

# Running Nightwatch at NERSC with Rancher2

Ana Lyons

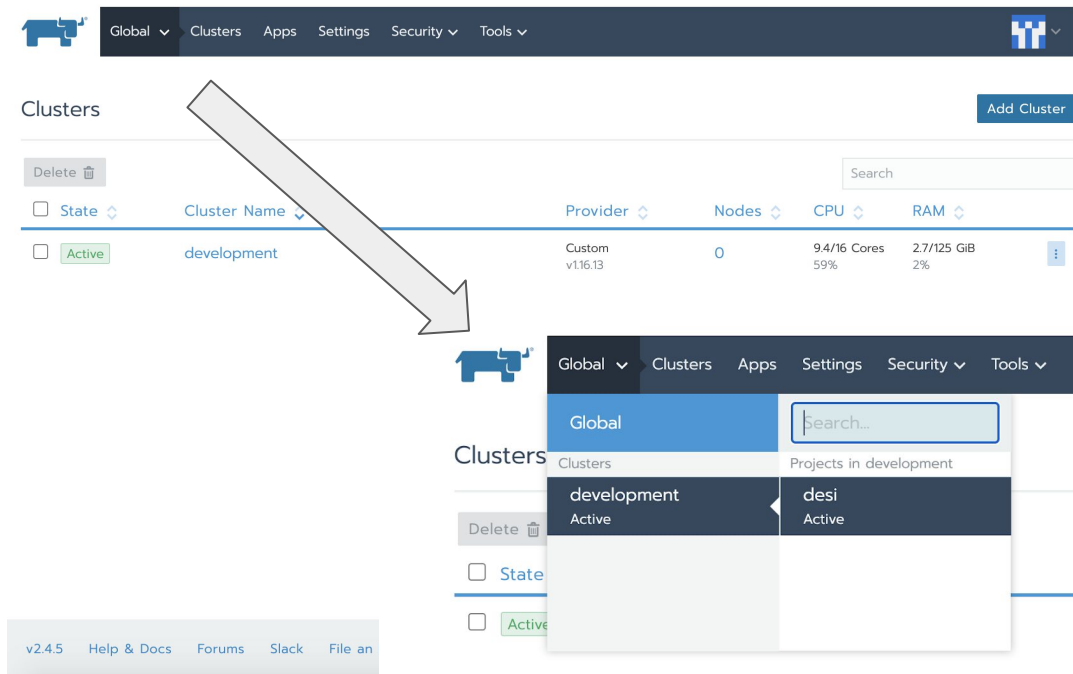
# Structure

Rancher2 uses Kubernetes instead of Docker, so the structure of the web app is slightly different than previously;

Instead of writing a docker-compose.yml file to specify the interaction between a uWSGI-flask container and a nginx container, we will write a kubernetes deployment yaml file that configures one container/workload, and a separate yaml file to configure a load-balancer/ingress, which will act in the same way our nginx docker container did.

We can do much of this “writing” through the rancher2 GUI.

# Rancher 2 at Spin



The screenshot shows the Rancher 2 web interface. The top navigation bar includes 'Global', 'Clusters', 'Apps', 'Settings', 'Security', and 'Tools'. The 'Clusters' page displays a table of clusters. A large arrow points from the 'development' cluster in the table to a detailed view of that cluster. The detailed view shows the 'development' cluster is 'Active' and contains a 'desi' project in development.

State	Cluster Name	Provider	Nodes	CPU	RAM
Active	development	Custom v1.16.13	0	9.4/16 Cores 59%	2.7/125 GiB 2%

From the detailed view, the 'Clusters' section shows the 'development' cluster is 'Active' and contains a 'desi' project in development.

From the homepage, rancher2.spin.nersc.gov, we want to navigate to the development cluster, and the desi project inside that cluster.

This is what the desi project homepage looks like- there is a list of all the workloads running in each “namespace” (sub-projects of the desi project). We can work in the nightwatchtest namespace, or create our own when we start creating a new workload.

The screenshot shows the desi project homepage. At the top, there is a dark blue header bar with the desi logo (a blue elephant) on the left. Next to it is a dropdown menu labeled "development desi". To the right of this are several navigation links: "Resources", "Apps", "Namespaces", "Members", and "Tools". On the far right of the header is a blue and white checkered icon with a dropdown arrow. Below the header, there is a row of tabs: "Workloads" (which is selected and underlined), "Load Balancing", "Service Discovery", and "Volumes". To the right of these tabs are several icons for different actions: a cube, a list, a refresh, a menu, and a button labeled "Import YAML". To the right of these is a blue button labeled "Deploy". Below the tabs, there is a row of action buttons: "Redeploy" (with a refresh icon), "Pause Orchestration" (with a pause icon), "Download YAML" (with a download icon), and "Delete" (with a trash icon). To the right of these buttons is a search bar labeled "Search". Below the action buttons, there is a table with columns: "State", "Name", "Image", and "Scale". The table has a header row with these columns. Below the header row, there is a section labeled "Namespace: nightwatchtest". In this section, there is one workload entry. The entry has a checkbox, a play button, a green box labeled "Active", a name "app" with a refresh icon and "80/http" below it, an image "registry.nersc.gov/desi/nightwatch/app-uwsgi-flask:latest", and a scale bar with the number "1" and a dropdown arrow. Below the workload entry, there is a footer row with the text "1 Pod / Created 19 days ago / Pod Restarts: 0".

development  
desi

Resources Apps Namespaces Members Tools

Workloads Load Balancing Service Discovery Volumes

Redeploy Pause Orchestration Download YAML Delete

Search

State Name Image Scale

Namespace: nightwatchtest

Active app 80/http registry.nersc.gov/desi/nightwatch/app-uwsgi-flask:latest 1 Pod / Created 19 days ago / Pod Restarts: 0

# Getting started; creating a deployment

Service Discovery Volumes

Import YAML **Deploy**

Download YAML Delete Search

Image Scale

regist 1 Pod

development desi Resources Apps Namespaces Members Tools

## Deploy Workload

Name \* e.g. myapp Workload Type Scalable deployment of 1 pod [More options](#)

Docker Image \* registry.nersc.gov/desi/nightwatch/app-uwsgi-flasklatest Namespace \* nightwatchtest [Add to a new namespace](#)

Port Mapping

Port Name	Publish the container port *	Protocol	As a	On listening port *
e.g. myport	e.g. 8080	TCI	NodePort (On every node)	Random

+ Add Port

# Base configurations

## Deploy Workload

Name \*

name for service goes here (ex. nightwatch-app, etc)

Workload Type

 Scalable deployment of 1 pod

[More options](#)

Docker Image \*

Use this docker image as the base

registry.nersc.gov/desi/nightwatch/app-uwsgi-flask:latest



Namespace \*

Can switch to a new namespace if you want

nightwatchtest



[Add to a new namespace](#)

Port Mapping

Port Name

Publish the container port \*

Protocol

As a

On listening port \*

pick a name for the port

5000

TCF 

Cluster IP (Internal only) 

Same as container port 



 Add Port

# Configuring options: mounting volumes

Below the base configurations, there are a bunch of dropdown menus containing all our other configuration options. First, we will deal with mounting volumes.

## ▶ Environment Variables

Set the environment that will be visible to the container, including injecting values from other resources like Secrets.

## ▶ Node Scheduling

Configure what nodes the pods can be deployed to.

## ▶ Health Check

Periodically make a request to the container to see if it is alive and responding correctly.

## ▼ Volumes

Persist and share data separate from the lifecycle of an individual container.

Add Volume... ▼

Click here to add a new volume

## ▶ Scaling/Upgrade Policy

Configure how pods are replaced when performing an upgrade.

# Adding a volume

We want a bind-mount volume, as we will be mounting files from the global file system

Need to name the volume, specify a mount point inside the container, and specify exactly what directory on the global file system is being mounted. Pick “an existing directory” to make sure Rancher verifies the directory exists.  
For most volumes, we will want the read-only option, as well.

- ▶ Add an ephemeral volume
- ▶ Add a new persistent volume (claim)
- ▶ Use an existing persistent volume (claim)
- ▶ Bind-mount a directory from the node
- ▶ Use a secret
- ▶ Use a config map
- ▶ Use a certificate
- ▶ Add Volume... ▼

Volume Name

static

Volume Type

Bind-Mount

Path on the Node

Path to directory on global file system

The Path on the Node must be

An existing directory

Mount Point

/app/static

Sub Path in Volume

Read-Only

☐

+ Add Mount

- Remove Volume

Next slide details what volumes we need for a Nightwatch deployment.



# Volumes for Nightwatch

- “Static”- mounted to /app/static in the container. This directory contains static nightwatch html files. Read only is false, as we need to be able to write new html files to this folder when the user interacts with the spectra plotting functionality, for instance.
- “Data”- mounted to /app/data in the container. Directory contains qa-\*.fits, qproc-\*.log, preproc-\*.fits files for use in dynamic nightwatch features. Read only.
- “Nightwatch”- mounted to /app/nightwatch in the container. Directory containing nightwatch code. Read only.
- “Desiutil”- mounted to /app/desiutil in the container. Directory containing desiutil code. Read only.
- “Desimodel”- mounted to /app/desimodel in the container. Directory containing desimodel code.

# Entrypoint command configurations

Entrypoint

uwsgi

Command

--ini app.ini --pyargv 'webapp -i ./static -d ./data'

Working Dir

/app

User ID

80355

Console

- ☐ Interactive & TTY (-i -t)
- ☐ Interactive (-i)
- ☐ TTY (-t)
- ☒ None

Auto Restart

- ☒ Always

Filesystem Group

58102

Stop Timeout

30

seconds

The container will have this long to stop on its own before it is forcefully terminated.

Fill out the configuration options so they match the ones to the left

Note: the entrypoint command differs slightly from the one used with docker/rancher1; we don't specify the group id or user id in the command because of how finicky uWSGI is with when UID and GID get assigned.

# Security configurations

Need to drop all capabilities except for NET\_BIND\_SERVICE; otherwise service will not run (it's a NERSC security requirement)

▼ Security & Host Config

Grant or limit the abilities of the container to affect the host it is running on.

Add Capabilities

MKNOD

NET\_ADMIN

NET\_BIND\_SERVICE

NET\_BROADCAST

NET\_RAW

Drop Capabilities

ALL

AUDIT\_CONTROL

AUDIT\_WRITE

BLOCK\_SUSPEND

Now the app pod should be ready to launch...

Launch

Cancel

# If everything worked...

Workloads Load Balancing Service Discovery Volumes

Icons: [Kubernetes Logo] [List Icon] [Refresh Icon] [Menu Icon] [Import YAML] [Deploy]

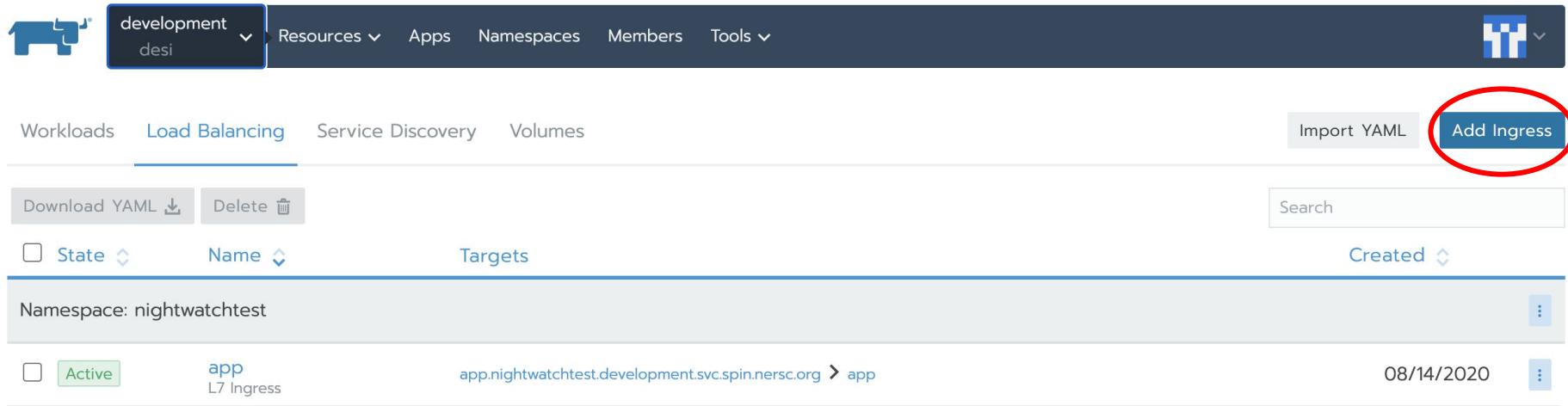
Buttons: Redeploy [Refresh Icon] Pause Orchestration [Pause Icon] Download YAML [Download Icon] Delete [Trash Icon]

Search

<input type="checkbox"/>	State ▾	Name ▾	Image ▾	Scale ▾
Namespace: nightwatchtest [More Icon]				
<input type="checkbox"/>	▶ Active	app  80/http	registry.nersc.gov/desi/nightwatch/app-uwsgi-flask:latest 1 Pod / Created 19 days ago / Pod Restarts: 0	<div><div></div></div> 1 [More Icon]

Now we need to configure a load-balancer to handle traffic from outside the pod

# Configuring a load-balancer



The screenshot shows the Kubernetes Dashboard interface. At the top, the navigation bar includes a logo, a dropdown menu for 'development desi', and links for 'Resources', 'Apps', 'Namespaces', 'Members', and 'Tools'. Below this, the 'Load Balancing' tab is selected in the main navigation area. On the right side of the dashboard, the 'Add Ingress' button is highlighted with a red circle. Below the navigation bar, there are buttons for 'Download YAML' and 'Delete'. A search bar is also present. The main content area displays a table of ingress resources. The first row shows a resource named 'app' in the 'nightwatchtest' namespace, which is 'Active' and was created on '08/14/2020'. The table headers are 'State', 'Name', 'Targets', and 'Created'.

development  
desi

Resources Apps Namespaces Members Tools

Workloads **Load Balancing** Service Discovery Volumes

Import YAML **Add Ingress**

Download YAML Delete

Search

State Name Targets Created

Namespace: nightwatchtest

<input type="checkbox"/>	Active	app L7 Ingress	app.nightwatchtest.development.svc.spin.nersc.org > app	08/14/2020
--------------------------	--------	-------------------	---	------------

Click on the load balancing tab on the homepage, and click add ingress to configure a new one

# Configuring ingress

## Add Ingress

Name [Add a Description](#) Namespace [Add to a new namespace](#)

**Pick a name, perhaps same as workload**

Rules

☐ Automatically generate a `.xip.io` hostname

☒ Specify a hostname to use

☐ Use as the default backend  
Ingress controller does not support default backend

Request Host  **Name that meets requirements**

Target Backend [+ Service](#) [+ Workload](#)

Path  **Don't need to fill out**

Target 

✓ Choose a Workload...

Namespace: nightwatchtest

app

Port  **Put exposed port from workload**

Can specify a hostname to use, but hostname needs to meet the NERSC naming requirements

Example hostname: `app.nightwatchtest.development.svc.spin.nersc.org`

Template: `[name].[namespace].[cluster (development)].svc.spin.nersc.org`

# Success! Hopefully...

If everything goes as it should, on the main page you should see:

Workloads Load Balancing Service Discovery Volumes

Redeploy Pause Orchestration Download YAML Delete

Search

<input type="checkbox"/>	State	Name	Image	Scale
Namespace: nightwatchtest				
<input type="checkbox"/>	▶ Active	app 80/http	registry.nersc.gov/desi/nightwatch/app-uwsgi-flasklatest 1 Pod / Created 19 days ago / Pod Restarts: 0	1

Workloads Load Balancing Service Discovery Volumes

Download YAML Delete

Search

<input type="checkbox"/>	State	Name	Targets	Created
Namespace: nightwatchtest				
<input type="checkbox"/>	Active	app L7 Ingress	app.nightwatchtest.development.svc.spin.nersc.org > app	08/14/2020

# Troubleshooting

If there is a problem with your deployment, you can click on it on the workload homepage and be brought to this page; you can view the logs, and see other metrics of the pod performance.

Workload: app

Active



Namespace: nightwatchtest

Image: registry.nersc.gov/desi/nightwatch/app-uwsgi-flask:latest

Workload Type: Deployment

Endpoints: 80/http

Config Scale: 1



Ready Scale: 1

Created: 08/12/2020

Pod Restarts: 0

Expand All

## ▼ Pods

Pods in this workload

Download YAML

Delete



State

Name

Image

Node



Running

app-574ddc8679-brqtd

registry.nersc.gov/desi/nightwatch/app-uwsgi-flask:latest  
10.42.3.121 / Created 12 days ago / Restarts: 0



Execute Shell

View Logs

View/Edit YAML

View in API

Delete

## ▶ Workload Metrics

Expand to see live metrics

## ▶ Events

Events of current Deployment

## ▶ Environment Variables