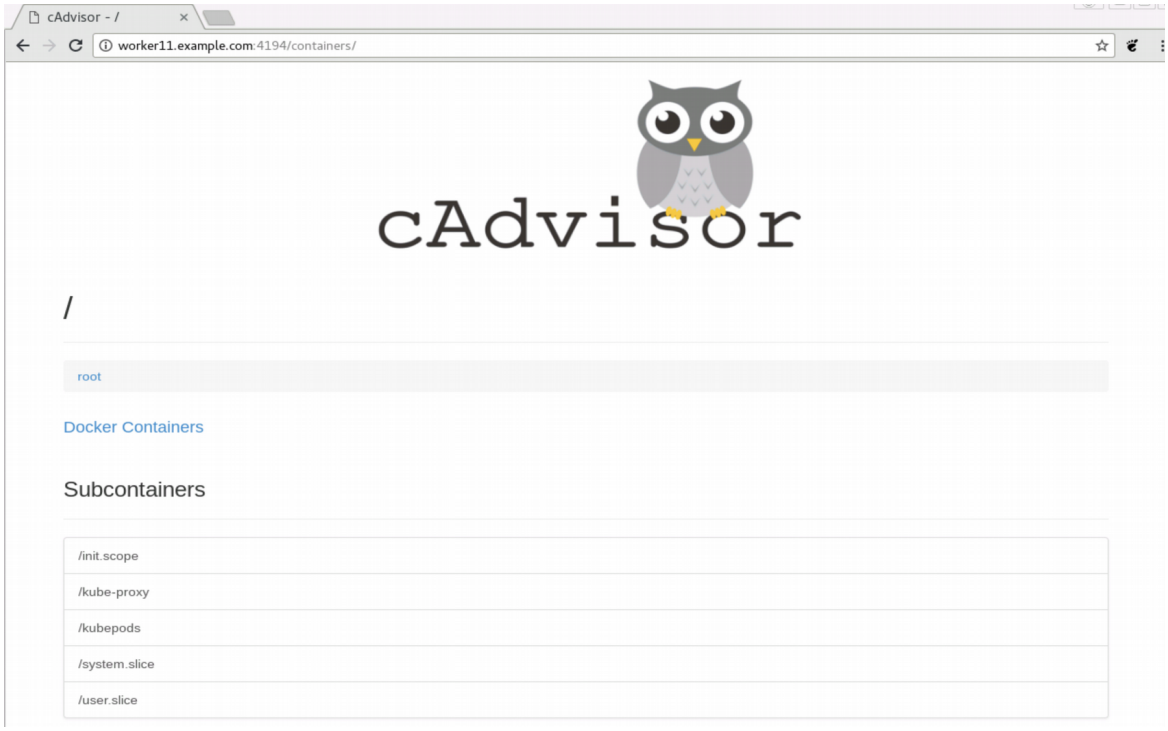
In this exercise, you access cAdvisor in the Kubernetes cluster.

Task 1: Access the cAdvisor Dashboard

1. To access the cAdvisor dashboard on the first worker node, open a web browser and point to:

   worker10.example.com:4194/containers

   You should see the cAdvisor Dashboard.



2. To view the containers that are running on this node, click on the blue **Docker Containers** link above the Subcontainers section

   You should see a list of containers running on this host. You can view more information about each container by clicking on each container entry.

3. Repeat the previous steps on each of the other worker nodes (**worker11**, **worker12** and **worker13**) as well as the master nodes (**master01**, **master02** and **master03**)

   Notice which containers are running on each node.

Summary:

In this exercise, you accessed the cAdvisor Dashboard on each of the cluster ndoes.

(End of Exercise)

6 Work With Kubernetes

Description:

In this section you are introduced to how to work with Kubernetes. You are first introduced to Kubernetes configuration and management utilities and manifests. You then deploy and manage pods on a Kubernetes cluster.

6- 1 Update a Deployment

Description:

In this exercise, you update a running pod.

Task 1: Create a New Manifest for the Deployment

1. In the text editor of your choice, open the `~/course_files/CAAS101/manifests/labs/nginx-update.yaml` file
2. Notice the section in **red**

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  replicas: 4
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: smt.example.com:5000/nginx:1.9.0
        ports:
        - containerPort: 80
```

Task 2: Update the Deployment

1. If you don't already have **nginx** deployed Deploy either through **KubeDashboard** using the file `~/course_files/CAAS101/manifests/labs/nginx-deployment.yaml` or the following command line:

   ```
   kubectl apply -f ~/course_files/CAAS101/manifests/labs/nginx-deployment.yaml
   ```

2. To display information on the current nginx deployment enter the following command:

   ```
   kubectl describe deployment -l app=nginx
   ```

   You should see the description of the nginx deployment displayed.

   Notice the image version is: nginx:1.7.9

3. Open another terminal and enter the following command to watch the running pods:

   ```
   watch kubectl get pods
   ```

You should see a list of the running pods displayed with the list updating every 2 seconds.

4. To update the deployment, open a terminal and enter the following command:

   ```
   kubectl apply -f ~/course_files/CAAS101/manifests/labs/nginx-update.yaml
   ```

   You should see the deployment “nginx-deployment” was configured.

5. Enter the following command to view the deployments:

   ```
   kubectl get deployments
   ```

   You should see that 4 instances of the **nginx-deployment** pod are DESIRED and, depending on when you ran the command, the values in the CURRENT, UP-TO-DATE and AVAILABLE columns my be more, fewer or the same number as the update happens .

6. In the terminal where you are watching the pods enter `ctrl+c` to stop the watch command

7. Enter the following command to display information about the running deployment of nginx:

   ```
   kubectl describe pods -l app=nginx
   ```

   Notice the image version is now: nginx:1.9.0

Summary:

In this exercise, you created a new manifest to update the running nginx deployment. You then updated the deployment and verified that it was updated.

(End of Exercise)

6- 2 Update a Deployment Via Rolling Updates

Description:

In this exercise, you update a running pod using rolling updates.

Task 1: Create a New Manifest for the Deployment

1. On the **management workstation**, In the text editor of your choice, open the `~/course_files/CAAS101/manifests/labs/nginx-rolling_update.yaml` file
2. Notice the changes are in **red**

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  replicas: 4
  revisionHistoryLimit: 5
  minReadySeconds: 20
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxUnavailable: 25%
      maxSurge: 2
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: smt.example.com:/5000/nginx:1.12.0
        ports:
        - containerPort: 80
```

3. Save the file and close the text editor

Task 2: Update the Deployment

1. To display information on the current nginx deployment enter the following command:

```
kubectl describe deployment -l app=nginx
```

You should see the description of the nginx deployment displayed.
Notice the image version is: **nginx:1.9.0**

2. Open another terminal and enter the following command to watch the running pods:

```
watch kubectl get pods
```

You should see a list of the running pods displayed with the list updating every 2 seconds.

3. To update the deployment, open a terminal and enter the following command:

```
kubectl apply -f ~/course_files/CAAS101/manifests/labs/nginx-rolling_update.yaml
```

You should see the deployment “nginx-deployment” was configured.

In the terminal where you are watching the pods you should see the number of running pods drop to 3 (because of maxUnavailable: 25%) and 3 new pods start deploying. Shortly after the 3 new pods are running you should see the number of running pods scale up to 6 (because of maxSurge: 2) and then back to 4.

4. Enter the following command to view the deployments:

```
kubectl get deployments
```

You should see that 4 instances of the **nginx-deployment** pod are running.

5. In the terminal where you are watching the pods enter `ctrl+c` to stop the watch command
6. Enter the following command to display information about the running deployment of nginx:

```
kubectl describe deployment -l app=nginx
```

Notice the image version is now: **nginx:1.12.0**

Summary:

In this exercise, you created a new manifest to update the running nginx deployment using the rolling update type. You then updated the deployment and verified that it was updated.

(End of Exercise)

6- 3 Expose a Service Running in a Pod

Description:

In this exercise, you expose a service running in a pod.

Task 1: Create a Manifest for the Service

1. On the **CAAS101-management VM**, in the text editor of your choice, open the file:
   `~/course_files/CAAS101/manifests/labs/nginx-service.yaml`

```
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  type: NodePort
  ports:
    - port: 80
      nodePort: 30000
  selector:
      app: nginx
```

Task 2: Define the Service

1. To define the service in the cluster, open a terminal and enter the following command:

   ```
   kubectl apply -f ~/course_files/CAAS101/manifests/labs/nginx-service.yaml
   ```

   You should see the service “nginx-service” was created.
2. Enter the following command to view the services:

   ```
   kubectl get services
   ```

   You should see that the **nginx-service** service is defined. Notice the 80:30000 under ports showing external port 30000 will be redirected into internal port 80.

Task 3: Access the Exposed Service

1. To access the exposed service, open a web browser and point to:

   http://worker.example.com:30000

   You should see the Welcome to nginx web page.

Note:

If desired, you can change the URL to point to a specific worker node (i.e. **worker10.example.com**) and see that the web page is still accessible. This demonstrates that the kube-proxy is working on all of the cluster nodes.

Summary:

In this exercise, you defined a service in the cluster exposing port 80 that allowed access to the nginx pod running on the cluster. You then accessed the nginx pod in a web browser.

(End of Exercise)

6- 4 Define Limits for Containers and Pods in Kubernetes

Description:

In this exercise, you define limits for containers and pods in the Kubernetes cluster.

Task 1: Create a New Namespace in the Cluster

1. On the **CAAS101-management VM**, at the command line, enter the following command to create a new namespace in the Kubernetes cluster:

   ```
   kubectl create namespace limit-example
   ```

   You should see that a new namespace was created.
2. Enter the following command to display the namespaces:

   ```
   kubectl get namespaces
   ```

   You should see the new namespace listed.

Task 2: Create a Manifest for the Limits

1. In the text editor open the file:

   ~/course_files/CAAS101/manifests/labs/limits.yaml

```
apiVersion: v1
kind: LimitRange
metadata:
  name: mylimits
spec:
  limits:
  - max:
      cpu: "2"
      memory: 1Gi
    min:
      cpu: 200m
      memory: 6Mi
    type: Pod
  - default:
      cpu: 300m
      memory: 200Mi
    defaultRequest:
      cpu: 200m
      memory: 100Mi
    max:
      cpu: "2"
      memory: 1Gi
    min:
      cpu: 100m
      memory: 3Mi
    type: Container
```

Task 3: Apply the Limits to the Namespace

1. To deploy the pod, open a terminal and enter the following command:

   ```
   kubectl apply -f ~/course_files/CAAS101/manifests/labs/limits.yaml
   --namespace=limit-example
   ```

   You should see the limitrange “mylimits” was created.
2. Enter the following command to display the limitranges:

   ```
   kubectl get limitranges  --namespace=limit-example
   ```

   You should see the **mylimits** limitrange listed.

Summary:

In this exercise, you created a new namespace in the Kubernetes cluster. You then defined limits for containers and pods and applied them to the new namespace.

(End of Exercise)

6- 5 Introduction to Helm

Description:

In this exercise, you are introduced to the helm utility.

Task 1: Initialize Helm

1. On the **CAAS101-management VM**, in a terminal, enter the following command to initial helm:

   ```
   helm init --client-only
   ```

   You should see that the `.helm` directory structure was created in your home directory.

Task 2: Use Some Basic Helm Commands

1. In a terminal, enter the following command to source the Helm auto-completion functions into your shell environment:

   ```
   source <(helm completion bash)
   ```

   You shell environment should now have the helm auto-completion functions available.
2. Enter the following command to display a list of available helm charts:

   ```
   helm search
   ```

   You should see a list of available helm charts listed.
3. Try running the command again but rather than typing in the entire command, only type:

   ```
   helm sea[Tab]
   ```

   (Where [Tab] is the tab key)
   Notice that the "`search`" option is auto-completed.
4. Enter the following command to search for a specific helm chart:

   ```
   helm search dokuwiki
   ```

   You should see the dokuwiki chart listed.
5. Enter the following command to display the list of currently configured repos:

   ```
   helm repo list
   ```

   You should see at least one repo listed (probably the stable repo for **https://kubernetes-charts.storage.googleapis.com**) in addition to the local repo.

Summary:

In this exercise, you initialized helm. You then ran some basic helm commands such as configuring auto-completion, searching for helm charts and displaying repositories.

(End of Exercise)

6- 6 Deploy an Application with Helm

Description:

In this exercise, you deploy an application from a helm chart using the helm command.

Task 1: Create Helm Chart Config File

1. On the **CAAS101-management VM**, in a terminal, enter the following command to search for a dokuwiki helm chart:

   ```
   helm search dokuwiki
   ```

   You should see a helm chart named stable/dokuwiki listed with its available chart version.
2. Enter the following command to view the default configuration for the dokuwiki chart:

   ```
   helm inspect stable/dokuwiki | less
   ```

   You should see the configuration displayed in the less pager. Page through the configuration to see what variables are being set.
3. In the text editor of your choice, create/open the `~/dokuwiki-config.yaml` file
4. Enter the following in the file:

   ```
   dokuwikiPassword: password123
   serviceType: NodePort
   persistence:
     enabled: false
   ```

5. Save the file and close the text editor

Task 2: Deploy the Helm Chart

1. In a terminal, enter the following command to view the current helm releases:

   ```
   helm list
   ```

   You should not see the **dokuwiki** application listed.
2. Enter the following command to deploy the chart:

   ```
   helm install -f ~/dokuwiki-config.yaml --name mywiki stable/dokuwiki
   ```

   You should see the chart was deployed.

   In the output of the chart deployment, in the **==> v1/Service** section, notice the IP and port(s) the application is listening on. The NodePort(s) that the application is listening on are the number after the colon (**:**) in the **PORT(S)** column.

   Example: **80:32313/TCP,443:31034/TCP**

In this example the NodePorts are **32313** for http and **31034** for https.

Record the http NodePort here:

DOKUWIKI_PORT=_____

3. Enter the following command to display the current helm releases:

```
helm list
```

You should now see the dokuwiki chart listed with a name of **mywiki**. Notice the REVISION column shows the number **1** as this is the initial deployment of this release.

4. Enter the following command to view the release history for the application:

```
helm history mywiki
```

You should see that only a single release of **mywiki** exists.

5. Enter the following command to view the status of the **mywiki** release:

```
helm status mywiki
```

You should see output similar to what was displayed when the chart was first deployed.

6. You can also enter the following `kubectl` commands:

```
kubectl get deployments
kubectl get pods
kubectl get services
```

Notice that you see that same info about the deployed deployments/pods/services as if you were to have deployed them from manifests.

Task 3: Access the Application Deployed by the Helm Chart

1. Open a web browser and point to:

http://worker.example.com:*DOKUWIKI_PORT*

You should see the My Wiki page displayed.

2. On the top right of the page click: **Login**
3. Enter the following credentials:

Username: **user**
Password: **password123**

You should be logged in.

After logging in you probably see a number of warnings of available hotfixes and/or updates.

Task 4: Update the Application Release

1. On the CAAS101-management VM, in a terminal, make a copy of the chart config file you created at the beginning of this exercise:

```
cp ~/dokuwiki-config.yaml ~/dokuwiki-config-update.yaml
```

2. In the text editor of your choice, open the `~/dokuwiki-config-update.yaml` file
3. Edit the file to match the following (changes are in **red**):

```
image: bitnami/dokuwiki:latest
dokuwikiPassword: password123
serviceType: NodePort
persistence:
  enabled: false
```

4. Save the file and close the text editor
5. Enter the following command to upgrade the mywiki release:

```
helm upgrade -f ~/dokuwiki-config-update.yaml mywiki stable/dokuwiki
```

   You should see the chart was successfully deployed.
6. Enter the following command to view the current helm releases:

```
helm list
```

   You should see the dokuwiki chart named **mywiki** listed. Notice the REVISION column now shows the number **2** as the release has been updated once.
7. Enter the following command to display the release history for the mywiki release:

```
helm history mywiki
```

   Notice that there are 2 revisions. Also notice the the first revision's STATUS is **SUPERSEDED** and the second revision's STATUS is **DEPLOYED**.
8. In the web browser, refresh the web page (or log back in if you have logged out)
   You should no longer see the warning messages about available hotfixes and/or updates.

Task 5: Delete a Deployed Helm Chart Release

1. In a terminal, enter the following command to delete the mywiki release:

```
helm delete mywiki
```

   You should see that the release was deleted.
2. Enter the following command to list the current helm releases:

```
helm list
```

   You should no longer see the dokuwiki chart named **mywiki** displayed.
3. Enter the following command to display the status of the mywiki release:

```
helm status mywiki
```

   Notice that the STATUS is **DELETED** but also that it remembered that it had been deployed as it has a date listed in LAST DEPLOYED.

Summary:

In this exercise, you first deployed a helm chart. You then accessed the application that was deployed, Next you updated the release, verifying is was updated. Finally you deleted the release.

(End of Exercise)

7 Storage

Description:

How to handle persistent storage in pods.

7- 1 Configure NFS Persistent Storage

Description:

In this exercise, you configure a persistent volume on an NFS server. You then create a pod that updates a file on the persistent volume and a pod that exports the file via http

Task 1: Create the Persistent Volume on the NFS Server

1. On the **SMT server**, as the **root** user with the password of **Linux** enter the following commands in a terminal session to create the persistent volume and `index.html` file:

   ```
   mkdir -p /export/vol-01
   touch /export/vol-01/index.html
   ```

   The directory and file should now exist.

Task 2: Create the Manifest for the Persistent Volume

2. On the **CAAS101-management VM,** in the text editor of your choice, open the file:

   `~/course_files/CAAS101/manifests/labs/pv/nfs-pv-vol-01.yaml`

   Enter the following in the file:

```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: nfs-vol-01
spec:
  capacity:
    storage: 1Mi
  accessModes:
    - ReadWriteMany
  nfs:
    # FIXME: use the right IP
    server: 192.168.110.2
    path: "/export/vol-01"
```

Task 3: Create the Manifest for the Persistent Volume Claim

1. In the text editor of your choice, open the file:

   `~/course_files/CAAS101/manifests/labs/pv/nfs-pvc-vol-01.yaml`

```yaml
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: nfs-vol-01
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 1Mi
```

Task 4: Create the Manifest for the Busybox Instance

1. In the text editor of your choice, open the file:
   `~/course_files/CAAS101/manifests/labs/pv/nfs-busybox-deployment.yaml`

```yaml
# This mounts the nfs volume claim into /mnt and continuously
# overwrites /mnt/index.html with the time and hostname of the pod.

apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: nfs-busybox
spec:
  replicas: 1
  template:
    metadata:
      labels:
        name: nfs-busybox
    spec:
      containers:
      - image: smt.example.com:5000/busybox
        command:
          - sh
          - -c
          - 'while true; do date > /mnt/index.html; hostname >> /mnt/index.html; sleep $(($RANDOM % 5 + 5)); done'
        imagePullPolicy: IfNotPresent
        name: busybox
        volumeMounts:
          # name must match the volume name below
          - name: nfs-vol-01
            mountPath: "/mnt"
      volumes:
      - name: nfs-vol-01
        persistentVolumeClaim:
          claimName: nfs-vol-01
```

Task 5: Create the Manifests for the Web Frontend

1. In the text editor of your choice, create/open the file:
   `~/course_files/CAAS101/manifests/labs/pv/nfs-web-deployment.yaml`

```
# This pod mounts the nfs volume claim into /usr/share/nginx/html and
# serves a simple web page.

apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: nfs-web
spec:
  replicas: 1
  template:
    metadata:
      labels:
        app: web-frontend
    spec:
      containers:
      - name: web
        image: smt.example.com:5000/nginx
        ports:
          - name: web
            containerPort: 80
        volumeMounts:
            # name must match the volume name below
            - name: nfs-vol-01
              mountPath: "/usr/share/nginx/html"
      volumes:
      - name: nfs-vol-01
        persistentVolumeClaim:
          claimName: nfs-vol-01
```

2. Next, open the file:
   `~/course_files/CAAS101/manifests/labs/pv/nfs-web-service.yaml`

```
apiVersion: v1
kind: Service
metadata:
  name: nfs-web
spec:
  type: NodePort
  ports:
    - port: 80
      nodePort: 30080
  selector:
    app: web-frontend
```

Task 6: Deploy the Objects

1. To deploy the volumes/pods/service, open a terminal and enter the following command:

   ```
   kubectl apply -f ~/course_files/CAAS101/manifests/labs/pv
   ```

   You should see the following were created (not necessarily in this order):

deployment “nfs-busybox”
persistentvolume “nfs-vol-01”
persistenvolumeclaim “nfs-vol-01”
deployment “nfs-web”
deployment “nfs-web”

2. Enter the following command to view the deployments:

   ```
   kubectl get deployments
   ```

   You should see the **nfs-busybox** and **nfs-web** deployments listed.

3. Enter the following command to view the pods:

   ```
   watch kubectl get pods
   ```

   You should see the pods for the **nfs-busybox** and **nfs-web** deployments listed.

4. Enter the following command to view the persistent volumes:

   ```
   kubectl get pv
   ```

   You should see the persistent volume **nfs-vol-01** listed.

5. Enter the following command to view the persistent volumes:

   ```
   kubectl get pvc
   ```

   You should see the persistent volume claim **nfs-vol-01** listed.

Task 7: Test the Persistent Data

1. On the **SMT server**, enter the following command to view the content of the `index.html` file on the NFS server:

   ```
   cat /export/vol-01/index.html
   ```

   You should see a line with the date and time and a line with the pod name of the **nfs-busybox** pod. If you rerun the command you should see that the time stamp is updating.

2. On the **management workstation**, open a web browser and point to:

   http://worker10.example.com:30080

   You should see the content of the `index.html` file. When you refresh the page you should see the time stamp updating. (Also notice that if you change the URL to the different worker nodes you will see the same thing.)

Task 8: Remove the Objects from the Cluster

1. In the first terminal, enter the following commands to delete all of the objects:

   ```
   kubectl delete service nfs-web
   kubectl delete deployment nfs-web
   ```

```
kubectl delete deployment nfs-busybox
kubectl delete pv nfs-vol-01
kubectl delete pvc nfs-vol-01
```

You should see that the objects were deleted.

Summary:

In this exercise, you created manifests for a persistent volume, a persistent volume claim, a pod to that attached to the volume and writes data to an index.html file in the volume, and a web server that attaches to the volume and displays the index.html file. You then verified that the index.html file was being updated by looking at the file both on the NFS volume and the web server.

(End of Exercise)

7- 2 Configure Persistent Storage with a NFS StorageClass

Description:

In this exercise, you define a StorageClass that will automatically create volumes on an NFS server. You then create a volume claim that will cause the volume to be created on the NFS server and a pod that will write a file in the volume.

Task 1: Create the Directory for the Persistent Volumes on the NFS Server

1. On the **SMT server**, as the **root** user with a password of **Linux**, enter the following commands to create the directory that will contain the volumes:

```
mkdir -p /export/volumes
chmod 777 /export/volumes
```

The directory and file should now exist.

Task 2: Define a Service Account, Cluster Role and Cluster Role Binding

1. On the **CAAS101-management VM,** in the text editor of your choice, open the file: `~/course_files/CAAS101/manifests/labs/nfs-client-storageclass/auth/serviceaccount.yaml`

```
apiVersion: v1
kind: Service
metadata:
  name: nfs-web
spec:
  type: NodePort
  ports:
    - port: 80
      nodePort: 30080
  selector:
    app: web-frontend
```

2. In the text editor of your choice, open the file:
3. `~/course_files/CAAS101/manifests/labs/nfs-client-storageclass/auth/clusterrole.yaml`

```
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1beta1
metadata:
  name: nfs-client-provisioner-runner
rules:
  - apiGroups: [""]
    resources: ["persistentvolumes"]
    verbs: ["get", "list", "watch", "create", "delete"]
  - apiGroups: [""]
    resources: ["persistentvolumeclaims"]
    verbs: ["get", "list", "watch", "update"]
  - apiGroups: ["storage.k8s.io"]
    resources: ["storageclasses"]
    verbs: ["get", "list", "watch"]
  - apiGroups: [""]
    resources: ["events"]
    verbs: ["list", "watch", "create", "update", "patch"]
```

4. In the text editor of your choice, open the file:
`~/course_files/CAAS101/manifests/labs/nfs-client-storageclass/auth/clusterrolebinding.yaml`

```
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1beta1
metadata:
  name: run-nfs-client-provisioner
subjects:
  - kind: ServiceAccount
    name: nfs-client-provisioner
    namespace: default
roleRef:
  kind: ClusterRole
  name: nfs-client-provisioner-runner
  apiGroup: rbac.authorization.k8s.io
```

5. Enter the following command to open the service, clusterrole and clusterrolebinding:

```
kubectl apply -f ~/course_files/CAAS101/manifests/labs/nfs-client-storageclass/auth
```

You should see the following were created (not necessarily in this order):

serviceaccount “nfs-client-provisioner”
clusterrole “nfs-client-provisioner-runner”
clusterrolebinding “run-nfs-client-provisioner”

6. Enter the following command to view the service accounts:

   ```
   kubectl get serviceaccounts
   ```

   You should see **nfs-client-provisioner** listed.

7. Enter the following command to view the cluster roles:

   ```
   kubectl get clusterroles
   ```

   You should see **nfs-client-provisioner-runner** listed.

Task 3: Define a Storage Class

1. On the **CAAS101-management VM**, in the text editor of your choice, open the file:

   ```
   ~/course_files/CAAS101/manifests/labs/nfs-client-storageclass/deploy/class.yaml
   ```

   ```
   apiVersion: v1beta1
   kind: ServiceAccount
   metadata:
     name: nfs-client-provisioner
   ```

2. In the text editor of your choice, open the file:

   ```
   ~/course_files/CAAS101/manifests/labs/nfs-client-storageclass/deploy/deployment.yaml
   ```

```
kind: Deployment
apiVersion: extensions/v1beta1
metadata:
  name: nfs-client-provisioner
spec:
  replicas: 1
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        app: nfs-client-provisioner
    spec:
      containers:
        - name: nfs-client-provisioner
          image: smt.example.com:5000/quay.io/external_storage/nfs-client-provisioner:latest
          volumeMounts:
            - name: nfs-client-root
              mountPath: /persistentvolumes
          env:
            - name: PROVISIONER_NAME
              value: smt.example.com
            - name: NFS_SERVER
              value: 192.168.110.2
            - name: NFS_PATH
              value: /export/volumes
      serviceAccount: nfs-client-provisioner
      serviceAccountName: nfs-client-provisioner
      volumes:
        - name: nfs-client-root
          nfs:
            server: 192.168.110.2
            path: /export/volumes
```

3. Enter the following command to create the storage class and provisioner:

   kubectl apply -f ~/course_files/CAAS101/manifests/labs/nfs-client-storageclass/deploy

   You should see the following were created (not necessarily in this order):

   storageclass “managed-nfs-storage”
   deployment “nfs-client-provisioner”

4. Enter the following command to view the service accounts:

   kubectl get storageclasses

   You should see **managed-nfs-storage** listed.

5. Enter the following command to view the cluster roles:

   kubectl get deployments

   You should see **nfs-client-provisioner** listed.

Task 4: Create a Persistent Volume Claim and Pod that Uses It

6. On the **CAAS101-management VM**, in the text editor of your choice, open the file:

   ~/course_files/CAAS101/manifests/labs/nfs-client-storageclass/test/test-claim.yaml

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: test-claim
  annotations:
    volume.beta.kubernetes.io/storage-class: "managed-nfs-storage"
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 1Mi
```

7. In the text editor of your choice, open the file:

   `~/course_files/CAAS101/manifests/labs/nfs-client-storageclass/test/test-pod.yaml`

```
kind: Pod
apiVersion: v1
metadata:
  name: test-pod
spec:
  containers:
  - name: test-pod
    image: smt.example.com:5000/gcr.io/google_containers/busybox:1.24
    command:
      - "/bin/sh"
    args:
      - "-c"
      - "touch /mnt/SUCCESS && exit 0 || exit 1"
    volumeMounts:
      - name: nfs-pvc
        mountPath: "/mnt"
  restartPolicy: "Never"
  volumes:
    - name: nfs-pvc
      persistentVolumeClaim:
        claimName: test-claim
```

8. Enter the following command to create the volume claim and the pod:

   `kubectl apply -f ~/course_files/CAAS101/manifests/labs/nfs-client-storageclass/test`

   You should see the following were created (not necessarily in this order):

   persistentvolumeclaim “test-claim”
   pod “test-pod”

9. Enter the following command to view the persistent volume claims:

```
kubectl get pvc
```

You should see **test-claim** listed.

10. Enter the following command to view the details about the volume claim:

```
kubectl describe pvc test-claim
```

Under the **Events** section you should see that the provisioner successfully provisioned the volume followed by the name of the volume. Record the name of the volume here:

NFS_VOLUME_NAME=_____

This will be in the directory name you will see on the NFS server

11. Enter the following command to view the pods:

```
kubectl get pods
```

You should note see **test-pod** listed because it was defined to deploy, run a command to write a file to the volume and then exit.

12. On the **SMT server**, enter the following command to view the directory that was created for the volume:

```
ls -l /export/volumes/
```

You should see a directory named:

default-test-claim-*NFS_VOLUME_NAME*

13. Enter the following command:

```
ls -l /export/volumes/default-test-claim-NFS_VOLUME_NAME
```

You should see a file named: **SUCCESS**

Task 5: Create a Persistent Volume Claim and Pod that Uses It

14. On the **CAAS101-management VM**, enter the following command to remove the volume claim:

```
kubectl delete pvc test-claim
```

Yous should see that the persistentvolumeclaim was deleted.

15. Enter the following command to view the persistent volume claims:

```
kubectl get pvc
```

You should no longer see **test-claim** listed.

16. On the **SMT server**, enter the following command to view the directory that was created for the volume:

```
ls -l /export/volumes/
```

You should see that the directory for the volume has been renamed to:

`archived-default-test-claim-`***NFS_VOLUME_NAME***

If desired, you could apply the manifests in the test directory again to see another directory created.

Task 6: Remove the Objects from the Cluster

1. On the **CAAS101-management VM,** in the first terminal, enter the following commands to delete all of the objects:

```
kubectl delete -f ~/course_files/CAAS101/manifests/labs/nfs-client-storageclass/test
kubectl delete -f ~/course_files/CAAS101/manifests/labs/nfs-client-storageclass/deploy
kubectl delete -f ~/course_files/CAAS101/manifests/labs/nfs-client-storageclass/auth
```

You should see that the objects were deleted.

Summary:

In this exercise, you created manifests for a service account, a role, a role binding, a storageclass, a persistent volume claim, a pod to that attached to the volume and writes data to a file in the volume. You then verified that the volume was being created and the file in it was created on the NFS server.

(End of Exercise)

8 Horizontal Scale

Description:

In this section you are introduces to Microservices, Containers and Container Orchestration.

8- 1 Configure Horizontal Pod Autoscaling

Description:

In this exercise, you configure and then test horizontal autoscaling of pods in a deployment.

Note :

If desired, there is a script in `~/course_files/CAAS101/scripts/` named `tmux-watch-kubectl.pod-deployment-hpa.sh` that uses `tmux` to split your terminal into 4 quadrants and then runs the three watch commands specified in Task 5 in three of the panes leaving one pane to enter the other commands.

To switch between panes in the `tmux` terminal enter `Ctrl+a` followed by one of the directional arrow keys.

To close out the panes in the `tmux` terminal enter the following key strokes in each pane until all panes are closed: `Ctrl+c Ctrl+d`

(`killall tmux` will work as well.)

Task 1: Create the Manifest for the App to be Scaled

1. In the text editor of your choice, open the file:

   `~/course_files/CAAS101/manifests/labs/hpa/app/php-apache.yaml`

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  labels:
    run: php-apache
  name: php-apache
spec:
  replicas: 1
  selector:
    matchLabels:
      run: php-apache
  strategy:
    rollingUpdate:
      maxSurge: 1
      maxUnavailable: 1
    type: RollingUpdate
  template:
    metadata:
      labels:
        run: php-apache
    spec:
      containers:
      - image: smt.example.com:5000/gcr.io/google_containers/hpa-example
        name: php-apache
        ports:
        - containerPort: 80
          protocol: TCP
        resources:
          requests:
            cpu: 200m
      dnsPolicy: ClusterFirst
      restartPolicy: Always
      schedulerName: default-scheduler
      securityContext: {}
      terminationGracePeriodSeconds: 30
```

Task 2: Open the Manifest for the App's Service

1. In the text editor of your choice, open the file:
   `~/course_files/CAAS101/manifests/labs/hpa/app/php-apache-service.yaml`

```yaml
apiVersion: v1
kind: Service
metadata:
  name: php-apache
spec:
  clusterIP: 172.24.253.251
  ports:
  - port: 80
    protocol: TCP
    targetPort: 80
  selector:
    run: php-apache
  type: ClusterIP
```

Task 3: Open the Manifest for the Autoscaler

1. In the text editor of your choice, open the file:
   `~/course_files/CAAS101/manifests/labs/hpa/app/php-apache-autoscaler.yaml`

```yaml
apiVersion: autoscaling/v1
kind: HorizontalPodAutoscaler
metadata:
  name: php-apache
spec:
  scaleTargetRef:
    apiVersion: apps/v1beta1
    kind: Deployment
    name: php-apache
  minReplicas: 1
  maxReplicas: 10
  targetCPUUtilizationPercentage: 50
```

Task 4: Open the Manifest for the Load Generator

1. In the text editor of your choice, open the file:

   `~/course_files/CAAS101/manifests/labs/hpa/load/load-generator.yaml`

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: load-generator
spec:
  replicas: 1
  strategy:
    rollingUpdate:
      maxSurge: 1
      maxUnavailable: 1
    type: RollingUpdate
  template:
    metadata:
      labels:
        name: load-generator
    spec:
      containers:
      - image: smt.example.com:5000/busybox
        name: busybox
        command:
          - sh
          - -c
          - 'while true; do wget -q -O- http://php-apache.default.svc.cluster.local; done'
```

Task 5: Deploy the AutoScaler Objects

1. To deploy the pods/services, open a terminal and enter the following command:

   ```
   kubectl apply -f ~/course_files/CAAS101/manifests/labs/hpa/app
   ```

   You should see the following were created (not necessarily in this order):

   deployment “php-apache”
   service “php-apache”
   horizontalpodautoscaler “php-apache”

2. Open a second terminal and enter the following command to view the deployments:

   ```
   watch kubectl get deployments
   ```

   You should see the **php-apache** deployment listed.
   The output should refresh every 2 seconds.

3. Open a third terminal and enter the following command to view the pods:

   ```
   watch kubectl get pods
   ```

   You should see the pods for the **php-apache** pod listed.
   The output should refresh every 2 seconds.

4. Open a fourth terminal and enter the following command to view the autoscalers:

   ```
   kubectl get hpa
   ```

   You should see the autoscaler **php-apache** listed. Notice the percentages in the TARGETS column. (It may take a minute or few for the percentage to show up on the left side of the `/` in the TARGETS column so be patient.)

5. Enter the following command to view the details of the autoscaler:

```
kubectl describe hpa php-apache
```

You should see details about the autoscaler displayed.

6. Run the watch command for the autoscaler as well:

```
watch kubectl get hpa
```

The output should refresh every 2 seconds.

Task 6: Cause the Deployment To Scale Out

1. In the first terminal (the one that is not watching the other commands), enter the following command to deploy the **load-generator** deployment:

```
kubectl apply -f ~/course_files/CAAS101/manifests/labs/hpa/load
```

You should see the following was created:

deployment “load-generator”

2. Look at the terminal that is watching the `kubectl get hpa` command
   Notice that after a short while, in the TARGETS column, the percentages start to rise.
3. Look at the terminal that is watching the `kubectl get pods` command
   Notice that when the percentages exceed 50% in the other window that new pods are created to handle the load.
4. Look at the terminal that is watching the `kubectl get deployments` command
   Notice that when the percentages exceed 50% in the other window and the additional pods are created that the numbers here are adjusted accordingly. If the scale out goes on long enough notice that the numbers do not exceed the MAXPODS value defined in the autoscaler.

Task 7: Cause the Deployment To Scale Back

1. In the first terminal (the one that is not watching the other commands), enter the following command to delete the **load-generator** deployment:

```
kubectl delete -f ~/course_files/CAAS101/manifests/labs/hpa/load
```

2. Watch the other terminal windows
   After a short while you should see the percentages drop (you may see them increase before they drop due to a lag in the autoscaler getting data from the monitoring).
   A short while after the percentages drop you should see the number of **php-apache** pods decrease.

Task 8: Clean Up the Deployments

1. In the first terminal, enter the following commands to delete all of the objects:

```
kubectl delete deployment load-generator
kubectl delete hpa php-apache
```

```
kubectl delete service php-apache
kubectl delete deployment php-apache
```

In the terminals watching the other commands you should see the objects being removed.

2. Stop all of the watch commands in the other terminals by selecting the terminal and entering: `Ctrl+c`

Summary:

In this exercise, you created manifests for an application, a service to export the application, and other application use to generate load and an autoscaler to scale the application. You then deployed the objects and started generating load. As the load increased the application was scaled out. Finally you scaled back the application and removed the objects.

(End of Exercise)

9 Deploy an Online shop

Description:

In this section you will deploy a full online shop that is broken down into several Microservices

9- 1 Install sock shop microservices demo in a Kubernetes Cluster

---

Description:

In this exercise, you deploy the Sock Shop Microservices Demo

---

Task 1: Create a new namespace for our demo

1. From a **Terminal**

   ```
   kubectl create namespace sock-shop
   ```

Task 2: Watch Pods as the launch

1. Open a new **Terminal** to watch the pods launch

   ```
   watch kubectl -n sock-shop get pods
   ```

Task 3: Install Sock Shop

1. In the text editor of your open the file:

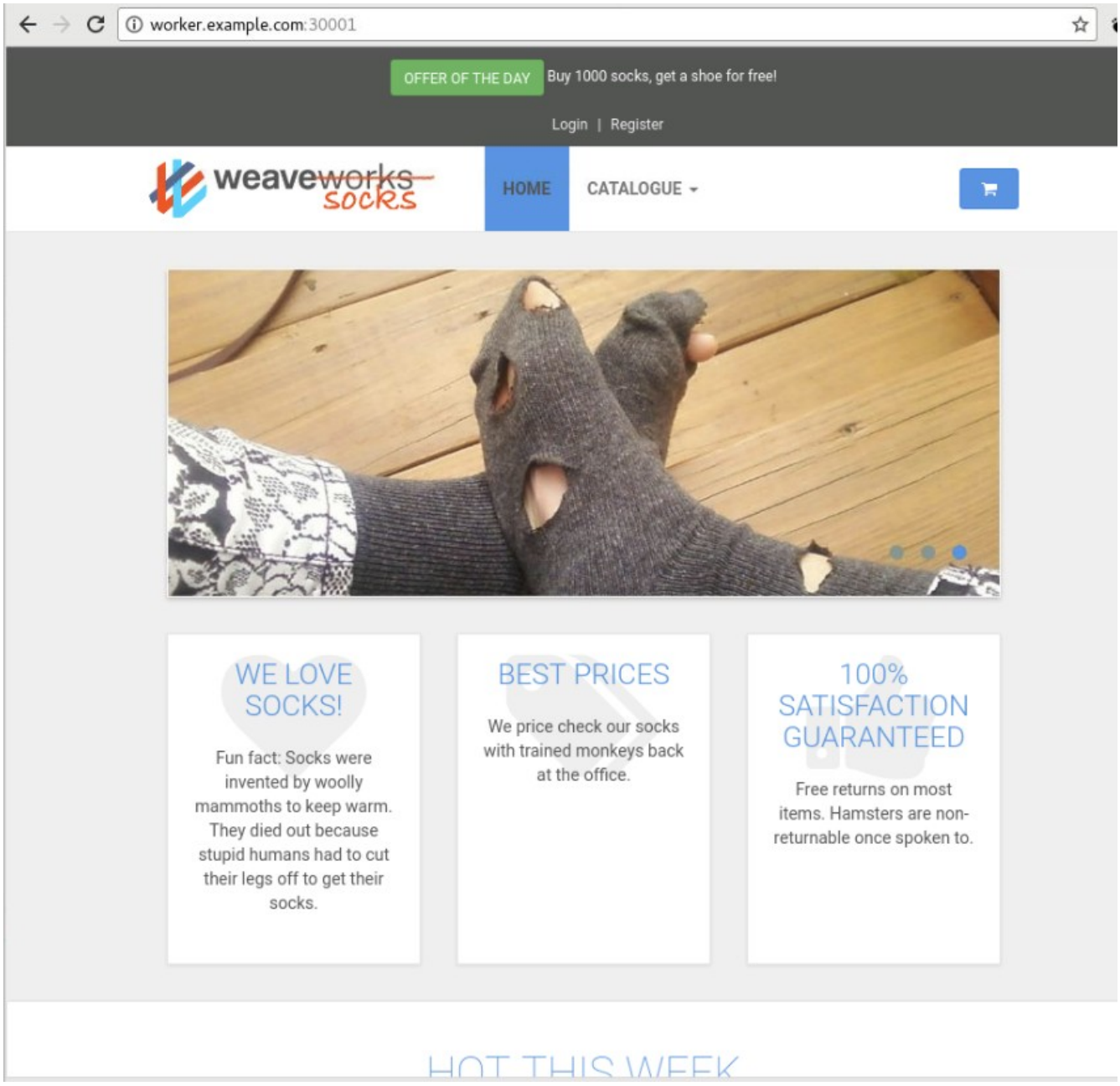
   ```
   ~/course_files/CAAS101/microservices_demo/complete-demo.yaml
   ```

2. Notice how the yaml file defines the Deployment and then defines the Service that the deployment will use.
3. `kubectl apply - f ~/course_files/CAAS101/microservices_demo/complete-demo.yaml`

Task 4: Visit sock shop website

1. On the **Management Workstation** open a web browser and point to: **`worker.example.com:30001`**

   ***Note** it will take a minutre of 2 for the Website to come up*



Summary:

(End of Exercise)

10 Appendix: Bonus Labs

Description:

Bonus Labs you can play with if you have time

10- 1 Deploy a Stateful App with Volume Storage on Kubernetes

---

Description:

In this exercise, you deploy the MySQL database server as a stateful app with a storage volume on the Kubernetes cluster.

Note:

If you do not wish to type in the manifest you are instructed to create, it has been pre-created and can be found in the `~/course_files/`***COURSE_ID***`/manifests/labs/` directory.

Task 1: Create a Manifest for the Deployment

1. On the **management workstation**, in the text editor of your choice, create/open the file:

   `~/mysql-deployment.yaml`

2. Enter the following in the file:

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: mysql
spec:
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        app: mysql
    spec:
      containers:
      - image: mysql:5.6
        name: mysql
        env:
          # Use secret in real usage
        - name: MYSQL_ROOT_PASSWORD
          value: password
        ports:
        - containerPort: 3306
          name: mysql
        volumeMounts:
        - name: mysql-persistent-storage
          mountPath: /var/lib/mysql
      volumes:
```

```
  - name: mysql-persistent-storage
    emptyDir: {}
```

3. Save the file and close the text editor

Task 2: Deploy the Pod

1. To deploy the pod, open a terminal and enter the following command:

   ```
   kubectl apply -f ~/mysql-deployment.yaml
   ```

   You should see the deployment “mysql” was created.
2. Enter the following command to view the deployments:

   ```
   kubectl get deployments
   ```

   You should see that a single instance of the **mysql** pod is running.
3. Enter the following command to view the deployed pods:

   ```
   kubectl get pods
   ```

   You should see a single instance of the **mysql** pod running.
4. Enter the following command to view details about the deployment:

   ```
   kubectl describe deployment -l app=mysql
   ```

   You should see the details of the **mysql** deployment.

   Notice the Environment shows the environment variable that was defined and the Mounts section shows the volume that was defined.

Summary:

In this exercise, you deployed MySQL with a storage volume.

(End of Exercise)

10- 2 Use Environment Variables in a Pod

Description:

In this exercise, you deploy ta simple container setting an environment variable in the container in the process.

Note:

If you do not wish to type in the manifest you are instructed to create, it has been pre-created and can be found in the `~/course_files/`***COURSE_ID***`/manifests/labs/` directory.

Task 1: Create a Manifest for the Deployment

1. On the **management workstation**, in the text editor of your choice, create/open the file:

   `~/envvars.yaml`

2. Enter the following in the file:

```
apiVersion: v1
kind: Pod
metadata:
  name: envar-demo
  labels:
    purpose: demonstrate-envars
spec:
  containers:
  - name: envar-demo-container
    image: gcr.io/google-samples/node-hello:1.0
    env:
    - name: DEMO_GREETING
      value: “SUSE Rocks!”
```

3. Save the file and close the text editor

Task 2: Deploy the Pod

1. To deploy the pod, open a terminal and enter the following command:

   `kubectl apply -f ~/envvars.yaml`

   You should see the deployment “envar-demo” was created.

2. Enter the following command to view the deployed pods:

   `kubectl get pods`

You should see a single instance of the **envar-demo** pod running.

Task 3: Display the Environment Variable in the Container

1. To enter the pod, open a terminal and enter the following command:

   ```
   kubectl exec -it envar-demo -- /bin/bash
   ```

   You should now be at a bash prompt in the container.
2. Enter the following command to display the environment variables:

   ```
   printenv
   ```

   You should see the environment variables that are set in the container. In this list you should see the DEMO_GREETING variable is set to “SUSE Rocks!”.
3. Enter the following command to exit the container:

   ```
   exit
   ```

   You should be back on the management workstation.

Task 4: Delete the Pod

1. Enter the following command to delete the pod:

   ```
   kubectl delete pod envar-demo
   ```

   You should see the pod **envar-demo** was deleted.

Summary:

In this exercise, you launched a simple container setting an environment variable in the container in the process. You then launched a bash shell inside the container and displayed the environment variable that was set.

(End of Exercise)

10- 3 Define and Access a Secret in Kubernetes

Description:

In this exercise, you define a secret in the Kubernetes cluster.

Note:

If you do not wish to type in the manifest you are instructed to create, it has been pre-created and can be found in the `~/course_files/`***COURSE_ID***`/manifests/labs/` directory.

Task 1: Create a Manifest for the Deployment

1. On the **management workstation**, at the command line, enter the following command to base64 encode a username:

   ```
   echo "tux" | base64
   ```

   Record this value here:
   SECRET_USERNAME=_____

2. Enter the following command to base64 encode a password:

   ```
   echo -n "suse1234" | base64
   ```

   Record this value here:
   SECRET_PASSWORD=_____

3. In the text editor of your choice, create/open the file:

   `~/secret.yaml`

4. Enter the following in the file:

   ```
   apiVersion: v1
   kind: Secret
   metadata:
     name: mysecret
   type: Opaque
   data:
     username: SECRET_USERNAME
     password: SECRET_PASSWORD
   ```

5. Save the file and close the text editor

Task 2: Define the Secret

1. To deploy the pod, open a terminal and enter the following command:

```
kubectl apply –f ~/secret.yaml
```

You should see the secret “mysecret” was created.

Task 3: Access the Secret

1. To access the newly created secret enter the following command:

   ```
   kubectl get secret mysecret –o yaml
   ```

   You should see the contents of **mysecret** displayed in yaml format. You could use the `base64 ––decode` command to decode the username and password displayed in the yaml output.

Summary:

In this exercise, you defined a secret in the cluster. You then accessed the secret using the `kubectl` command.

(End of Exercise)

10- 4 Deploy an Invalid Pod on Kubernetes

Description:

In this exercise, you attempt to deploy a pod with a cpu value that exceeds the limits applied to a namespace on the Kubernetes cluster.

Note:

If you do not wish to type in the manifest you are instructed to create, it has been pre-created and can be found in the `~/course_files/`***COURSE_ID***`/manifests/labs/` directory.

Task 1: Create a Manifest for the Pod

1. On the **management workstation**, in the text editor of your choice, create/open the file:

   `~/invalidpod.yaml`

2. Enter the following in the file:

```
apiVersion: v1
kind: Pod
metadata:
  name: invalid-pod
spec:
  containers:
  - name: nginx-bad
    image: nginx
    resources:
      limits:
        cpu: "3"
        memory: 100Mi
```

3. Save the file and close the text editor

Task 2: Deploy the Pod

1. To deploy the pod, open a terminal and enter the following command:

   `kubectl apply -f ~/invalidpod.yaml --namespace=limit-example`

   You should see the pod “invalid-pod” was not created because it tried to exceed the maximum CPU limit of 2.

2. Enter the following command to view the deployed pods:

   `kubectl get pods`

You should see that the **invalid-pod** pod is not running.

Summary:

In this exercise, you defined a pod with a cpu value that exceeded the limits applied to a namespace and then tried to deploy the pod on the cluster.

(End of Exercise)

10- 5 Delete a Namespace from a Kubernetes Cluster

Description:

In this exercise, you delete a namespace from the Kubernetes cluster.

Task 1: Delete a Namespace from the Cluster

1. On the **management workstation**, at the command line, enter the following command to delete a namespace from the Kubernetes cluster:

   ```
   kubectl delete namespace limit-example
   ```

   You should see that the namespace was deleted.
2. Enter the following command to display the namespaces:

   ```
   kubectl get namespaces
   ```

   You should no longer see the namespace listed.

Summary:

In this exercise, you delete a namespace from the Kubernetes cluster.

(End of Exercise)

Lab Variables:

CAASP_ADMIN_MAC=____52:54:00:ca:99:01_____

CAASP_MASTER01_MAC=____52:54:00:ca:07:01_____

CAASP_MASTER02_MAC=____52:54:00:ca:08:01_____

CAASP_MASTER03_MAC=____52:54:00:ca:09:01_____

CAASP_WORKER10_MAC=____52:54:00:ca:10:01_____

CAASP_WORKER11_MAC=____52:54:00:ca:11:01_____

CAASP_WORKER12_MAC=____52:54:00:ca:12:01_____

CAASP_WORKER13_MAC=____52:54:00:ca:13:01_____

CAASP_AUTOYAST_URL=____http://192.168.110.99/autoyast_____

CAASP_ADMIN_IP=____192.168.110.99_____

CAASP_MASTER01_IP=____192.168.110.7_____

CAASP_MASTER02_IP=____192.168.110.8_____

CAASP_MASTER03_IP=____192.168.110.9_____

CAASP_WORKER10_IP=____192.168.110.10_____

CAASP_WORKER11_IP=____192.168.110.11_____

CAASP_WORKER12_IP=____192.168.110.12_____

CAASP_WORKER13_IP=____192.168.110.13_____