

# **Technical Workshop - SUSE CaaS Platform v4 Beta**

## **-Workbook-**

**Course ID: Technical Workshop - SUSE Container as a Platform**

**Version: 4 Beta**

**Date: 2019-8-15**



## **Proprietary Statement**

Copyright © 2018 SUSE LLC. All rights reserved.

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

No part of this publication may be reproduced, photocopied, stored on a retrieval system, or transmitted without the express written consent of the publisher.

SUSE  
Maxfeldstrasse 5  
90409 Nuremberg  
Germany  
[www.suse.com](http://www.suse.com)

(C) 2018 SUSE LLC. All Rights Reserved. SUSE and the SUSE logo are registered trademarks of SUSE LLC in the United States and other countries. All third-party trademarks are the property of their respective owners.

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
<b>Section 1 : Demonstration of SUSE CaaS Platform.....</b>	<b>9</b>
(No Exercises).....	10
Exercise 1 : Start the Lab VMs.....	11
Task 1: Start the Lab VMs.....	11
Exercise 2 : Install a CaaS Platform Admin Node (Needs Updated).....	13
Task 1: Install the Admin Node.....	13
Task 2: Configure the Admin Node.....	15
Exercise 3 : Install a CaaS Platform Nodes.....	18
Task 1: Deploy the Kubernetes Dashboard.....	18
Task 2: Test to ensure SSH keys are properly set.....	19
Exercise 4 : Setup SSH Keys.....	21
Task 1: Deploy the Kubernetes Dashboard.....	21
Task 2: Run ssh-keygen on all nodes.....	21
Task 3: Copy SSH keys to Management workstation.....	22
Task 4: Test to ensure SSH keys are properly set.....	22
Exercise 5 : Install a CaaS Platform Nodes.....	24
Task 1: Deploy the first Kubernetes Master node.....	24
<b>Section 2 : Connecting to the Cluster.....</b>	<b>27</b>
Exercise 1 : Use the kubectl Command to Display Info About the Cluster.....	28
Task 1: Configure workstation for kubectl command.....	28
Task 2: Use the kubectl Command.....	28
Exercise 2 : Deploy the Kubernetes Dashboard.....	31
Task 1: Deploy the Kubernetes Dashboard.....	31
Task 2: Access the Kubernetes Dashboard.....	31
Exercise 3 : Install a CaaS Platform Nodes.....	33
Task 1: Install Helm on Workstation.....	33
Task 2: Create Service account and Cluster Role for Helm.....	33
Task 3: Initialize Helm.....	33
Task 4: Add SUSE Charts to Helm.....	34
Task 5: Install Centralize Loggin.....	35
<b>Section 3 : Deploying a Workload.....</b>	<b>36</b>
Exercise 1 : Deploy a Simple Pod on Kubernetes.....	37
Task 1: View a Manifest for the Deployment.....	37
Task 2: Deploy a simple application (GUI option).....	38
Task 3: Deploy a simple application (Command Line option).....	40
Exercise 2 : Delete a Deployment on Kubernetes.....	42
Task 1: Delete the Deployment (GUI Option).....	42
Task 2: Delete the Deployment (Command Line option).....	42
Exercise 3 : Scale Out a Deployment.....	44
Task 1: View the Manifest for the Deployment.....	44
Task 2: Scale the Deployment (GUI option).....	44

Task 3: Scale the Deployment (Command Line Option).....	46
<b>Section 4 : Work With Kubernetes.....</b>	<b>47</b>
Exercise 1 : Update a Deployment.....	48
Task 1: Create a New Manifest for the Deployment.....	48
Task 2: Update the Deployment.....	48
Exercise 2 : Update a Deployment Via Rolling Updates.....	49
Task 1: Create a New Manifest for the Deployment.....	50
Task 2: Update the Deployment.....	50
Exercise 3 : Expose a Service Running in a Pod.....	52
Task 1: Create a Manifest for the Service.....	52
Task 2: Define the Service.....	52
Task 3: Access the Exposed Service.....	53
Exercise 4 : Setup Readiness and Liveness Probes.....	54
Task 1: View manifest for the Deployment.....	54
Task 2: Deploy Manifest.....	55
Task 3: View the Replica sets.....	55
Task 4: Kill an nginx server.....	55
Exercise 5 : Define Limits for Containers and Pods in Kubernetes.....	57
Task 1: Create a New Namespace in the Cluster.....	57
Task 2: Create a Manifest for the Limits.....	57
Task 3: Apply the Limits to the Namespace.....	58
Exercise 6 : Introduction to Helm.....	59
Task 1: Use Some Basic Helm Commands.....	59
Exercise 7 : Deploy an Application with Helm.....	61
Task 1: Create Helm Chart Config File.....	61
Task 2: Deploy the Helm Chart.....	61
Task 3: Access the Application Deployed by the Helm Chart.....	62
Task 4: Update the Application Release.....	63
Task 5: Delete a Deployed Helm Chart Release.....	63
<b>Section 5 : Deploying Advanced Workload.....</b>	<b>65</b>
Exercise 1 : Run Heimdall as a standalone container.....	66
Task 1: Create a place for Heimdall to store its files.....	66
Task 2: Run Heimdall as a container.....	66
Task 3: On the Workstation launch the Chrome Browser.....	67
Task 4: Create an App in Helm.....	68
Task 5: Stop and restart container.....	69
Task 6: Verify with Browser.....	69
Task 7: Stop Container Instance.....	69
Task 8: Try and run multiple versions of Heimdall.....	69
Exercise 2 : Launch Heimdall.....	71
Task 1: Create the Manifest for the Persistent Volume.....	71
Task 2: Deploy Heimdall.....	72
Task 3: View the Deployment.....	72
Task 4: View Heimdall in a Browser.....	74
Task 4: Re-deploy Heimdall.....	74
Exercise 3 : Configure Heimdall to use NFS Persistent Storage.....	75

<u>SUSE U Advanced - SUSE CaaS Platform</u>	
Task 1: Create the Manifest for the Persistent Volume.....	.75
Task 2: Create the Manifest for the Persistent Volume Claim.....	.75
Task 3: Create the Manifest for the Service so we can access the deployment.....	.76
Task 4: Create the Manifests for the Web Frontend.....	.76
Task 5: Deploy the Objects.....	.77
Task 6: Test the Persistent Data.....	.78
Task 7: Remove the Objects from the Cluster.....	.78
<b>Section 6 : Appendix: Bonus Labs.....</b>	<b>79</b>
<b>Section 7 : Horizontal Scale.....</b>	<b>80</b>
Exercise 1 : Configure Horizontal Pod Autoscaling.....	81
Task 1: Install Metric Server.....	81
Task 2: View the Manifest for the App to be Scaled.....	82
Task 3: View the Manifest for the App's Service.....	83
Task 4: View the Manifest for the Autoscaler.....	83
Task 5: Deploy the AutoScaler Objects.....	83
Task 6: Create the Manifest for the Load Generator.....	84
Task 7: Cause the Deployment To Scale Out.....	84
Task 8: Cause the Deployment To Scale Back.....	85
Task 9: Clean Up the Deployments.....	85
Exercise 2 : CaaS Platform Network and Logs.....	87
Task 1: Display all Flannel networks.....	87
Task 2: Deploy Counting Pod.....	87
Step 3: Review Kubernetes Events.....	87
Step 4: View Container Logs.....	88
Task 5: Gather Supportconfig logs.....	88
Task 6: Review Supportconfig.....	89
Exercise 3 : Deploy a Stateful App with Volume Storage on Kubernetes.....	90
Task 1: Create a Manifest for the Deployment.....	90
Task 2: Deploy the Pod.....	91
Exercise 4 : Configure DNS for CaaSP.....	92
Task 1: Install DHCP and DNS.....	92
Task 2: Configure DNS for CaaSP.....	92
Exercise 5 : Install and Configure DHCP for CaaSP.....	95
Task 1: Install DHCP.....	95
Task 2: Configure DHCP for CaaSP.....	95

## Documentation Conventions:

---

The following typographical conventions are used in this manual:

<b>Bold</b>	Represents things you should pay attention to or buttons you click, text or options that you should click/select/type in a GUI.
<b>Bold Gray</b>	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.
<b>Bold Red</b>	Represents warnings or very important information.
<b>Option &gt; Option &gt; Option</b>	Represents a chain of items selected from a menu.
<b><i>BOLD_UPPERCASE_ITALIC</i></b>	Represents an “exercise variable” that you replace with another value.
<b>bold monospace</b>	Represents text displayed in a terminal or entered in a file.
<b>bold monospace blue</b>	Represents commands entered at the command line.
<b>bold monospace green</b>	Represents a file name.

SUSE U Advanced - SUSE CaaS Platform

SUSE U Advanced - SUSE CaaS Platform

# 1 Demonstration of SUSE CaaS Platform

---

## Description:

How to highlight the Key Benefits of SUSE CaaS Platform

**(No Exercises)**

## 1- 1 Start the Lab VMs

### Description:

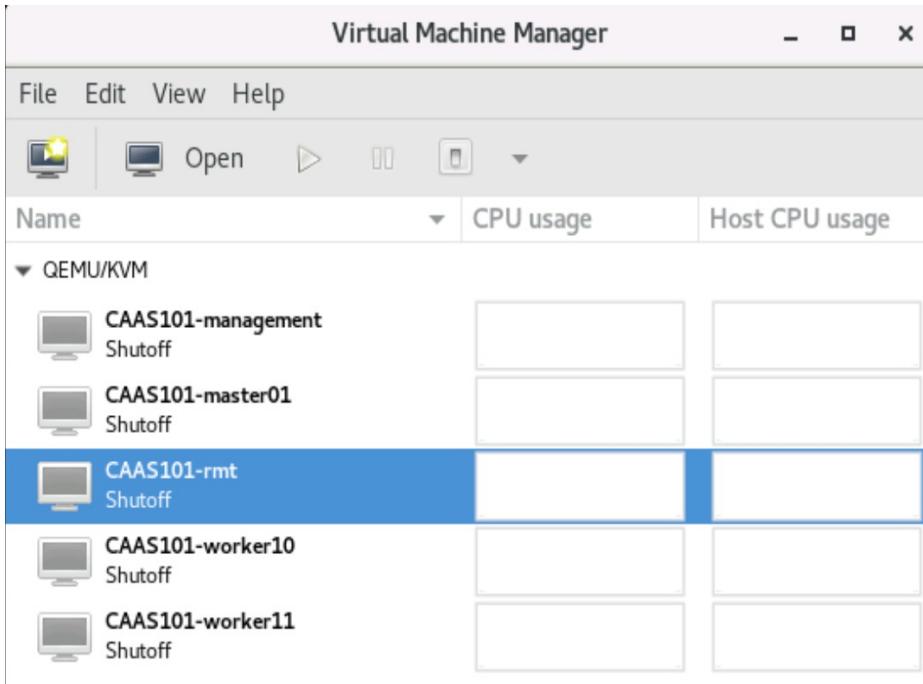
In this exercise, you use the Virt-Manager utility to start the lab VM(s) in the proper order.

### Task 1: Start the Lab VMs

1. On the VM host, launch the Virt-Manager utility  
(4<sup>th</sup> 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-rmt** VM and select **Run**



3. Double-click on the **CAAS101-rmt** VM to view its console

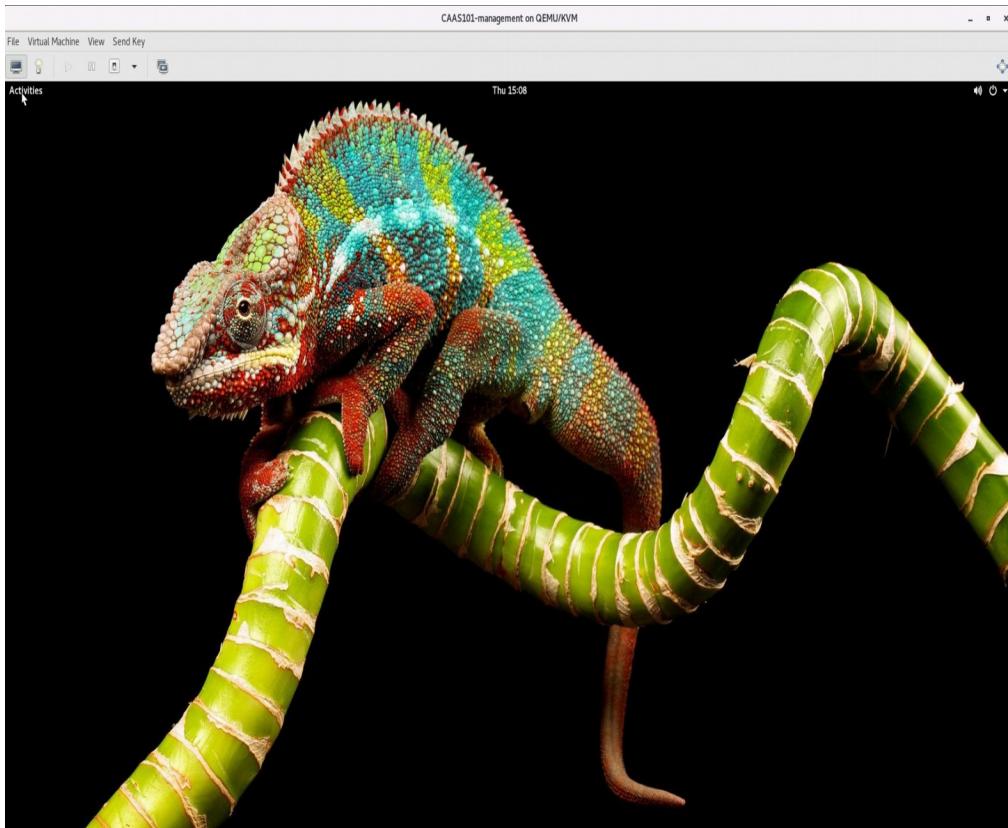
When the **CAAS101-rmt** VM has finished booting completely, close its console window.  
(You can use the CPU utilization graph in Virt-Manager to see when the node is finished booting and starting services).

4. Power on the reset of the Virtual Machines **CAAS101-management**, **CAAS101-master**, **CAAS101-worker10** and **CAAS101-worker10**

\*note you can power all of the on at the same time

When the **Workstation** 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 3<sup>rd</sup> icon down



---

### **Summary:**

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

(End of Exercise)

## 1- 2 Install a CaaS Platform Admin Node (Needs Updated)

---

### 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 PXE boot screen, use the arrow keys to select the following boot target:  
**Install CAASP\_ADMIN\_MAC (admin)**



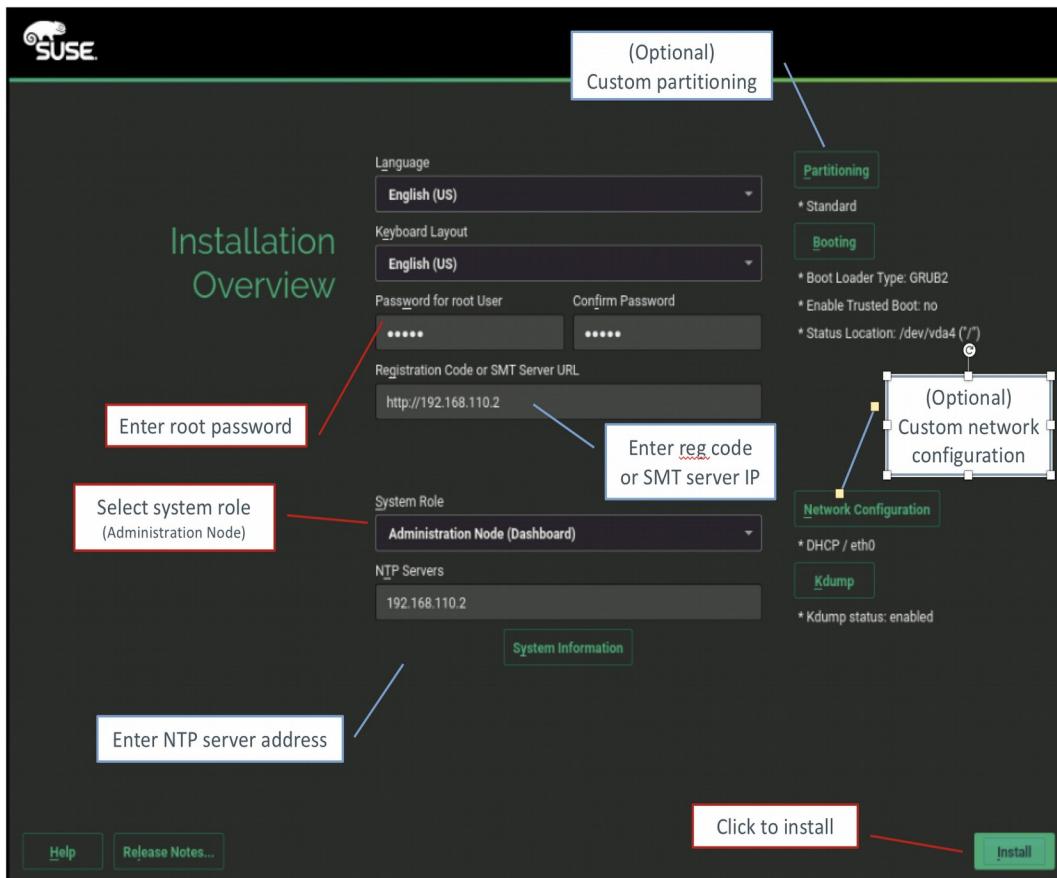
5. The installation process should begin.
6. On the **Install Overview** screen select/enter the following (leaving all others with their default values):

**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 **Trust** the certificate

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. 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**

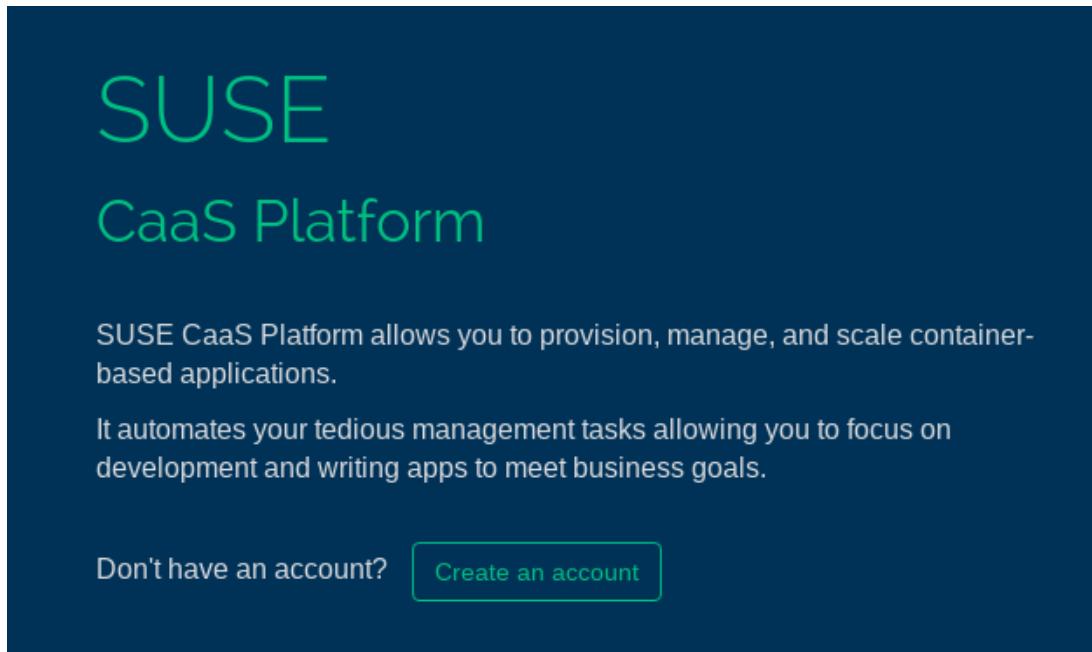


**Note:**

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 a Service Platform** login screen.

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



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

**Email:** admin@example.com

**Password:** susecaasp

4. Click: **Create Admin**

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

The screenshot shows the 'Initial CaaSP Configuration' screen with the following sections and their current configurations:

- Generic settings**:
  - Internal Dashboard Location**: 192.168.110.99
- Cluster services**:
  - Install Tiller (Helm's server component)
- Overlay network settings**:
  - Show button
- Proxy settings**:
  - Enable button
  - Disable button
- SUSE registry mirror**:
  - Enable button
  - Disable button
- Container runtime**:
  - The choice of container runtime is completely transparent to end-users of the cluster. Neither Kubernetes manifests nor container images need to be changed.
  - Choose the runtime**: Docker open source engine (cri-o) is selected.
  - Docker open source engine (default) is a production-ready runtime, fully supported by SUSE.
- System wide certificate**:
  - Show button

A cursor arrow is pointing towards the 'Next' button at the bottom right of the screen.

**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)**

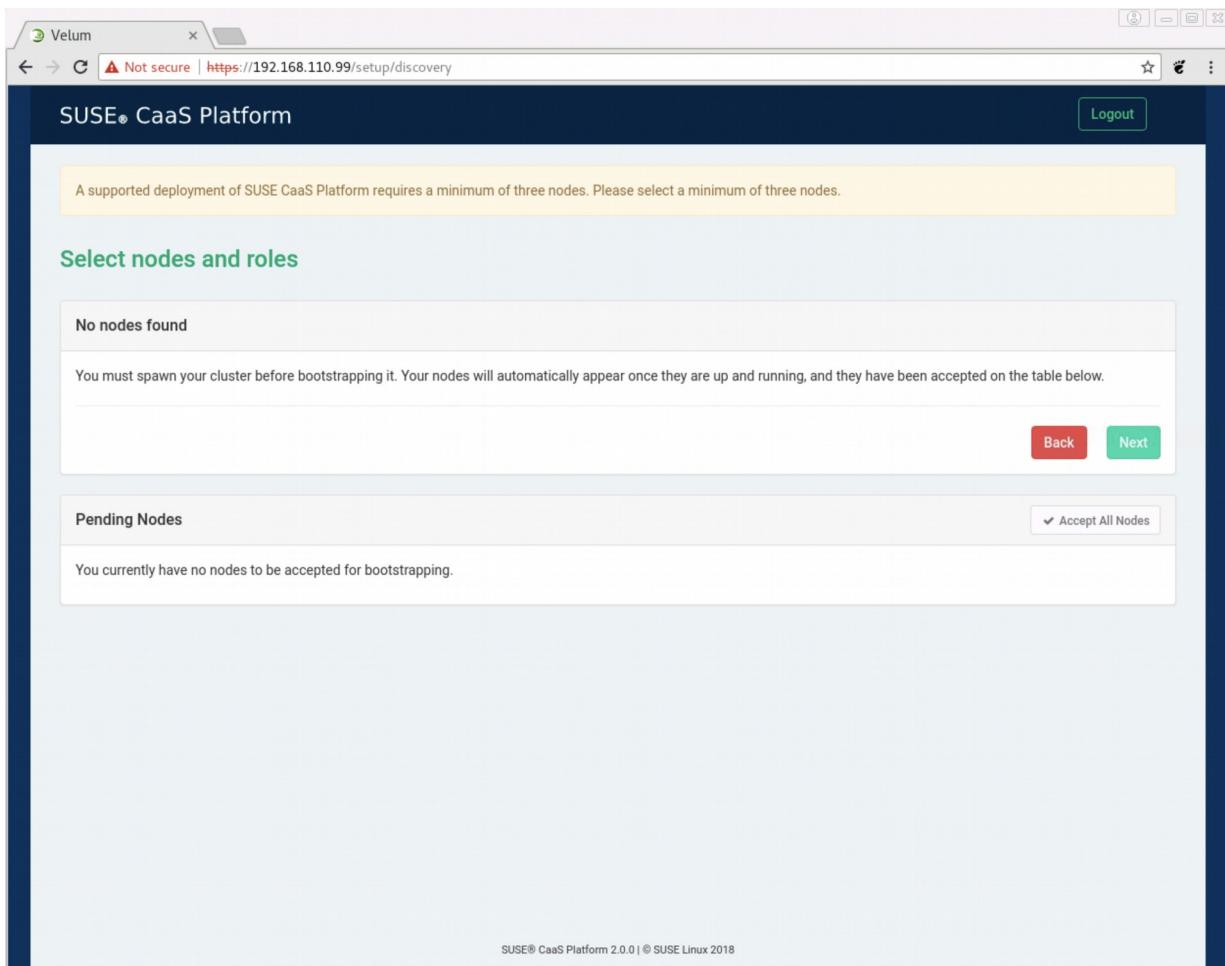
**Proxy settings:** **(disable)**

**SUSE registry mirror:** **(disable)**

6. 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.
7. 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.



8. 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)

## 1-3 Install a CaaS Platform Nodes

---

### Description:

In this exercise, you install a SUSE CaaS Platform all nodes.

### Dependencies:

The RMT server must be configured (Optional Docker Registry)

DHCP, PXE and DNS must be configured.

Master, and 2 worker nodes with SLES 15 SP1 installed and CaaS Platform Repos added

---

### Task 1: Deploy the Kubernetes Dashboard

1. On the **Workstation**, open a terminal
2. Enter the following command to install skuba and the kubernetes client

**sudo zypper in skuba kubernetes-client**

I will tell you the amount of additional drive space needed and it will prompt you to press **y** to continue

3. Press the '**Y**' key to continue

when it finished you should see a screen similar to the image below

```
tux@workstation:~ x
File Edit View Search Terminal Help
Retrieving: kubernetes-common-1.15.0-2.4.1.x86_64.rpm ..... [done]
Retrieving package skuba-0.8.1-2.10.2.x86_64
(2/5), 14.4 MiB ( 45.5 MiB unpacked)
Retrieving: skuba-0.8.1-2.10.2.x86_64.rpm ..... [done (10.7 MiB/s)]
Retrieving package skuba-update-0.8.1-2.10.2.noarch
(3/5), 54.4 KiB ( 33.5 KiB unpacked)
Retrieving: skuba-update-0.8.1-2.10.2.noarch.rpm ..... [done]
Retrieving package kubernetes-client-1.15.0-2.4.1.x86_64
(4/5), 1.2 MiB (743.3 KiB unpacked)
Retrieving: kubernetes-client-1.15.0-2.4.1.x86_64.rpm ..... [done]
Retrieving package kubectl-caasp-0.8.1-2.10.2.x86_64
(5/5), 14.4 MiB ( 45.4 MiB unpacked)
Retrieving: kubectl-caasp-0.8.1-2.10.2.x86_64.rpm ..... [done]
Checking for file conflicts: ..... [done]
(1/5) Installing: kubernetes-common-1.15.0-2.4.1.x86_64 ..... [done]
(2/5) Installing: skuba-0.8.1-2.10.2.x86_64 ..... [done]
(3/5) Installing: skuba-update-0.8.1-2.10.2.noarch ..... [done]
Additional rpm output:
Updating /etc/sysconfig/skuba-update ...

(4/5) Installing: kubernetes-client-1.15.0-2.4.1.x86_64 ..... [done]
(5/5) Installing: kubectl-caasp-0.8.1-2.10.2.x86_64 ..... [done]
tux@workstation:~> |
```

## Task 2: Test to ensure SSH keys are properly set

1. On the **Workstation**, open a terminal
2. Enter the following command in the terminal to test your **ssh keys** to master01  
**ssh root@master01**
3. You should see a red prompt that says master01

```
tux@workstation:~  
File Edit View Search Terminal Help  
tux@workstation:~> ssh root@master01  
Last login: Sat Aug  3 08:54:16 2019 from 192.168.110.108  
master01:~ #
```

4. Notice that you did not need to type in a password or accept a key?  
This is because of the pre-work done on the workstation
5. Exit your ssh session from the terminal prompt  
**exit**
6. Repeat the step 2 and 3 for **worker10** and **worker11**

---

**Summary:**

In this exercise, you installed Skuba and the Kubernetes Client Utilities. You also tested that the SSH keys were set up properly and you could seamlessly connect to all of the nodes.

(End of Exercise)

## 1-4 Setup SSH Keys

---

### Description:

In this exercise, you install setup all of the machines so you can ssh to any of the nodes with a username or password

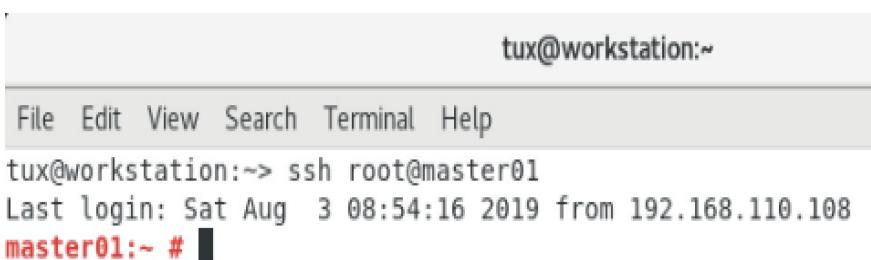
---

### Task 1: Deploy the Kubernetes Dashboard

1. On the **Workstation**, open a terminal
2. Enter the following command to Start the ssh-agent  
**eval "\$(ssh-agent -s)"**
3. Enter the following command to create an ssh key  
**ssh-keygen**  
When it asks you questions just take the default

### Task 2: Run ssh-keygen on all nodes

1. On the **Workstation**, open a terminal
2. Enter the following command in the terminal to test your **ssh keys** to master01  
**ssh root@master01**  
password is **linux**
3. You should see a red prompt that says master01



A screenshot of a terminal window. The title bar says "tux@workstation:~". The menu bar includes "File Edit View Search Terminal Help". The command line shows "tux@workstation:~> ssh root@master01". Below that, it says "Last login: Sat Aug 3 08:54:16 2019 from 192.168.110.108". At the bottom, the prompt is "master01:~ #".

4. Enter the following command in the terminal to create an **ssh key**  
**ssh-keygen**
5. Exit your ssh session from the terminal prompt

**exit**

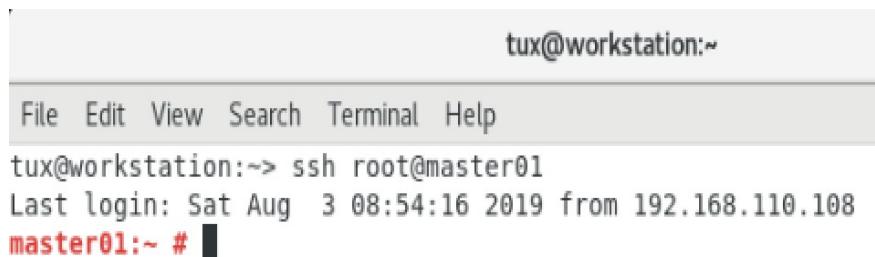
6. Repeat the step 2 and 3 for **worker10** and **worker11**

### Task 3: Copy SSH keys to Management workstation

1. On the **Workstation**, open a terminal
2. Enter the following command in the terminal to test your **ssh keys** to master01  
**ssh-copy-id -i ~/.ssh/id\_rsa.pub root@master**
3. Repeat the step 2 and 3 for **worker10** and **worker11**

### Task 4: Test to ensure SSH keys are properly set

1. On the **Workstation**, open a terminal
2. Enter the following command in the terminal to test your **ssh keys** to master01  
**ssh root@master01**
3. You should see a red prompt that says master01



A screenshot of a terminal window. The title bar says "tux@workstation:~". The menu bar includes "File Edit View Search Terminal Help". The command line shows "tux@workstation:~> ssh root@master01". Below that, it says "Last login: Sat Aug 3 08:54:16 2019 from 192.168.110.108". At the bottom, the prompt is "master01:~ #".

4. Notice that you did not need to type in a password or accept a key?  
This is because of the pre-work done on the workstation
5. Exit your ssh session from the terminal prompt  
**exit**
6. Repeat the step 2 and 3 for **worker10** and **worker11**

---

**Summary:**

In this exercise, you set it up so that you can ssh from the Management Workstation to any of the nodes without asking for a username or a password

(End of Exercise)

## 1- 5 Install a CaaS Platform Nodes

---

### Description:

In this exercise, you install a SUSE CaaS Platform all nodes.

### Dependencies:

The RMT server must be configured (Optional Docker Registry)

DHCP, PXE and DNS must be configured.

Master, and 2 worker nodes with SLES 15 SP1 installed and CaaS Platform Repos added

---

### Task 1: Deploy the first Kubernetes Master node

1. On the **Workstation**, open a terminal
2. Enter the following command create the basic config to our **kubernetes cluster**

**skuba cluster init --control-plane master.example.com my-cluster**

```
tux@workstation:~> skuba cluster init --control-plane master.example.com my-cluster
** This is a BETA release and NOT intended for production usage. **
[init] configuration files written to /home/tux/my-cluster
tux@workstation:~> █
```

3. Enter the following command to change into the newly created folder

**cd my-cluster**

4. Look at the cluster config filename **my-cluster**

**less kubeadm-init.conf**

```
tux@workstation:~/my-cluster> cat kubeadm-init.conf
apiVersion: kubeadm.k8s.io/v1beta1
kind: InitConfiguration
bootstrapTokens: []
localAPIEndpoint:
  advertiseAddress: ""
---
apiVersion: kubeadm.k8s.io/v1beta1
kind: ClusterConfiguration
apiServer:
  certSANs:
    - master.example.com
extraArgs:
  oidc-issuer-url: https://master.example.com:32000
  oidc-client-id: oidc
  oidc-ca-file: /etc/kubernetes/pki/ca.crt
  oidc-username-claim: email
  oidc-groups-claim: groups
clusterName: my-cluster
controlPlaneEndpoint: master.example.com:6443
dns:
  imageRepository: registry.suse.com/caasp/v4
  imageTag: 1.3.1
  type: CoreDNS
etcd:
  local:
    imageRepository: registry.suse.com/caasp/v4
    imageTag: 3.3.11
imageRepository: registry.suse.com/caasp/v4
kubernetesVersion: 1.15.0
networking:
  podSubnet: 10.244.0.0/16
  serviceSubnet: 10.96.0.0/12
useHyperKubeImage: true
tux@workstation:~/my-cluster> █
```

5. Install, configure and bootstrap the cluster-info

**skuba node bootstrap --target master.example.com master01**

```
tux@workstation:~/my-cluster> skuba node bootstrap --target master.example.com master01
** This is a BETA release and NOT intended for production usage. **
W0808 17:13:25.590213    2588 ssh.go:306]
The authenticity of host '192.168.110.7:22' can't be established.
ECDSA key fingerprint is 54:3e:9b:2e:21:fc:5c:70:ff:c4:99:7c:fa:b4:08:d0.
I0808 17:13:25.590316    2588 ssh.go:307] accepting SSH key for "master.example.com:22"
I0808 17:13:25.590327    2588 ssh.go:308] adding fingerprint for "master.example.com:22" to "known_hosts"
[bootstrap] updating init configuration with target information
[bootstrap] writing init configuration for node
[bootstrap] applying init configuration to node
[bootstrap] downloading secrets from bootstrapped node "master.example.com"
[bootstrap] successfully bootstrapped node "master.example.com" with Kubernetes: "1.15.0"
tux@workstation:~/my-cluster> █
```

---

6. add a **worker10** node to the cluster

**skuba node join --role worker --target worker10.example.com worker10**

```
tux@workstation:~/my-cluster> skuba node join --role worker --target worker10.example.com worker10
** This is a BETA release and NOT intended for production usage. **
W0808 17:23:50.580677    2632 ssh.go:306]
The authenticity of host '192.168.110.10:22' can't be established.
ECDSA key fingerprint is 19:7c:db:38:d2:c5:34:f6:cb:90:21:4e:1a:b5:5e:d7.
I0808 17:23:50.580863    2632 ssh.go:307] accepting SSH key for "worker10.example.com:22"
I0808 17:23:50.580916    2632 ssh.go:308] adding fingerprint for "worker10.example.com:22" to "known_hosts"
[join] applying states to new node
[join] node successfully joined the cluster
tux@workstation:~/my-cluster> █
```

7. add a **worker11** node to the cluster

**skuba node join --role worker --target worker11.example.com worker11**

You should seem message very similar to the message you received in the previous step.

8. Verify Cluster Status

**skuba cluster status**

```
tux@workstation:~/my-cluster> skuba cluster status
** This is a BETA release and NOT intended for production usage. **
NAME      OS-IMAGE          KERNEL-VERSION   KUBELET-VERSION   CONTAINER-RUNTIME   HAS-UPDATES   HAS-DISRUPTIVE-UPDATES
master01  SUSE Linux Enterprise Server 15 SP1  4.12.14-197.10-default  v1.15.0        cri-o://1.15.0    <none>       <none>
worker10  SUSE Linux Enterprise Server 15 SP1  4.12.14-197.10-default  v1.15.0        cri-o://1.15.0    <none>       <none>
worker11  SUSE Linux Enterprise Server 15 SP1  4.12.14-197.10-default  v1.15.0        cri-o://1.15.0    <none>       <none>
tux@workstation:~/my-cluster> █
```

---

**Summary:**

In this exercise, you installed Skuba and the Kubernetes Client Utilities. You also tested that the SSH keys were set up properly and you could seamlessly connect to all of the nodes.

(End of Exercise)

## 2 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.

## 2- 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: Configure workstation for kubectl command

1. On the Workstation, open a terminal
2. Create official folder for kubernetes config filename  
**mkdir ~/.kube**
3. On the Workstation, open a terminal
4. Create config file

**cp ~/my-cluster/admin.conf ~/.kube/config**

### 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.

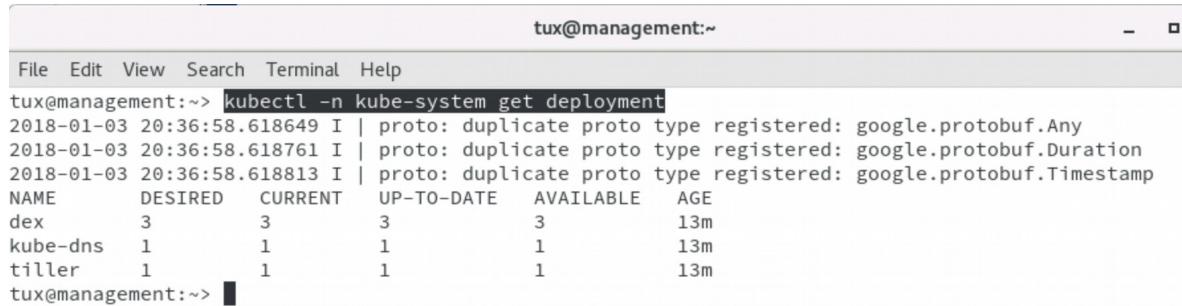
## SUSE U Advanced - SUSE CaaS Platform

```
tux@management:~/course_files/CAAS101/scripts> kubectl get nodes
2018-08-08 10:02:50.924418 I | proto: duplicate proto type registered: google.protobuf.Any
2018-08-08 10:02:50.924462 I | proto: duplicate proto type registered: google.protobuf.Duration
2018-08-08 10:02:50.924473 I | proto: duplicate proto type registered: google.protobuf.Timestamp
NAME      STATUS     AGE      VERSION
master01   Ready      1d      v1.9.8
master02   Ready      1d      v1.9.8
master03   Ready      1d      v1.9.8
worker10   Ready      1d      v1.9.8
worker11   Ready      1d      v1.9.8
worker12   Ready      1d      v1.9.8
tux@management:~/course_files/CAAS101/scripts> █
```

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.



The screenshot shows a terminal window with the title "tux@management:~". The menu bar includes File, Edit, View, Search, Terminal, and Help. The command "kubectl -n kube-system get deployment" is entered in the terminal. The output shows deployment details for three components: dex, kube-dns, and tiller. The output is as follows:

```
tux@management:~> kubectl -n kube-system get deployment
2018-01-03 20:36:58.618649 I | proto: duplicate proto type registered: google.protobuf.Any
2018-01-03 20:36:58.618761 I | proto: duplicate proto type registered: google.protobuf.Duration
2018-01-03 20:36:58.618813 I | proto: duplicate proto type registered: google.protobuf.Timestamp
NAME      DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
dex       3          3          3           3           13m
kube-dns  1          1          1           1           13m
tiller    1          1          1           1           13m
tux@management:~> █
```

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.

```
tux@management:~$ kubectl -n kube-system get pods
2018-01-03 20:34:37.713518 I | proto: duplicate proto type registered: google.protobuf.Any
2018-01-03 20:34:37.713579 I | proto: duplicate proto type registered: google.protobuf.Duration
2018-01-03 20:34:37.713607 I | proto: duplicate proto type registered: google.protobuf.Timestamp
NAME                           READY   STATUS    RESTARTS   AGE
dex-3532614258-cssdz          1/1     Running   2          10m
dex-3532614258-nzjvq          1/1     Running   2          10m
dex-3532614258-xvpc9          1/1     Running   2          10m
haproxy-6c4cbef4e0545fc97a891c78b2f2b1c.infra.caasp.local 1/1     Running   0          9m
haproxy-b099b425f0194d18902aa68729c2a12e.infra.caasp.local 1/1     Running   0          10m
haproxy-d115a10918394ac88ab8c88bf7350556.infra.caasp.local 1/1     Running   0          10m
haproxy-dda6d06e26214c6d9c208ec2899748d2.infra.caasp.local 1/1     Running   0          10m
haproxy-e8185390bbc544efba4965e117a960eb.infra.caasp.local 1/1     Running   0          10m
haproxy-f7f84cefb02e4eadbcd03ce48642eef.infra.caasp.local 1/1     Running   0          9m
kube-dns-1144198277-3zf34      3/3     Running   0          11m
tiller-3881891813-4251j        1/1     Running   0          10m
tux@management:~$
```

---

**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)

## 2- 2 Deploy the Kubernetes Dashboard

---

### Description:

In this exercise, you deploy the Kubernetes Dashboard.

We will also apply an RBAC configuration that turns off token based authentication and stops the dashboard from timing out after 10 mins.

---

### Task 1: Deploy the Kubernetes Dashboard

1. On the **Workstation**, open a terminal
2. Enter the following command to deploy the Kubernetes Dashboard:

**kubectl apply -f ~/STW-CaaSPv4/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**

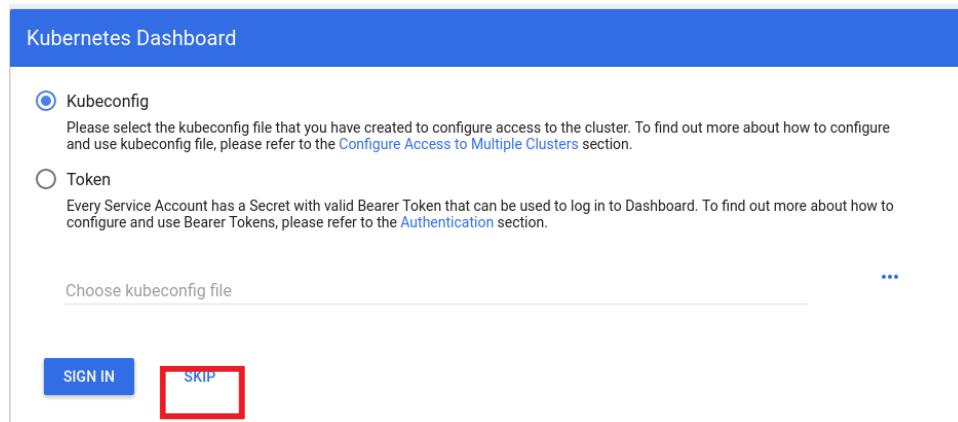
You should be detached from the screen session and back at a command prompt.

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)

## 2- 3 Install a CaaS Platform Nodes

---

### Description:

In this exercise, you install Helm and config it to pull charts from SUSE.

---

### Task 1: Install Helm on Workstation

1. On the **Workstation**, open a terminal
2. Enter the following command to install skuba and the kubernetes client

**sudo zypper in helm**

I will tell you the amount of additional drive space needed and it will prompt you to press **y** to continue

### Task 2: Create Service account and Cluster Role for Helm

1. On the **Workstation**, open a terminal
2. Enter the following command in the terminal to create a service account for Helm to use  
**kubectl create serviceaccount --namespace kube-system tiller**
3. Enter the following command in the terminal to create a cluster role for Helm to use

**kubectl create clusterrolebinding tiller --clusterrole=cluster-admin --serviceaccount=kube-system:tiller**

### Task 3: Initialize Helm

1. On the **Workstation**, open a terminal
2. Enter the following command in the terminal to create a service account for Helm to use  
**helm init --tiller-image registry.suse.com/caasp/v4/helm-tiller:2.8.2 --service-account tiller**

3. Look to see if tiller is currently running (give it a minute if it is not in the **Ready** state)

### **kubectl get pods –all-namespaces**

```
tux@localhost:~> kubectl get pods --all-namespaces
NAMESPACE     NAME                               READY   STATUS    RESTARTS   AGE
kube-system   cilium-82kz9                         1/1    Running   0          47m
kube-system   cilium-bzsp7                         1/1    Running   0          47m
kube-system   cilium-operator-7d6dddbf5-qnbp5      1/1    Running   0          50m
kube-system   cilium-rwwjw                         1/1    Running   0          47m
kube-system   coredns-69c4947958-75vdp            1/1    Running   1          50m
kube-system   coredns-69c4947958-vmf6d            1/1    Running   3          50m
kube-system   etcd-master01                        1/1    Running   0          49m
kube-system   kube-apiserver-master01             1/1    Running   0          49m
kube-system   kube-controller-manager-master01      1/1    Running   0          49m
kube-system   kube-proxy-2nhh6                      1/1    Running   0          50m
kube-system   kube-proxy-ftbck                     1/1    Running   0          47m
kube-system   kube-proxy-pj698                      1/1    Running   0          48m
kube-system   kube-scheduler-master01              1/1    Running   0          49m
kube-system   kubernetes-dashboard-5f9c6b756f-hkwxh 1/1    Running   0          45m
kube-system   kured-4qzsf                          1/1    Running   0          45m
kube-system   kured-7tcfj                          1/1    Running   0          49m
kube-system   kured-9z4g7                          1/1    Running   0          47m
kube-system   oidc-dex-55fc689dc-mccw8            1/1    Running   1          50m
kube-system   oidc-gangway-7b7fbbdbdf-4w6z8        1/1    Running   0          50m
kube-system   tiller-deploy-7c666b7c99-7whpp       1/1    Running   0          42m
tux@localhost:~>
```

## Task 4: Add SUSE Charts to Helm

1. On the **Workstation**, open a terminal
2. Enter the following command in the terminal to create a service account for Helm to use  
**helm repo add suse-charts <https://kubernetes-charts.suse.com>**
3. Look at available charts from SUSE

### **helm search suse**

```
tux@localhost:~> helm search suse
NAME          CHART VERSION APP VERSION DESCRIPTION
suse-charts/cf           2.17.1      1.4.1      A Helm chart for SUSE Cloud Foundry
suse-charts/cf-usb-sidecar-mysql 1.0.1       ...
suse-charts/cf-usb-sidecar-postgres 1.0.1       ...
suse-charts/console        2.4.0       2.4.0      A Helm chart for deploying Stratos UI Console
suse-charts/log-agent-rsyslog 1.0.1       8.39.0     Log Agent for forwarding logs of K8s control pl...
suse-charts/metrics         1.0.0       1.0.0      A Helm chart for Stratos Metrics
suse-charts/minibroker       0.2.0       ...
suse-charts/nginx-ingress    0.28.4      0.15.0     An nginx Ingress controller that uses ConfigMap...
suse-charts/uaa              2.17.1      1.4.1      A Helm chart for SUSE UAA
tux@localhost:~>
```

## Task 5: Install Centralize Loggin

1. On the **Workstation**, open a terminal
2. Enter the following command in the terminal to create a service account for Helm to use

```
helm install suse-charts/log-agent-rsyslog --name 1.0.1 --set server.host=rsyslog-server.default.svc.cluster.local --set server.port=514
```

---

### **Summary:**

In this exercise, you installed Helm on the Management workstation. You then setup and configured it to pull Charts directly from SUSE. You then installed the CaaS Platform's Centralized Logging Service.

(End of Exercise)

## 3 Deploying a Workload

---

### Description:

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

## 3- 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:

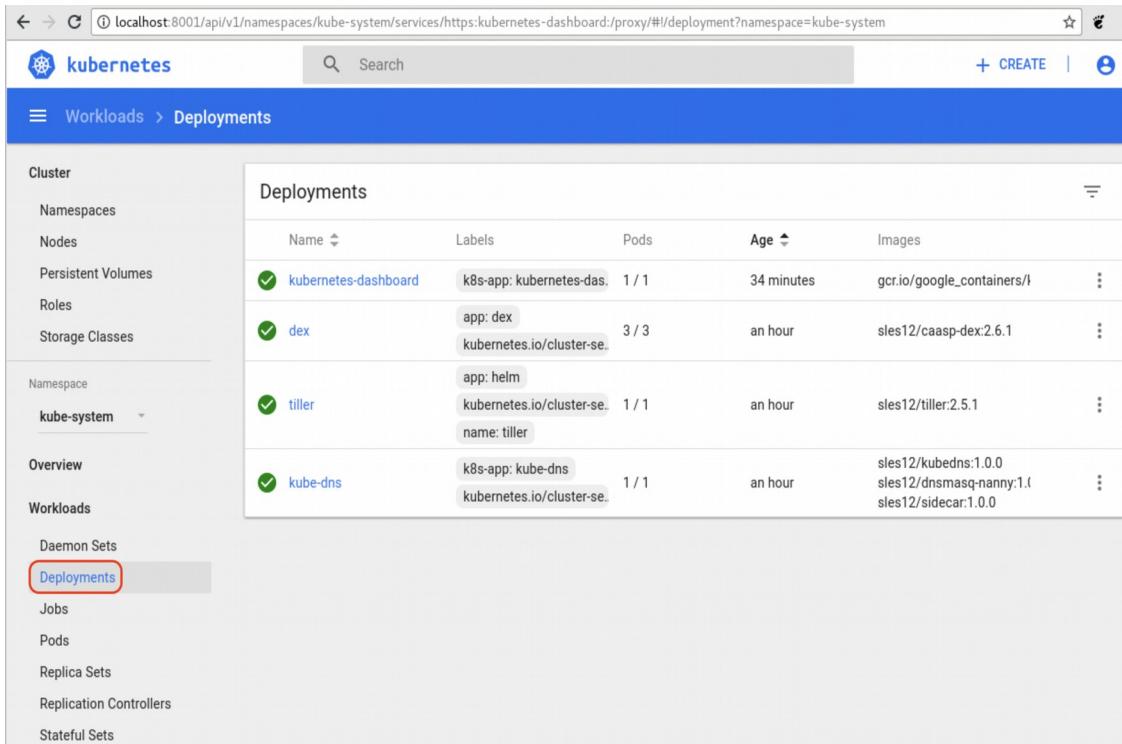
**~/STW-CaaSPv4/labs/nginx-deployment.yaml**

```
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/sles12sp3_nginx
          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

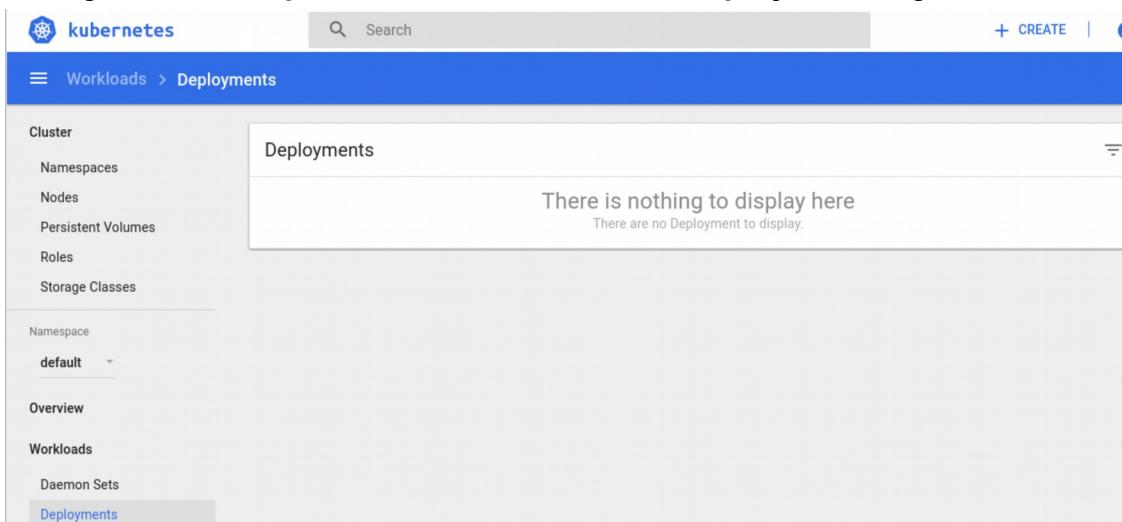


The screenshot shows the Kubernetes Dashboard interface. The left sidebar is titled 'Workloads' and includes options for Cluster, Namespaces, Nodes, Persistent Volumes, Roles, Storage Classes, Namespace (set to 'kube-system'), Overview, Workloads (Daemon Sets, Deployments, Jobs, Pods, Replica Sets, Replication Controllers, Stateful Sets), and a 'Deployments' button which is highlighted with a red box. The main content area is titled 'Deployments' and lists four entries:

Name	Labels	Pods	Age	Images
kubernetes-dashboard	k8s-app: kubernetes-dash-	1 / 1	34 minutes	gcr.io/google_containers/k
dex	app: dex kubernetes.io/cluster-se.	3 / 3	an hour	sles12/caasp-dex:2.6.1
tiller	app: helm kubernetes.io/cluster-se. name: tiller	1 / 1	an hour	sles12/tiller:2.5.1
kube-dns	k8s-app: kube-dns kubernetes.io/cluster-se.	1 / 1	an hour	sles12/kubedns:1.0.0 sles12/dnsmasq-nanny:1.0 sles12/sidecar:1.0.0

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



The screenshot shows the Kubernetes Dashboard interface with the 'default' namespace selected in the 'Namespace' dropdown on the left sidebar. The 'Deployments' button is also highlighted with a red box. The main content area displays a message: 'There is nothing to display here' and 'There are no Deployment to display.'

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

4. Click on **+ CREATE**
5. Select **Create from File** and the press ‘...’ to select file

CREATE FROM TEXT INPUT      **CREATE FROM FILE**      CREATE AN APP

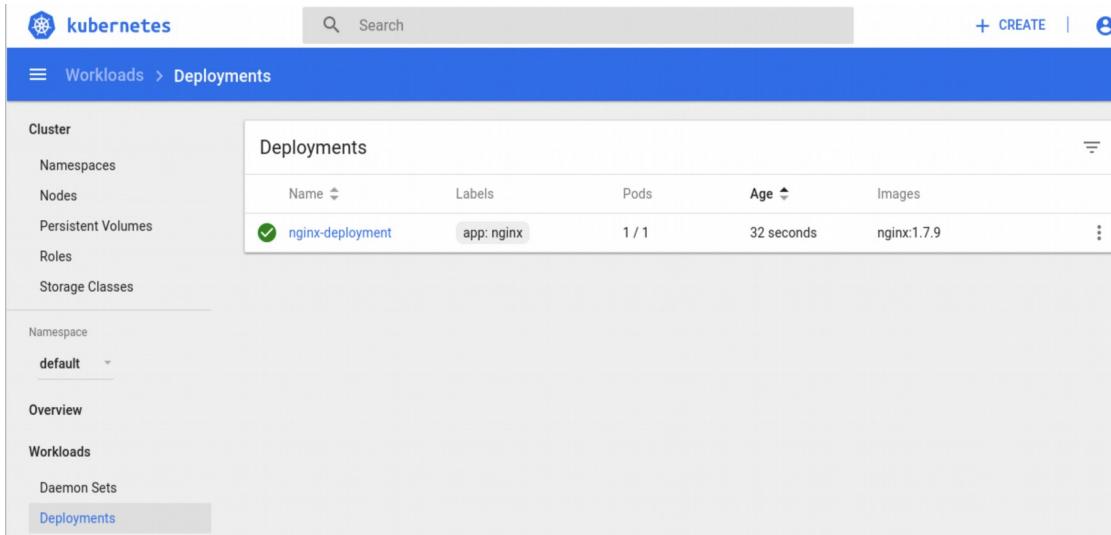
---

Select YAML or JSON file specifying the resources to deploy to the currently selected namespace. [Learn more](#) 

Choose YAML or JSON file  

**UPLOAD**      [CANCEL](#)

6. Browse to **~/STW-CaaSPv4/labs/nginx-deployment.yaml**
7. Press Upload
8. Select **Deployments** view and we should now see the nginx-deployment



The screenshot shows the Kubernetes interface under the 'Workloads' section. The 'Deployments' tab is selected. On the left, there's a sidebar with 'Cluster' and 'Namespace' sections, and a 'Workloads' section containing 'Overview', 'Daemon Sets', and 'Deployments'. The main area displays a table for 'Deployments' with the following data:

Name	Labels	Pods	Age	Images
nginx-deployment	app: nginx	1 / 1	32 seconds	nginx:1.7.9

## 9. Explore the Kubernetes Dashboard

Make sure you look at the **Overview, Workloads, and Pods** views

Name	Labels	Pods	Age	Images
nginx-deployment	app: nginx	4 / 4	4 minutes	smt.example.com:5000/sli

Name	Node	Status	Restarts	Age
nginx-deployment-58bd7995b-rq2t8	worker10	Running	0	4 minutes
nginx-deployment-58bd7995b-6gbsj	worker12	Running	0	4 minutes
nginx-deployment-58bd7995b-6sntr	worker10	Running	0	4 minutes
nginx-deployment-58bd7995b-nshqh	worker11	Running	0	4 minutes

Name	Labels	Pods	Age	Images
nginx-deployment-58bd791	app: nginx pod-template-hash: 146...	4 / 4	4 minutes	smt.example.com:5000/sli

## 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 ~/STW-CaaSPv4/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)

## 3- 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**

The screenshot shows the Kubernetes UI for managing workloads. In the top navigation bar, 'Deployments' is selected under 'Workloads'. The main view shows the 'nginx-deployment' details, including its name, namespace, labels, selector, strategy, and current status (10 updated, 10 total, 10 available, 0 unavailable). Below this, a 'New Replica Set' table lists one pod: 'nginx-deployment-431080' with 10/10 pods ready, created 39 minutes ago, and using the 'nginx:1.7.9' image. A red box highlights the 'DELETE' button in the top right corner of the deployment details section.

### Task 2: Delete the Deployment (Command Line option)

3. To view the deployments, on the **Workstation**, 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)

## 3- 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, view the **~/STW-CaaSPv4/labs/nginx-deployment.yaml** file

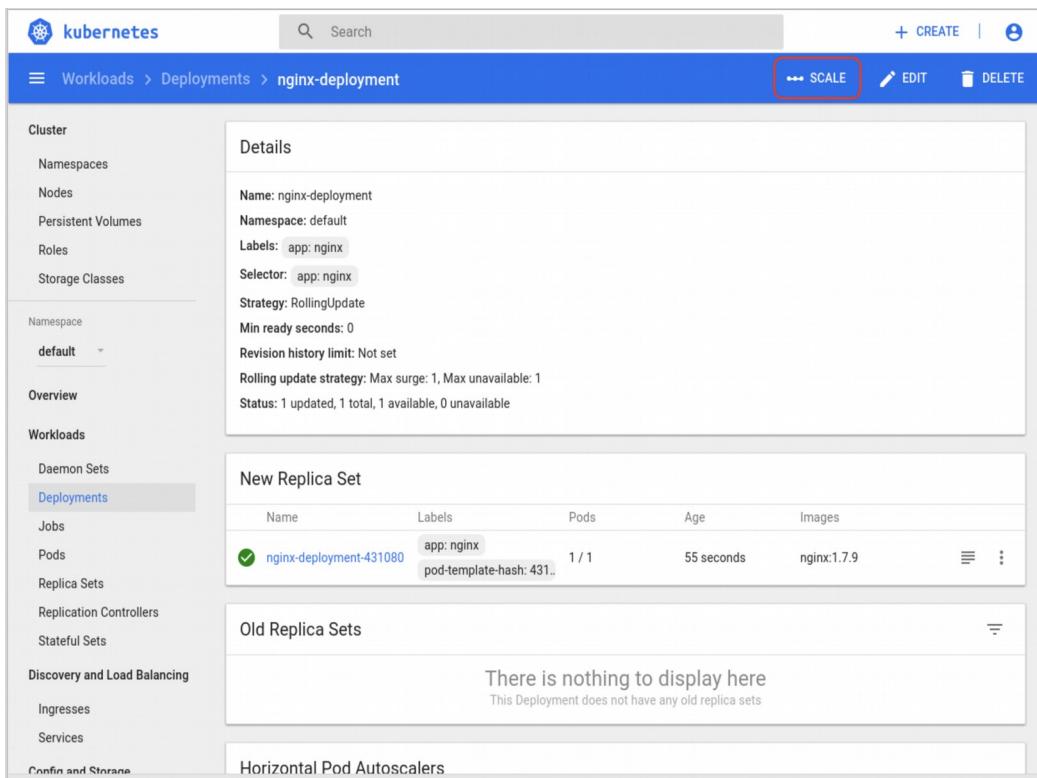
```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  replicas: 4
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: smt.example.com:5000/sles12sp3_nginx
          ports:
            - containerPort: 80
```

### Task 2: Scale the Deployment (GUI option)

1. From **Kubernetes Dashboard** select **Create** and open **~/STW-CaaSPv4/labs/nginx-scale.yaml**
2. Press the **Deploy** button
3. Select the **nginx-deployment** under **Deployments**

## SUSE U Advanced - SUSE CaaS Platform

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

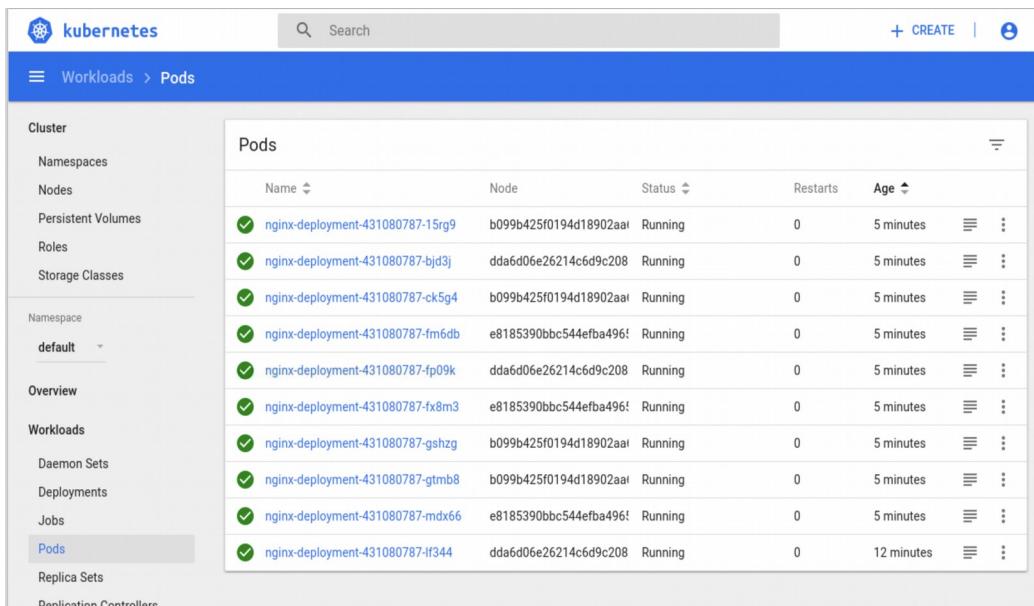


The screenshot shows the Kubernetes UI for managing a Deployment named 'nginx-deployment'. The left sidebar is collapsed. The main area displays the 'Details' tab for the deployment. Key configuration shown includes:

- Name: nginx-deployment
- Namespace: default
- Labels: app: nginx
- Selector: app: nginx
- Strategy: RollingUpdate
- Min ready seconds: 0
- Revision history limit: Not set
- Rolling update strategy: Max surge: 1, Max unavailable: 1
- Status: 1 updated, 1 total, 1 available, 0 unavailable

Below the details, the 'New Replica Set' section shows a single replica named 'nginx-deployment-431080' with 1 pod. The 'Old Replica Sets' section indicates there are none.

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



The screenshot shows the Kubernetes UI for managing Pods. The left sidebar is collapsed. The main area displays a table of running pods for the 'nginx-deployment' deployment. The table columns are:

Name	Node	Status	Restarts	Age	⋮
nginx-deployment-431080787-15rg9	b099b425f0194d18902aa	Running	0	5 minutes	⋮
nginx-deployment-431080787-bjd3j	dda6d06e26214c6d9c208	Running	0	5 minutes	⋮
nginx-deployment-431080787-ck5g4	b099b425f0194d18902aa	Running	0	5 minutes	⋮
nginx-deployment-431080787-fm6db	e8185390bbc544efba496f	Running	0	5 minutes	⋮
nginx-deployment-431080787-fp09k	dda6d06e26214c6d9c208	Running	0	5 minutes	⋮
nginx-deployment-431080787-fx8m3	e8185390bbc544efba496f	Running	0	5 minutes	⋮
nginx-deployment-431080787-gshzg	b099b425f0194d18902aa	Running	0	5 minutes	⋮
nginx-deployment-431080787-gtmb8	b099b425f0194d18902aa	Running	0	5 minutes	⋮
nginx-deployment-431080787-mdx66	e8185390bbc544efba496f	Running	0	5 minutes	⋮
nginx-deployment-431080787-if344	dda6d06e26214c6d9c208	Running	0	12 minutes	⋮

6. Scale the Pods back down to 4

## Task 3: Scale the Deployment (Command Line Option)

1. In the text editor of your choice, open the **~/STW-CaaSPv4/labs/nginx-scale.yaml** file

Note the the only difference is the number of replicas

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  replicas: 10
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: smt.example.com:5000/sles12sp3_nginx
          ports:
            - containerPort: 80
```

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

**kubectl apply -f ~/STW-CaaSPv4/labs/nginx-scale.yaml**

3. Enter the following command to view the deployments:

**kubectl get deployments**

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

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

**kubectl get pods**

You should see 10 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)

## 4 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.

## 4- 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 **~/STW-CaaSPv4/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: rmt.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 **~/STW-CaaSPv4/labs/nginx-pre-update.yaml** or the following command line:  
**kubectl apply -f ~/STW-CaaSPv4/labs/nginx-pre-update.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 ~/STW-CaaSPv4/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 may 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)

---

## **4- 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 **~/STW-CaaSPv4/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: rmt.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 ~/STW-CaaSPv4/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)

## 4- 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 Workstation, in the text editor of your choice, open the file:

**~/STW-CaaSPv4/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 ~/STW-CaaSPv4/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)

## 4- 4 Setup Readiness and Liveness Probes

---

### Description:

In this exercise, you learn how to setup and configure Readiness and Liveness Probes

---

### Task 1: View manifest for the Deployment

1. In the text editor of your choice, open the **~/STW-CaaSPv4/labs/nginx-deployment-health.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/sles12sp3_nginx
          ports:
            - containerPort: 80
          readinessProbe:
            httpGet:
              path: /
              port: 80
            initialDelaySeconds: 5
            timeoutSeconds: 1
            periodSeconds: 15
          livenessProbe:
            httpGet:
              path: /
              port: 80
            initialDelaySeconds: 15
            timeoutSeconds: 1
            periodSeconds: 15
```

2. Notice the ReadinessProbe and LivenessProbe settings

## Task 2: Deploy Manifest

1. Deploy the manifest

**kubectl apply -f ~/STW-CaaSPv4/labs/nginx-deployment-health.yaml**

## Task 3: View the Replica sets

2. On the lefthand side of the Kubernetes Dashboard, under Workload, select Replica Sets

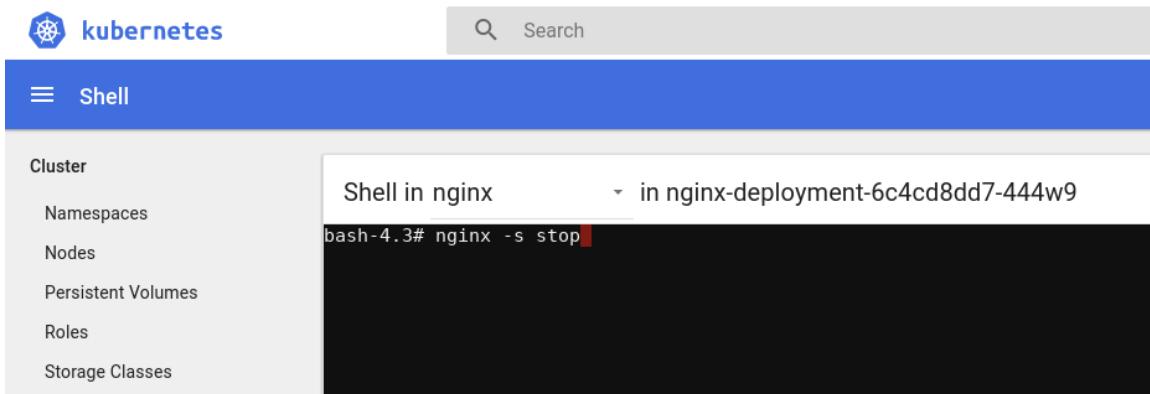
Name	Node	Status	Restarts	Age	CPU (cores)	Memory (bytes)
nginx-deploymer	worker12	Running	1	an hour	0	9.359 Mi
nginx-deploymer	worker11	Running	0	an hour	0	9.504 Mi
nginx-deploymer	worker10	Running	1	an hour	0	9.348 Mi
nginx-deploymer	worker10	Running	0	an hour	0	9.355 Mi

3. Notice the number of Restarts on the container

## Task 4: Kill an nginx server

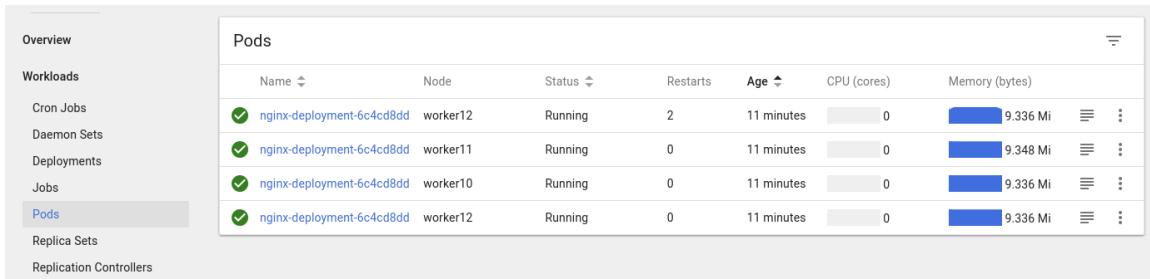
1. Select one of the pods labeled nginx-deployment-#### by clicking on it
2. Click the EXEC button to be taken to a shell prompt inside the running pod

- From the Shell enter the following command to kill the nginx process  
**nginx -s stop**



The screenshot shows the Kubernetes UI with a sidebar on the left containing 'Cluster', 'Namespaces', 'Nodes', 'Persistent Volumes', 'Roles', and 'Storage Classes'. The main area is titled 'Shell' and shows a terminal session in a pod named 'nginx-deployment-6c4cd8dd-444w9'. The command 'bash-4.3# nginx -s stop' is visible in the terminal window.

- Click on Pods to view the current pods running



Overview	Pods						
Workloads	Name	Node	Status	Restarts	Age	CPU (cores)	Memory (bytes)
Cron Jobs	nginx-deployment-6c4cd8dd	worker12	Running	2	11 minutes	0	9.336 Mi
Daemon Sets	nginx-deployment-6c4cd8dd	worker11	Running	0	11 minutes	0	9.348 Mi
Deployments	nginx-deployment-6c4cd8dd	worker10	Running	0	11 minutes	0	9.336 Mi
Jobs	nginx-deployment-6c4cd8dd	worker12	Running	0	11 minutes	0	9.336 Mi
<b>Pods</b>							
Replica Sets							
Replication Controllers							

- Notice the number of restarts has increased. This is because Kubernetes could no longer reach the server on port 80 so it restarted that Nginx Container inside the Pod

---

### 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)

## 4- 5 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 **Workstation**, 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:

**~/STW-CaaSPv4/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 ~/STW-CaaSPv4/labs/limits.yaml --namespace=limit-example
```

You should see the limitrange "mylimits" was created.

2. Enter the following command to display the limitranges:

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

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

Name:	mylimits							
Namespace:	limit-example	Type	Resource	Min	Max	Default Request	Default Limit	Max Limit/Request Ratio
---	-----	---	----	---	---	-----	-----	-----
Pod	cpu	200m	2	-	-	-	-	-
Pod	memory	6Mi	1Gi	-	-	-	-	-
Container	cpu	100m	2	200m	300m	-	-	-
Container	memory	3Mi	1Gi	100Mi	200Mi	-	-	-

---

### 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)

## 4- 6 Introduction to Helm

---

### Description:

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

---

### Task 1: 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)**

Your 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 dokewiki**

You should see the dokewiki chart listed.

5. Enter the following command to display the list of currently configured repos:

**helm repo list**

You should see a repo for <https://kubernetes-charts.storage.googleapis.com>, [kubernetes-charts.suse.com](https://kubernetes-charts.suse.com) and an 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.

SUSE U Advanced - SUSE CaaS Platform

(End of Exercise)

## 4- 7 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 **Workstation**, 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 **~/STW-CaaSPv4/labs/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 ~/STW-CaaSPv4/labs/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 **dokewiki** 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 **Workstation**, 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**):

**images: 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 it was updated. Finally you deleted the release.

(End of Exercise)

## 5 Deploying Advanced Workload

---

### Description:

How to deploy a workload from an existing Docker Image

## 5- 1 Run Heimdall as a standalone container

---

### Description:

In this exercise, you configure and run Heimdall as standalone container

---

### Task 1: Create a place for Heimdall to store it's files

1. On the Workstation, enter the following commands in a terminal session to create the ~/docker/heimdall folder:

```
sudo mkdir -p /docker/heimdall
sudo chmod 777 /docker -R
cd /docker/heimdall
```

The directory and file should now exist.

### Task 2: Run Heimdall as a container

1. On the Workstation, enter the following commands in a terminal session to create

```
sudo docker run -p 80:80 -v /docker/heimdall:/config -e
TZ=Europe/London smt.example.com:5000/heimdall
```

This tells docker to run the heimdall container with the following options:

**Map port 80 in the container to 80 on the host**

**mount ~/docker/heimdall in the container in /config**

**Set the timezone within the container**

2. Notice that the first time it runs it created a default installation

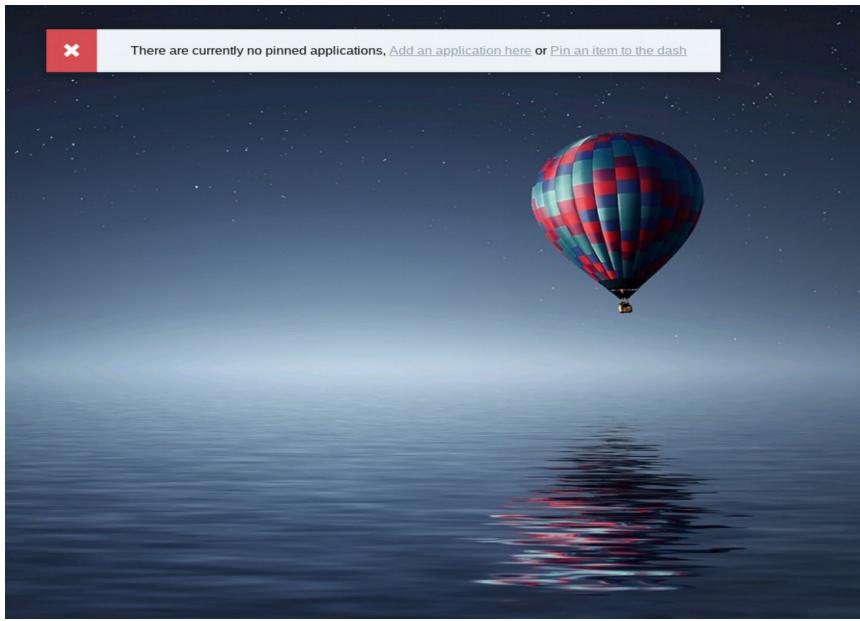
```
-----  
      ()  
     /--\ | | /--\ | |  
    \__\ \_\ \_\ \_\ /  
  
Brought to you by linuxserver.io  
We gratefully accept donations at:  
https://www.linuxserver.io/donate/  
-----  
GID/UID  
-----  
  
User uid: 911  
User gid: 911  
-----  
  
[cont-init.d] 10-adduser: exited 0.  
[cont-init.d] 20-config: executing...  
[cont-init.d] 20-config: exited 0.  
[cont-init.d] 30-keygen: executing...  
generating self-signed keys in /config/keys, you can rep  
Generating a RSA private key  
.....+++++  
.....  
writing new private key to '/config/keys/cert.key'  
----  
[cont-init.d] 30-keygen: exited 0.  
[cont-init.d] 50-config: executing...  
New container detected, installing Heimdall  
Creating app key. This may take a while on slower system  
Application key set successfully.  
Setting permissions  
[cont-init.d] 50-config: exited 0.  
[cont-init.d] 99-custom-scripts: executing...  
[custom-init] no custom scripts found exiting...  
[cont-init.d] 99-custom-scripts: exited 0.  
[cont-init.d] done.  
[services.d] starting services  
[services.d] done.  
■
```

### Task 3:On the Workstation launch the Chrome Browser

- 1 On the Workstation **launch the Chrome Browser**

<http://127.0.0.1>

- 2 You should see a default screen with no Apps Defined



## Task 4: Create an App in Helm

1. Click on 'Add an application here'

**Application Name:** SUSE

**URL:** <http://www.suse.com>

**Select:** Pinned

The screenshot shows the 'Add application' dialog box. At the top, there is a header with the title 'Add application' and three buttons: 'PINNED' (which is selected), 'SAVE', and 'CANCEL'. Below the header, there are four input fields: 'Application name \*' with the value 'SUSE', 'Application Type \*' with the value 'None' (selected from a dropdown), 'Colour \*' with the value 'Hex colour', and 'URL' with the value '<http://www.suse.com>'. There is also a 'Tags (Optional)' field and a 'Upload a file' button. In the bottom left corner of the dialog, there is a preview section labeled 'Preview' which shows a small thumbnail of the SUSE logo. At the very bottom of the dialog, there are three buttons: 'PINNED', 'SAVE', and 'CANCEL'.

2. Click Save to Save the App

## Task 5: Stop and restart container

1. On the Workstation, go back to the terminal session running the container and press the following Keys to stop the container instance

**<ctrl>C**

This will terminate the container instance

2. Restart the Docker Container

```
sudo docker run -p 80:80 -v ~/docker/heimdall:/config -e  
TZ=Europe/London smt.example.com:5000/heimdall
```

Notice How it start also immediately because it's using the configuration from the last time we launched the Application

## Task 6: Verify with Browser

1. On the Workstation launch the Chrome Browser

<http://12.0.0.1>

2. You should see the App you previously defined

## Task 7: Stop Container Instance

1. On the Workstation, go back to the terminal session running the container and press the following Keys to stop the container instance

**<ctrl>C**

This will terminate the container instance

## Task 8: Try and run multiple versions of Heimdall

1. On the Workstation, go back to the terminal session running the container and press the following Keys to stop the container instance

```
sudo docker run -d -p 80:80 -v ~/docker/heimdall:/config -e  
TZ=Europe/London smt.example.com:5000/heimdall
```

**\* notice we added the -d command to tell docker to run as a Daemon rather than in the foreground**

1. Try and run a second instance of the Heimdall

```
sudo docker run -d -p 80:80 -v ~/docker/heimdall:/config -e TZ=Europe/London smt.example.com:5000/heimdall
```

**Notice is fails because the port is already in use**

---

### **Summary:**

In this exercise, you launched a container from the command line to see how it behaved

(End of Exercise)

## 5- 2 Launch Heimdall

---

### Description:

In this exercise, we will Heimdall without dedicated storage

---

### Task 1: Create the Manifest for the Persistent Volume

1. On the Workstation, in the text editor of your choice, open the file:

**~/STW-CaaSPv4/labs/heimdall/app-health/heimdall-deployment\_ns.yaml**

Enter the following in the file:

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: heimdall-shared-deployment
spec:
  replicas: 1
  template:
    metadata:
      labels:
        app: heimdall-shared
    spec:
      containers:
        - name: heimdall-pod
          image: linuxserver/heimdall
          ports:
            - containerPort: 80
```

2. On the Workstation, in the text editor of your choice, open the file:

**~/STW-CaaSPv4/labs/heimdall/app-health/heimdall-service.yaml**

Enter the following in the file:

```
apiVersion: v1
kind: Service
metadata:
  name: heimdall-shared-service
spec:
  type: NodePort
  ports:
    - port: 80
      nodePort: 33000
  selector:
    app: heimdall-shared
```

## Task 2: Deploy Heimdall

1. From a Terminal prompt on the Management workstation

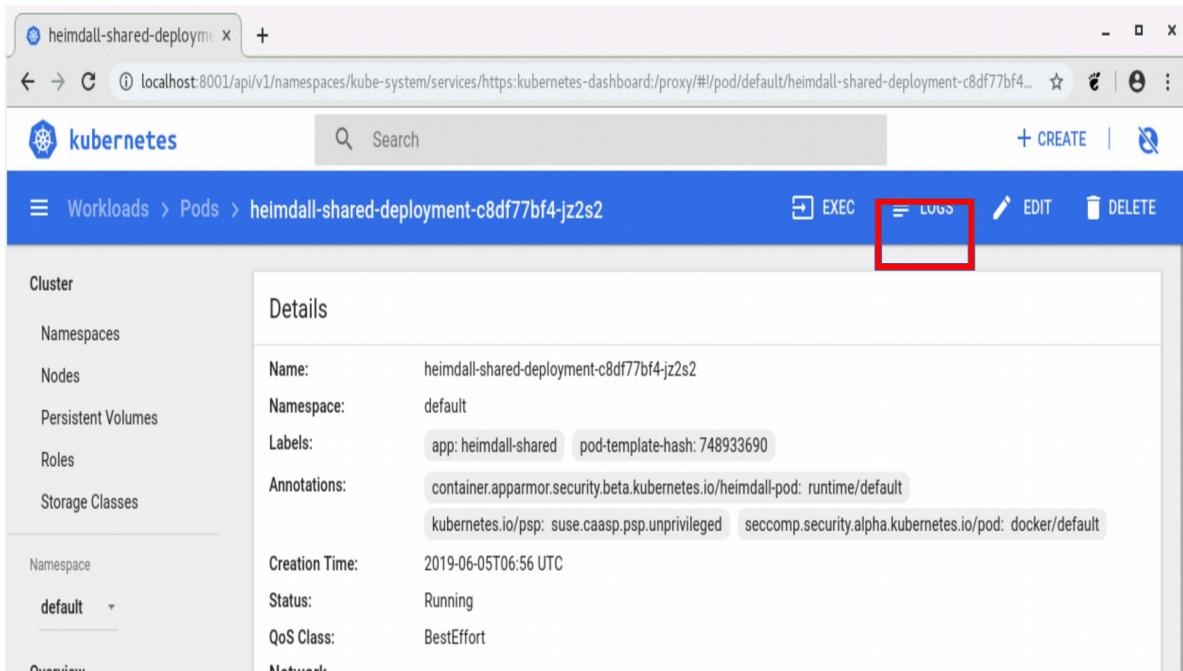
```
kubectl apply -f ~/STW-CaaSPv4/labs/heimdall/app-health/heimdall-deployment_ns.yaml
```

```
kubectl apply -f ~/STW-CaaSPv4/labs/heimdall/app-health/heimdall-service.yaml
```

## Task 3: View the Deployment

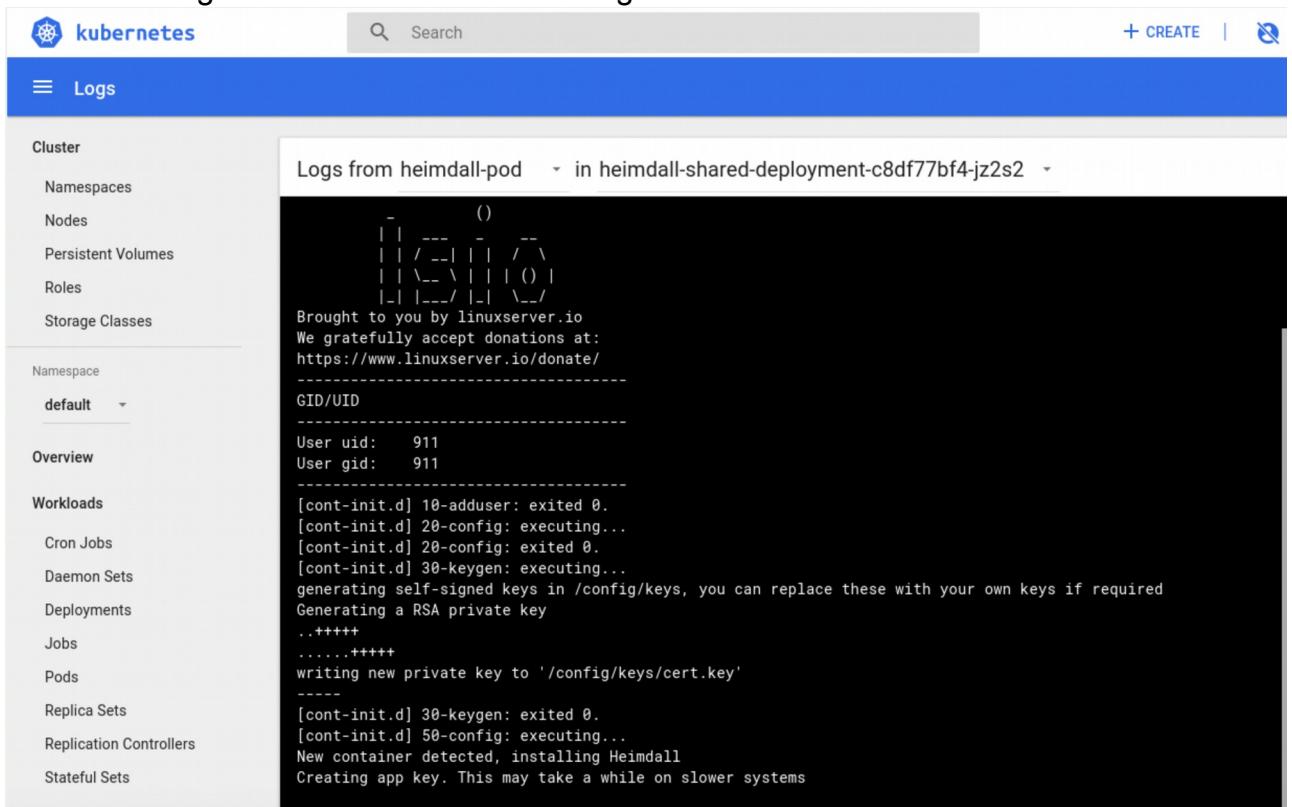
1. From Kubernetes Dashboard select Pods and then click on the Pod that was just deployed
2. Click on the Log button so we can see what is happening on that Pod

## SUSE U Advanced - SUSE CaaS Platform



The screenshot shows the Kubernetes dashboard interface. On the left, there's a sidebar with 'Cluster' and 'Namespaces' sections. In the main area, under 'Workloads > Pods', a pod named 'heimdall-shared-deployment-c8df77bf4-jz2s2' is selected. At the top right of this section are buttons for 'EXEC', 'LOGS' (which is highlighted with a red box), 'EDIT', and 'DELETE'. Below these buttons, there's a small preview window showing some log entries.

3. Look at the logs and notice how it is creating a new installation



The screenshot shows the 'Logs' page for the 'heimdall-shared-deployment-c8df77bf4-jz2s2' pod. The left sidebar includes 'Cluster', 'Namespaces', and 'Nodes' sections, along with 'Overview', 'Workloads', 'Cron Jobs', 'Daemon Sets', 'Deployments', 'Jobs', 'Pods', 'Replica Sets', 'Replication Controllers', and 'Stateful Sets'. The main area displays the logs from the 'heimdall-pod'. The logs show the deployment process, including generating RSA private keys and installing Heimdall.

```
Logs from heimdall-pod in heimdall-shared-deployment-c8df77bf4-jz2s2
-----
()_
| | _--_ | _ / --\_
| | / _-| | | | () |
| | \_--\ | | | \_/
|_| _--/_ |_| \_/
Brought to you by linuxserver.io
We gratefully accept donations at:
https://www.linuxserver.io/donate/
-----
GID/UID
-----
User uid: 911
User gid: 911
-----
[cont-init.d] 10-adduser: exited 0.
[cont-init.d] 20-config: executing...
[cont-init.d] 20-config: exited 0.
[cont-init.d] 30-keygen: executing...
generating self-signed keys in /config/keys, you can replace these with your own keys if required
Generating a RSA private key
.....+
writing new private key to '/config/keys/cert.key'
-----
[cont-init.d] 30-keygen: exited 0.
[cont-init.d] 50-config: executing...
New container detected, installing Heimdall
Creating app key. This may take a while on slower systems
```

4. Let that deployment continue to run until it is completely finished (Refresh the Browser to check Status)

## Task 4: View Heimdall in a Browser

1. On the **Workstation** open the **Chrome browser** and go to  
**worker.example.com:33000**
  
2. Notice how ever time it is deployed it has to rebuild the installation
  
3. Stop the deployment by either killing it on the Kubernetes Dashboard from the command line by typing in the following command

**kubectl delete -f ~/STW-CaaSPv4/labs/heimdall/app-health/heimdall-deployment\_ns.yaml**

## Task 4: Re-deploy Heimdall

1. Repeat all of the steps in **Task 2 and Task 3**
2. Notice how ever time it is deployed it has to rebuild the installation

---

### Summary:

In this exercise, you created manifests for a Heimdall. We then deployed it and watched it build the default configuration. We learned that every time we deployed the app it had to create the default installation.

(End of Exercise)

## 5- 3 Configure Heimdall to use NFS Persistent Storage

---

### Description:

In this exercise, you configure a persistent volume on an NFS server. You then launch a pod that use the persistent volume

---

### Task 1: Create the Manifest for the Persistent Volume

1. On the Workstation, in the text editor of your choice, open the file:

**`~/course_files/CAAS101/course_files/manifests/labs/heimdall/app-shared/heimdall-pv-shared.yaml`**

Enter the following in the file:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: heimdall-shared
spec:
  capacity:
    storage: 200Mi
  accessModes:
    - ReadWriteMany
  nfs:
    server: 192.168.110.2
    path: "/export/heimdall-shared"
```

### Task 2: Create the Manifest for the Persistent Volume Claim

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

**`~/course_files/CAAS101/course_files/manifests/labs/heimdall/app-shared/heimdall-pvc-shared.yaml`**

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: heimdall-shared-claim
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 200Mi
```

### Task 3: Create the Manifest for the Service so we can access the deployment

1. In the text editor of your choice, open the file:
2. **~/course\_files/CAAS101/course\_files/manifests/labs/heimdall/app-shared/heimdall-service.yaml**

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

### Task 4: Create the Manifests for the Web Frontend

1. In the text editor of your choice, create/open the file:
2. **~/course\_files/CAAS101/course\_files/manifests/labs/heimdall/app-shared/heimdall-deployment.yaml**

```

apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: heimdall-shared-deployment
spec:
  replicas: 1
  template:
    metadata:
      labels:
        app: heimdall-shared
    spec:
      containers:
        - name: heimdall-pod
          image: linuxserver/heimdall
          volumeMounts:
            - name: heimdall-pvc-shared
              mountPath: "/config"
      volumes:
        - name: heimdall-pvc-shared
          persistentVolumeClaim:
            claimName: heimdall-shared-claim

```

## Task 5: Deploy the Objects

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

**kubectl apply -f ~/STW-CaaSPv4/labs/heimdall/app-shared**

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

deployment.extensions "heimdall-shared-deployment" created  
 persistentvolume "heimdall-shared" created  
 persistentvolumeclaim "heimdall-shared-claim" created  
 service "heimdall-shared-service" created

2. Enter the following command to view the deployments:

**kubectl get deployments**

You should see the **heimdall-shared-deployment** deployments listed.

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

**kubectl get pv**

You should see the persistent volume **heimdall-shared** listed.

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

**kubectl get pvc**

You should see the persistent volume claim **heimdall-shared-claim** listed.

## Task 6: Test the Persistent Data

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

**http://worker10.example.com:30000**

Notice we have a Heimdall session pre-populated with 2 Applications

## Task 7: Remove the Objects from the Cluster

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

**kubectl delete -f ~/STW-CaaSPv4/labs/heimdall/app-shared**

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)

## 6 Appendix: Bonus Labs

---

### Description:

Bonus Labs you can play with if you have time

## 7 Horizontal Scale

---

### **Description:**

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

## 7- 1 Configure Horizontal Pod Autoscaling

---

### Description:

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

#### Note 1:

If you do not wish to type in all of the manifests you are instructed to create, they have been pre-created and can be found in the `~/STW-CaaSPv4/labs/hpa/` directory.

#### Note 2:

If desired, there is a script in `~/course_files/CAAS101/scripts/tmux/` 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: Install Metric Server

1. On the **management workstation**, at the command prompt, enter the following commands to apply RBAC Roles on the Metric Server in your cluster:

```
cd ~/STW-CaaSPv4/labs/hpa/  
kubectl apply -f metrics/metric-rbac.yaml
```

2. Create a Helm Template for the Metric Server

```
helm inspect values stable/metrics-server > metrics_values.yaml
```

3. Turn off authentication by entering the following command

```
sed -i 's/^args\|:/&\n - --authentication-skip-lookup/' metrics_values.yaml
```

4. Install the Metrics Sever via Helm

```
helm install stable/metrics-server --namespace=kube-system --values metrics_values.yaml
```

## Task 2: View the Manifest for the App to be Scaled

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

**~/STW-CaaSPv4/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 3: View the Manifest for the App's Service

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

**~/STW-CaaSPv4/labs/hpa/app/php-apache-service.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 4: View the Manifest for the Autoscaler

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

**~/STW-CaaSPv4/labs/hpa/app/php-apache-autoscaler.yaml**

```
apiVersion: autoscaling/v2beta1
kind: HorizontalPodAutoscaler
metadata:
  name: php-apache
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: php-apache
  minReplicas: 1
  maxReplicas: 10
  targetCPUUtilizationPercentage: 80
```

### Task 5: Deploy the AutoScaler Objects

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

## kubectl apply -f ~/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:

```
cd ~/course_files/CAAS101/scripts/tmux/
```

```
sh ./tmux-watch-kubectl.pod-deployment-hpa.sh
```

This script uses **tmux** to split your terminal into 4 quadrants and then runs the three watch commands specified in three of the panes leaving one pane to enter the other commands.

3. 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**

## Task 6: Create the Manifest for the Load Generator

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

**~/hpa/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:
        app: load-generator
    spec:
      containers:
        - image: smt.example.com:5000/busybox
          name: busybox
          command: ["/bin/sh"]
          args: ["-c", "while true; do wget -q -O- http://php-apache.default.svc.cluster.local; done"]
```

## Task 7: 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 ~/STW-CaaSPv4/labs/hpa/load-generator.yaml**

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 8: 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 deployment load-generator**

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 9: Clean Up the Deployments

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

**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)

## 7- 2 CaaS Platform Network and Logs

---

### Description:

In this exercise you will review Kubernetes events, pod logs, and gather all static logs from an entire node.

### Note:

This lab contains exercises using both back-ticks ` and apostrophes ' . Be careful about the punctuation in the exercise.

---

### Task 1: Display all Flannel networks

1. On the **management workstation**, in a terminal, enter the following command to get a list of networks:

**kubectl describe node | grep -A 1 PodCIDR:**

You should see one network range in CIDR format for each node and its name.

### Task 2: Deploy Counting Pod

1. On the **management workstation**, open a terminal and enter the following command to deploy counter.yaml

**kubectl create -f \  
~/course\_files/CAAS101/manifests/labs/counter.yaml**

This will create a pod that will count up once per second and output the count to the system log

2. Confirm that the pod is running

**kubectl get pod counter**

You should now see the new pod running

### Step 3: Review Kubernetes Events

1. On the **management workstation**, open a terminal and enter the following command to view the latest events in your kubernetes cluster.

**kubectl get events**

You should see the latest events where you added the counting pod

2. Enter the following command to view the latest events in chronological order, (use back ticks):

## **kubectl get events --sort-by={.firstTimestamp}**

### Step 4: View Container Logs

1. On the **management workstation**, open a terminal and enter the following command to review the container logs for the counter pod.

#### **kubectl logs counter**

You should see repeating timestamps in the pod's system log

2. View pods in the **kube-system** namespace.

#### **kubectl get pods -n kube-system**

You should see several system pods already there

3. Review the container logs on the first flannel pod:

#### **kubectl logs -n kube-system kube-flannel-XXXXX**

The last five character are random and specific to that server. As you can see, flannel's job includes updating the networks and ip tables.

### Task 5: Gather Supportconfig logs

1. Log into the **admin** node:

**ssh root@192.168.110.99**

The password is **linux**

2. Create and enter a directory to hold the **supportconfig** files:

**mkdir support && cd support**

3. Generate a **supportconfig** for every node:

```
docker exec `docker ps -q \
--filter name=salt-master` \
salt '*' cmd.run '/sbin/supportconfig'
```

This will take some time

4. Copy the supportconfig files into the support directory:

```
for i in {7..13}; do scp \
192.168.110.$i:/var/log/nts* /root/support/; \
done && cp /var/log/nts* /root/support/
```

You may need to answer yes if prompted to allow ssh to the new nodes.

5. Confirm that you have two files from every server. The first is the actual log file. The second is a md5 checksum.

**ls -l /root/support**

## Task 6: Review Supportconfig

1. On the management workstation, in a terminal and log into the admin node:

**ssh root@192.168.110.99**

The password is **linux**

Enter the **support** directory and expand the supportconfig file for the admin node:

**cd support**

**tar xfvj nts\_admin\_xxxxxx.tbz**

Each file is timestamped. Replace the example filename with your own.

2. Confirm that the file are in the newly created directory:

**ls -l nts\_admin\_xxxxxx/**

At this point, you can use the editor of your choice to review the contents of the log files.

---

### Summary:

In this exercise, you reviewed the network for your cluster, reviewed the Kubernetes events for your entire cluster, reviewed the logs for individual pods, and gathered system and kubernetes logs for your entire cluster using Supportconfig.

(End of Exercise)

## 7-3 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)

## 7-4 Configure DNS for CaaSP

---

### Description:

In this exercise, you configure DNS name resolution for the CaaSP cluster.

#### Note 1:

In a real life deployment the DNS server does not need to be configured on the SMT server specifically but it does need to be configured somewhere. In this lab environment we are consolidating the infrastructure services such as DNS on the SMT server for simplicity.

#### Note 2:

There is a script in `~/course_files/COURSE_ID/scripts/` on the management workstation and in `/root/bin/` on the admin node named **add-node-to-pxe.sh** that automates many of the steps in Task 2 of this exercise. It also automates many of the steps in Task 5 and Task 6 of the Configure PXE for CaaSP Deployment exercise. Running this script on the admin node with the appropriate arguments creates the PXE boot files, DHCP and DNS entries for the new node.

#### Note 3:

Depending on your lab environment the DNS server may already be installed and configured. If this is the case you may skip this exercise.

---

### Task 1: Install DHCP and DNS

1. Log into the SMT server as the **root** user
2. In a terminal enter the following command to install the DHCP and DNS Server pattern:

**zypper in -y -t pattern dhcp\_dns\_server**

The DHCP and DNS Server pattern should have been installed.

### Task 2: Configure DNS for CaaSP

1. Launch the DNS Server YaST module:

**yast2 dns-server**

2. On the DNS Server Installation: Forwarder Settings screen enter/select the following:

**Local DNS Resolution Policy: Automatic merging**

**Local DNS Resolution Forwarder: This name server (bind)**

(The Forwarder list should already be populated with the DNS servers configured when networking on the SMT server was configured.)

Click **Next**

3. On the DNS Server Installation: DNS Zones screen, under **Add New Zone**,

Enter **example.com**

Select type: **Master**

And then click **Add**

4. Under **Configured DNS Zones**, select the **example.com** zone and click **Edit**

5. On the **NS Records** tab, in the **Name Server to Add** field, enter:

**smt.example.com**

The click **Add**

You should see the server listed in the Name Server List.

6. On the **Records** tab, under **Record Settings**, enter/select the following to create a record for the SMT server:

**Record Key: smt.example.com**

**Type: A: IPv4 Domain Name Translation**

**Value: 192.168.110.2**

Click **Add**

7. Repeat the previous step for each of the nodes (**admin**, **master01**, **worker10**, **worker11**, etc) replacing the **hostnames** and **IP addresses** for those that match the node

8. To configure Round Robin DNS for a multi-master configuration, on the **Records** tab, under **Record Settings**, enter/select the following to create a record for the SMT server:

**Record Key: master.example.com**

**Type: A: IPv4 Domain Name Translation**

**Value: CAASP\_MASTER01\_IP**

9. Click **Add**

10. Repeat the previous step for the **master02** and **master03** nodes replacing the value with **CAASP\_MASTER02\_IP** and **CAASP\_MASTER03\_IP**.

11. Click **OK**

12. To configure Round Robin DNS for the worker nodes, on the **Records** tab, under **Record Settings**, enter/select the following to create a record for the SMT server:

**Record Key: worker.example.com**

**Type: A: IPv4 Domain Name Translation**

**Value: CAASP\_WORKER10\_IP**

13. Click **Add**
14. Repeat the previous step for the **woker11**, **worker12** and **worker13** nodes replacing the value with **CAASP\_WORKER11\_IP** , **CAASP\_WORKER12** and **CAASP\_WORKER13\_IP** .
15. Click **OK**
16. Back on the **DNS Server Installation: DNS Zones** screen click **Next**
17. On the **DNS Server Installation: Finish Wizard** screen, under **Start-up Behavior**, select: **On: Start Now and When Booting**
18. Click **Finish**

---

**Summary:**

In this exercise, you configure a new DNS zone for the CaaSP cluster (example.com) and then populated the zone file with the hostnames and IP addresses of the CaaSP cluster nodes including Round Robin DNS name resolution for a K8s multi-master configuration and the worker nodes.

(End of Exercise)

## 7-5 Install and Configure DHCP for CaaSP

---

### Description:

In this exercise, you install and configure DHCP for use in deploying and running SUSE Containers as a Service Platform.

#### Note 1:

In a real life deployment the DHCP server does not need to be configured on the SMT server specifically but it does need to be configured somewhere. In this lab environment we are consolidating the infrastructure services such as DHCP on the SMT server for simplicity.

#### Note 2:

Depending on your lab environment the DHCP server may already be installed and configured. If this is the case you may skip this exercise.

---

### Task 1: Install DHCP

1. Log into the **SMT** server as the **root** user
2. In a terminal enter the following command to install the DHCP and DNS Server pattern:

**zypper in -y -t pattern dhcp\_dns\_server**

The DHCP and DNS Server pattern should have been installed.

### Task 2: Configure DHCP for CaaSP

1. Launch the DHCP Server YaST module:

**yast2 dhcp-server**

2. On the first **DHCP Server Wizard** screen, from the list of network cards, select the first network card (probably eth0) and click: **Select**

Then click **Next**

3. On the second **DHCP Server Wizard** screen enter/select the following:

**Domain Name: example.com**

**Primary Name Server: 192.168.110.2**

**Secondary Name Server: (leave empty)**

**Default Gateway (Router): 192.168.110.1**

**NTP Server: 192.168.110.2**

(Accept all other defaults)

Then click **Next**

4. On the third **DHCP Server Wizard** screen enter/select the following:

**IP Address Range: First IP Address: 192.168.110.100**

**IP Address Range: Last IP Address: 192.168.110.199**

(Accept all other defaults)

Then click **Next**

5. On the fourth **DHCP Server Wizard** screen, select **Service Start: When Booting**

Then click **Finish**

---

### **Summary:**

In this exercise, you configured a basic DHCP server for the CaaSP cluster.

(End of Exercise)

