Universal Agent 7.2.x

Container Operations

Container Operations

Docker Containers

Red Hat OpenShift Start-Up Guide

Red Hat OpenShift Operator Start-

The information on these pages also is located in Universal Agent 7.0.x User Guide.pdf.

Docker Containers

Downloading a Universal Agent Docker Image

Images are based on RedHat Linux.

To download an image from the Docker Hub, run the following command:

```
docker pull stonebranch/universal-agent:latest
```

To list the downloaded images from the Stonebranch repository, run the following command:

```
docker images -a stonebranch/universal-agent
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
stonebranch/universal-agent	6.6.0.1	bc9dfbf7d367	19 hours ago	1.04GB
stonebranch/universal-agent	latest	bc9dfbf7d367	19 hours ago	1.04GB

These images were created with the following

```
# Set the base image to RedHat UBI
FROM registry.access.redhat.com/ubi8/ubi
# create label
LABEL name="universal-agent"
LABEL vendor="Stonebranch"
LABEL version="${ua_version}"
LABEL release="GA"
LABEL summary="Universal Agent installed in a UBI based Image"
LABEL description="Universal Agent can be used for Scheduling and File Transfer"
#
MAINTAINER "Colin Cocksedge <colin.cocksedge@stonebranch.com>"
#
# create license directory and add product license
RUN mkdir /licenses
ADD /licenses/Terms_and_Conditions.pdf /licenses
# Install ua
ARG ua_version
ADD /install-files/sb-${ua_version}-linux-3.10-x86_64.tar.Z /tmp
RUN zcat /tmp/sb-${ua_version}-linux-3.10-x86_64.tar.Z | tar xvf - && ./unvinst --network_provider oms --oms_servers 7878@localhost --python yes --
oms_port 7878 --security inherit --ubroker_start no && rm unvinst *.rpm *.tar upimerge.sh upimerge.log usrmode.inc install.log /tmp/sb-${ua_version}-linux-
3.10-x86_64.tar.Z
EXPOSE 7887 7878
#
# Set Permissions for Arbitrary ID Support
RUN chgrp -R 0 /etc/universal && chmod -R g=u /etc/universal && chgrp -R 0 /opt/universal && chmod -R g=u /opt/universal && chgrp -R 0 /var/opt
/universal && chmod -R g=u /var/opt/universal && chmod g=u /etc/passwd
# mkdir /var/opt/universal/uag/logs/ && ln -s /dev/null /var/opt/universal/uag/logs/agent.log &&
#
ENV PATH "$PATH:/opt/universal/bin"
# Set Default userid
USER 10010
# Add entrypoint script
COPY ./ua_entrypoint /
# Entry Point
ENTRYPOINT [ "./ua_entrypoint" ]
```

```
#!/bin/bash

echo "Entrypoint Version 1.6"

# Handle Docker Stop and Terminate Ubroker Cleanly
shutdown() {
    kill -TERM "$ubroker"
    wait "$ubroker"
    exit 0
}
# Recognize Terminaton
trap 'shutdown' SIGINT SIGTERM
# Support Arbitrary User IDs
if ! whoami &> /dev/null; then
  if [ -w /etc/passwd ]; then
    echo "${USER_NAME:-default}:x:$(id -u):0:${USER_NAME:-default} user:${HOME}:/sbin/nologin" >> /etc/passwd
  fi
fi

# Start the Agent
/opt/universal/ubroker/bin/ubroker -dest stderr &
ubroker=$!
wait "$ubroker"
```

Docker Environment Variables

When you create a Universal Agent container, you can configure the Universal Agent by specifying the following environment variables:

Environment Variable	Description	Example
OMSAUTOSTART	Specifies whether the Universal Broker starts an OMS server. Default = no	OMSAUTOSTART=yes
UAGAGENTCLUSTERS	List of Universal Controller Agent Clusters to join automatically. Default = 'Opswise - Default Linux/Unix Cluster, Opswise - Default Windows Cluster'	UAGAGENTCLUSTERS='Agent Cluster 1, Agent Cluster 2'
UAGAUTOSTART	Specifies whether the Universal Broker starts a UAG server. Default = yes	UAGAUTOSTART=no
UAGENABLESSL	Specifies whether the SSL/TLS protocol is used for network communication between UAG and OMS. Default = no	UAGENABLESSL=yes
UAGNETNAME	Sets the Agent ID to be used when the Universal Agent registers / connects to a Universal Controller Instance. Default = OPSAUTOCONF	UAGNETNAME=UADKR001
UAGOMSSERVERS	Specifies one or more OMS server addresses. Default = 7878@localhost	UAGOMSSERVERS=7878\@omsserver1, 7878\@omsserver2

UAGTRANSIENT	Specifies whether the Agent is Transient and will be deleted or decommissioned when the Agent shuts down or goes offline. Transient Agents are suspended from any Agent Clusters that they may belong to. Note If the Agent is referenced in any task definitions, the dynamic delete will fail. It should be understood that Transient Agents should never be specified directly in any task definition. They are designed to accept work via an Agent Cluster, so when configuring a Universal Agent to operate in a containerized environment, you should ensure that the Agent registers with one or more Agent Clusters via the agent_cluster uags.conf configuration option. Valid values are: • **yes** Agent is registered as a transient Agent. • **no** Agent is registered as a regular, persistent Agent. **Default is no.**	UAGTRANSIENT=yes
UEMAUTOSTART	Specifies whether the Universal Broker starts a UEM server. Default = yes	UEMAUTOSTART=no

Docker Container Ports

The following ports may need to be mapped when running containers from the Universal Agent image.

Port	Description
7887	Universal Broker listening port
7878	OMS Server listening port

Universal Agent Logs

The Universal Agent image configures the Universal Broker service to start in console mode, which writes all log data (unv.log and agent.log) to stdout. To view the log of a specific container, run the following command:

```
docker logs container-name
```

Running Universal Agent Docker Container Examples

To run a Universal Agent 6.6.0.1 container that connects to Universal Controller (via an OMS server) with SSL/TLS enabled and registers with an Agent ID of UA001, run the following command:

```
docker run --detach --env UAGNETNAME=UA001 --env UAGOMSSERVERS=7878@uchost --name ua-test stonebranch/universal-agent:6.6.0.1
```

Red Hat OpenShift Start-Up Guide

Introduction

This start-up guide describes a use case for how to securely transfer business data located on an on-premise Linux server to an application running on Open Shift, and vice versa, in real-time. As a result, the applications on Open Shift will always have the most up-to-date business data.

The solution also enables the delivery and reception of data from a Windows server, mainframe, or any cloud storage to an application running on Open Shift, and vice versa. For more information, refer to the solution paper Hybrid Cloud File Transfer Solution Paper [8].

Use Case Description

To improve scalability, reduce resource consumption, and shorten the time to develop, test, and roll-out new applications, many IT companies, are currently creating new applications or migrating their applications from their internal core IT environment into an Open Shift environment hosted on-premise and on public clouds.

These new or migrated applications are running in containers in an Open Shift POD. In many cases, they require business data from multiple sources, including on-premise business applications, mainframe, and various public cloud storage systems. Additionally, they both receive and provide data to connected systems, such as SAP business warehouse.

The following use case demonstrates how to securely transfer business data located on a LINUX server to a web application running on Open Shift, in real-time.

USE CASE: “news-flash application”

This sample use case demonstrates how all started instances of a web server for “breaking news” (one POD per web server) are updated in Open Shift with a new webpage. The new webpage HTML files and related pictures are sent from an on-premise server to all running PODs containing a web server instance started for the news-flash application in Open Shift. As a result, all users connected to the new-flash application will see the newly published webpage information.

Key Features

This start-up use case focuses on file transfer from an on-premise server to a group of PODs. The solution can also support many additional scenarios.

The following are its key features:

- Transfer files from any (virtual) Linux/Windows server to an application on Open Shift (and vice versa).
- Transfer files from the mainframe to an application on Open Shift (and vice versa).
- Transfer a file from any cloud storage platform to an application on Open Shift (and vice versa).
- Trigger either a time-based or an event-based file transfer (for example, from a web application using REST APIs).
- End-to-end monitoring of all file transfers (including monitoring of log files).
- Cloud (SaaS) or on-premise solution.
- Enhanced security – validated by regular penetration testing.
- High availability.

Solution Architecture

Universal Automation Center (UAC) is a web-based enterprise scheduler, available as SaaS in the cloud or on-premise.

UAC consists of:

Universal Controller	Web-based workflow, reporting, and orchestration engine.
Universal Agent	Workload execution component.
Universal Open Shift Agent	Workload execution component that runs in an Open Shift POD as Docker container.

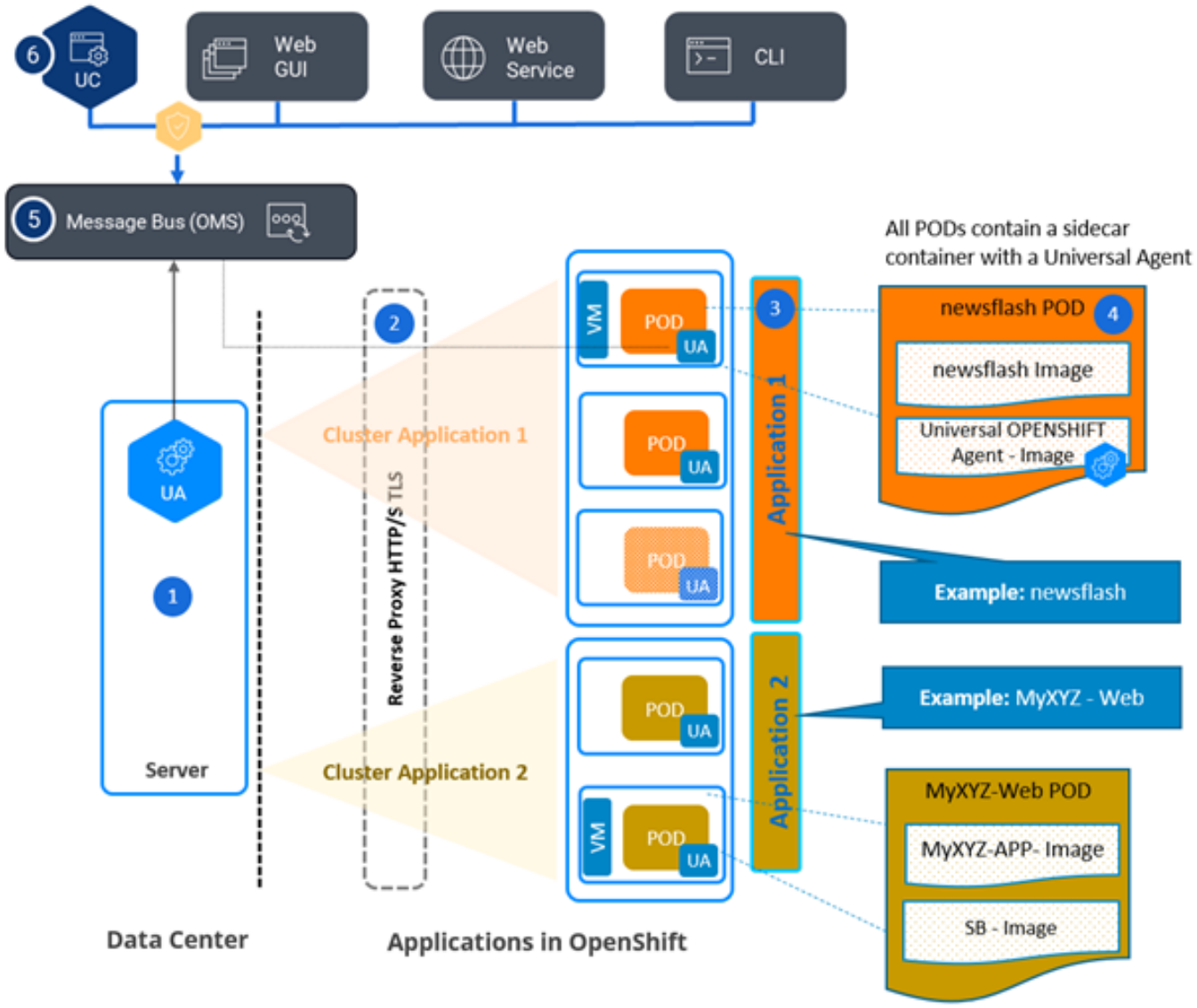
A Universal Agent specifically built for the Open Shift environment is called a Universal Open Shift Agent.

As soon as an agent is installed on a server, it automatically connects to the middleware message bus **OMS** of Universal Automation Center and is ready to execute remote commands/scripts and file transfers, regardless of whether the agent runs on a server, mainframe, or inside an Open Shift POD.

For applications that provide an API such as SAP, databases, or cloud storage services, no agent is required, as they are scheduled via their corresponding API.

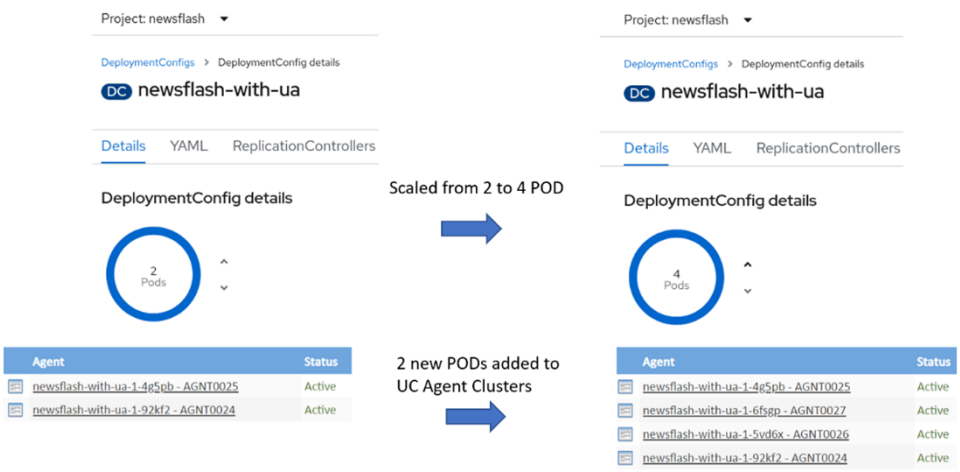
General Description of File Transfer Process

The architecture below outlines how data is transferred from an on-premise server to all instances of an application running on Open Shift. A transfer from the mainframe is performed in a similar way.



Provide business data from an on-premise server to an application running in a POD.

File Transfer Process Steps

Steps	Component	Description
Customer Data Centre		
Step 1	Linux Server	Universal Agent must be installed on the server that sends/receives files from an application distributed in an Open Shift POD. This agent can sen in the Data Center, and it can also connect to any public cloud storage for file transfer.
Open Shift PaaS		
Step 2	Reverse Proxy	Due to security regulations, all communication from and to Open Shift should be sent via reverse HTTPS proxy.
Step 3	Application Instance cluster	An application is deployed in a POD. If the application load **increases**, the Open Shift orchestration platform allows users to dynamically **scale up** the number of application instances I  ***Scaling Up from 2 to 4 Application Instances in Open Shift and Automatically in Universal Controller*** In this example, the number of PODs is increased in Open Shift from 2 to 4, and the Universal Agents installed in the PODs are added dynamicall cluster related to the application. If the load decreases, application instances, (that is, PODs) can be stopped in Open Shift.

Step 4	POD with Sidecar Container	All PODs contain a sidecar container with a Universal Open Shift Agent. Each sidecar container is based on the Red Hat UBI image with a Univers inside. The latest version of the image can be retrieved from the docker registry or the Red Hat catalog. Detailed documentation of the image can be found here [1]. When a POD (and, respectively, the sidecar container) is started, the Universal Open Shift Agent of the container automatically registers to a Univ dedicated to the application. Only a single outbound port is opened from the POD to the OMS (Controller message bus) “5”. No inbound port to Open Shift is required. For eacl Universal Agent cluster is created, containing the Universal Open Shift agents of all started instances of the application. The example in General Description of File Transfer Process, above, shows two applications: 1. newsflash 2. MyXYZ – APP Each instance of an application is represented by one POD. As soon as the Universal Open Shift Agent of the sidecar container is assigned to the related Universal Controller agent cluster, all related PODs any other Universal Agent installed on any server. In addition, the application running in the POD can be scheduled like any other application, ena automated business process. The Universal Agent cluster supports file transfers to just one agent (POD), or to all agents (that is, started PODs related to an application) in the U
Step 5	OMS	OMS (Controller message bus). Universal Agent in the POD connects on start-up automatically to the Controller message bus. The IP address of OMS is provided in the deploym the sidecar container.
Step 6	Universal Controller	Universal Controller is the web-based workflow, reporting, and orchestration engine. For each application, one Universal Controller Agent Cluster must be configured. When the Universal Agent in the POD is started, it registers automatically to the UC Agent Cluster, which is configured in the deployment configur container.

How to Get Started

This section outlines how this solution can be deployed in the form of a basic file transfer from a local server to a web application running in Open Shift.

USE CASE: “News-Flash Application”

This sample use case demonstrates how all started instances of a web server for “breaking news” (one POD per webserver) are updated in Open Shift with a new webpage. The new webpage HTML files and related pictures are sent from an on-premise server to all running PODs containing a web server's instance started for the news-flash application in Open Shift. As a result, all users connected to the new-flash application will see the newly published webpage information.

The setup consists of three steps:

1. Configure the file transfer workflow using the Universal Controller Web-GUI.
2. Add the Universal Sidecar container to the application deployment script.
3. Deploy the POD to Open Shift and scale up or down as required.

Prerequisites

To set up this sample use case, the following prerequisites are required:

Universal Controller 7.x or Above	This use case requires either an on-premise Universal Controller or Universal Controller Automation Center as SaaS in the cloud. A free trial version can be requested here [2]. An automation process can be set up using the Universal Controller's drag and drop workflow definition tool. Each workflow can define dependencies between numerous tasks, independent of the operating system on which they are executed. In the sample scenarios described here, only a single file transfer task is required. This task can be manually configured, or a pre-configured task can be uploaded from GitHub.

Universal Agent 7.x or Above	File transfers are performed between Universal Agents. Universal Agents can be deployed on any platform (Linux, Windows, z/OS, Open Shift, etc.). For this scenario, a Universal Agent must be installed on an on-premise Linux server. That agent can then transfer data to a Universal Agent installed as a sidecar container in an application running in Open Shift. The Universal Agent in Open Shift is installed automatically by Open Shift during the deployment of the POD. For the installation of a Universal Agent on a Linux server, please refer to the Universal Agent installation guide found here [4].
Open Shift 4.x	This scenario uses Open Shift 4 deployed via a Red Hat CodeReady Container (CRC) [3]. A CRC enables Open Shift to run on a local laptop or server. However, any Open Shift 4 deployment could be used.
Linux Server	This server is used to send and receive data from the Open Shift application. Any Linux Server can be used. A Universal Agent needs to be installed on this server in order to send and receive files from the Universal Agent installed on Open Shift. For information on the installation of a Universal Agent on a Linux server, please refer to the Universal Agent installation guide found here [4].

Configuration Steps

The installation consists of three steps:

1. Configure the file transfer workflow using the Universal Controller web GUI.
2. Add the Universal Sidecar container to the application deployment script.
3. Deploy the POD to Open Shift and scale up or down as required.

Set Up a File Transfer Task in Universal Controller

This scenario requires an update to the homepage of a very simple application (newsflash) consisting of NGNIX web servers. Therefore, the new homepage HTML files and related pictures should be sent from the on-premise Linux server to all running PODs containing a web server instance started for the news-flash application in Open Shift.

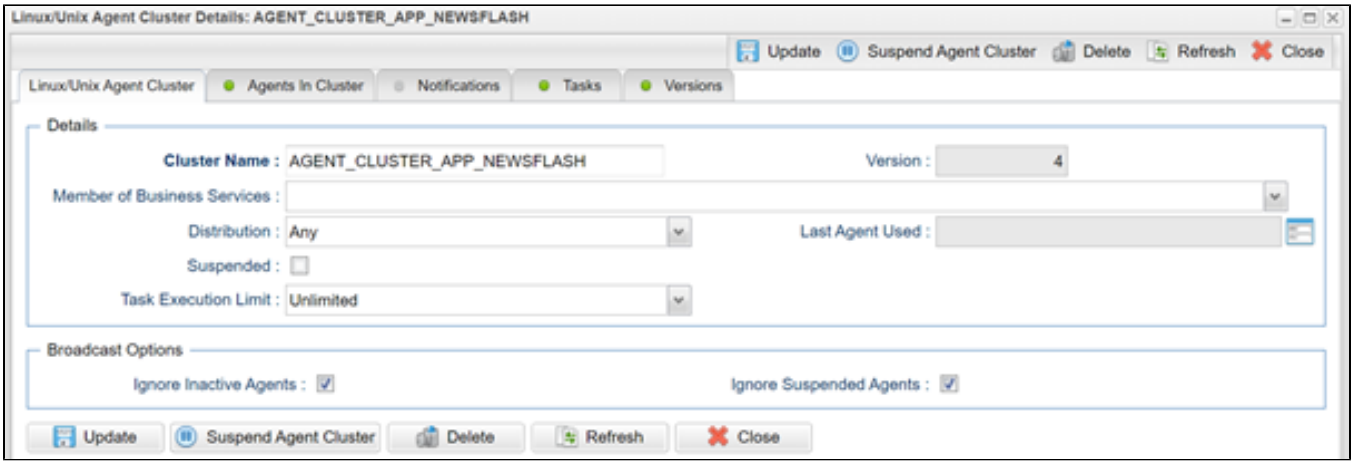
Setting up the file transfer task in Universal Controller requires two steps:

1. Define a new Agent Cluster for the Open Shift application “newsflash”.
2. Configure the file transfer task.

Define a New Agent Cluster for the “newsflash” Application

An agent cluster must be configured in Universal Controller for each application in Open Shift. When a POD is started for an application in Open Shift, the Universal Open Shift Agent, which is deployed in a sidecar container to the application POD, will automatically register to the Universal Controller agent cluster related to the application of the started POD.

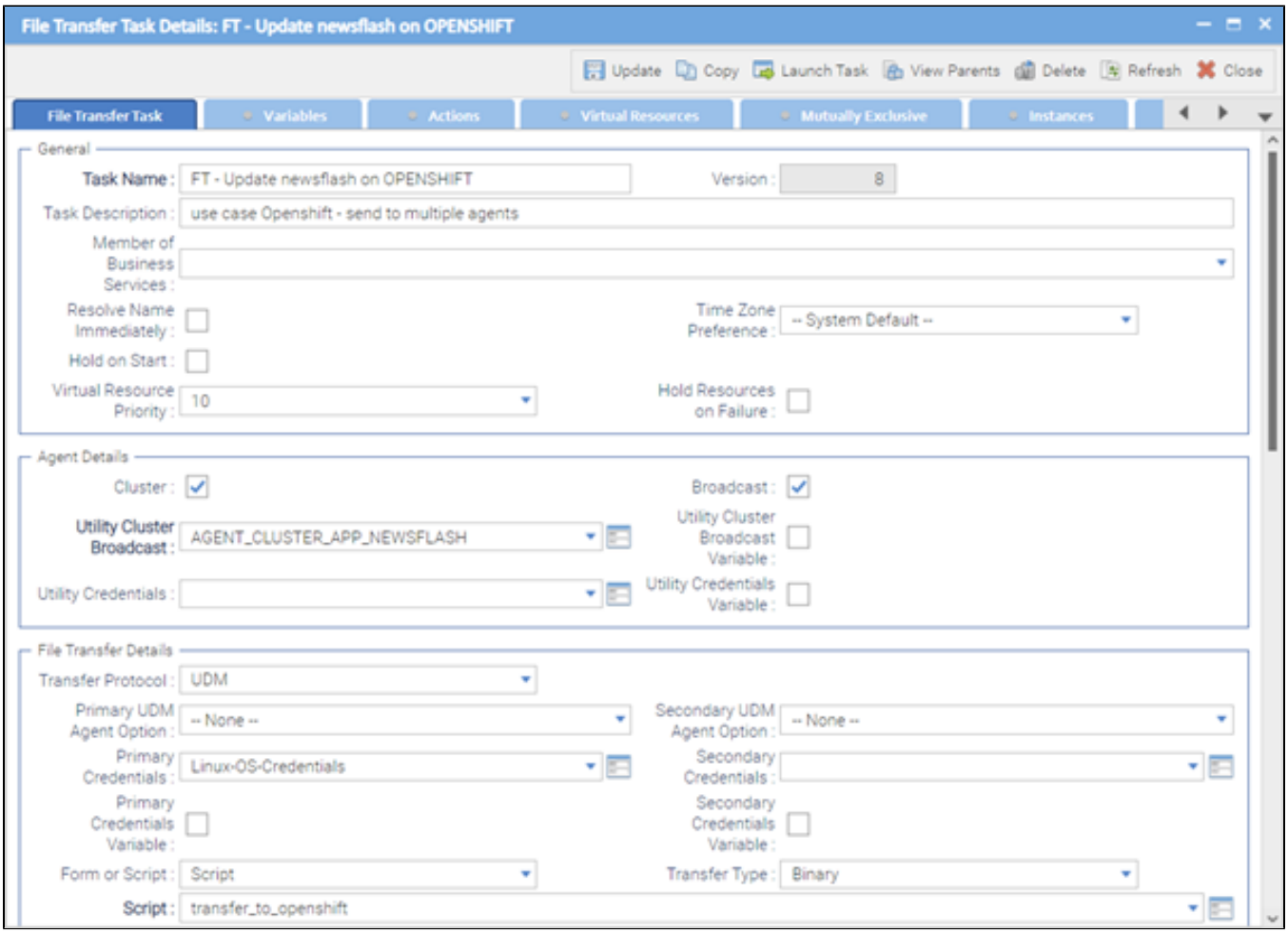
The agent cluster where all newsflash-related Open Shift agents will register is called *AGENT_CLUSTER_APP_NEWSFLASH.*



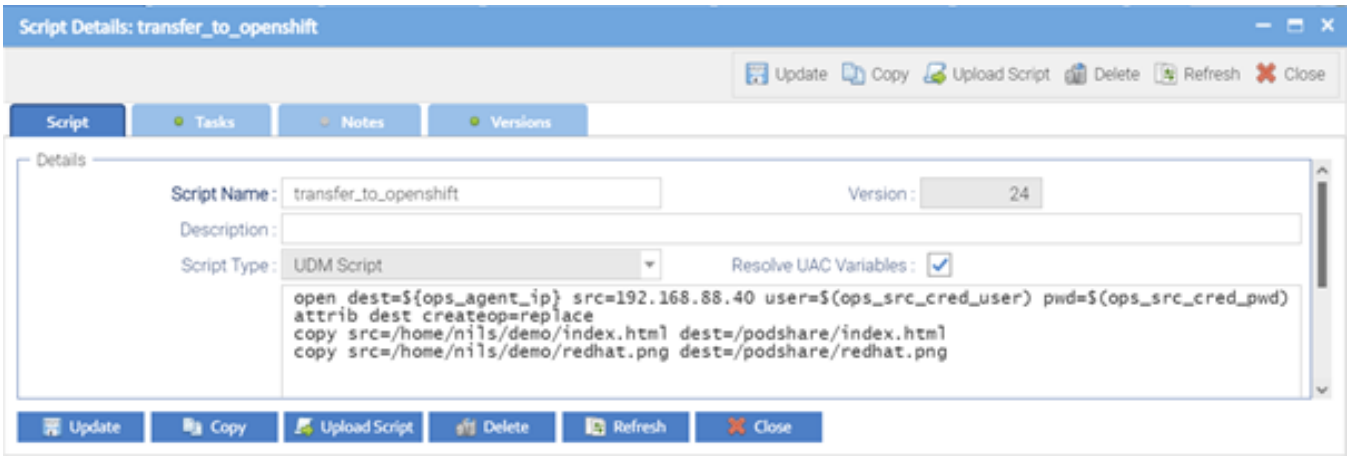
Universal Controller Agent Cluster

Configure the File Transfer Task

A new task must be configured for transferring the files from the on-premise Linux server to all agents assigned to the Universal Controller agent cluster.



Universal Controller File Transfer Task



Universal Controller File Transfer Task script

```
open dest=${ops_agent_ip} src=192.168.88.40 user=$(ops_src_cred_user) pwd=$(ops_src_cred_pwd)
attrib dest createop=replace
copy src=/home/nils/demo/index.html dest=/podshare/index.html
copy src=/home/nils/demo/redhat.png dest=/podshare/redhat.png
```

Description:

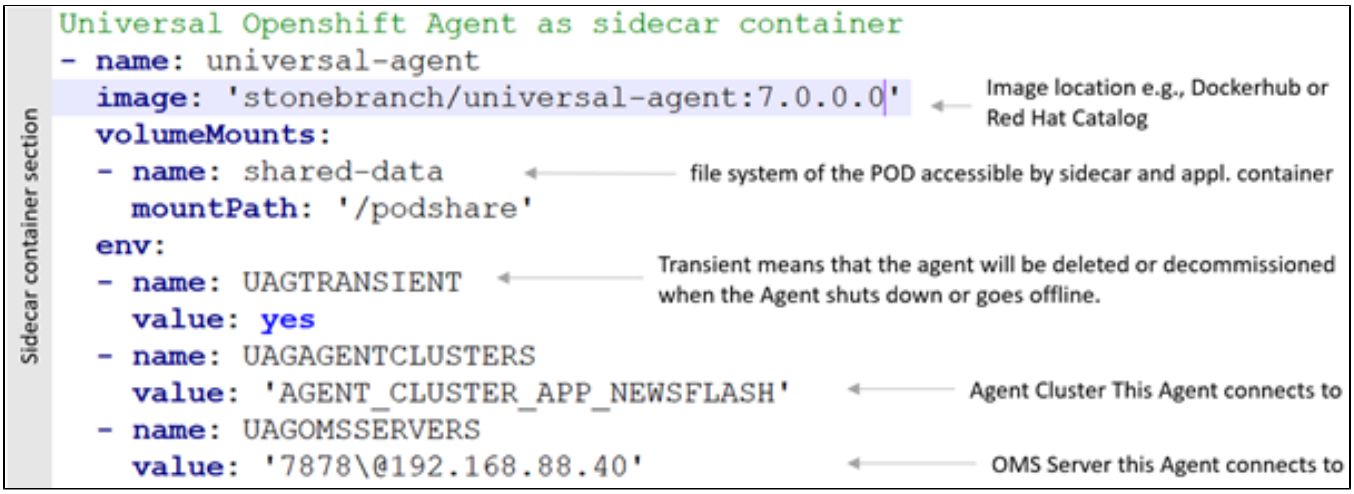
- Source Credentials: *Linux-OS-Credentials* (adjust according to your server credentials)
- Source Linux Server: *168.88.40* (adjust according to your server)
- Source Folder Linux Server: */home/nils/demo/out* (adjust according to your server)
- Files to transfer: *html, RedHat.png*
- Destination Agent Cluster: *AGENT_CLUSTER_APP_NEWSFLASH*
- Destination Folder in the POD: */podshare/ (this is the mounted POD directory)*

The export files of the file transfer task can be found here file transfer task.

Add the Universal Sidecar Container to the Application Deployment Script

Add the Universal Open Shift Agent as a Sidecar Container

To add the Universal Open Shift Agent as a sidecar container to your application, you must add the following section to your application deployment YAML file:



Universal Open Shift Agent deployment YAML

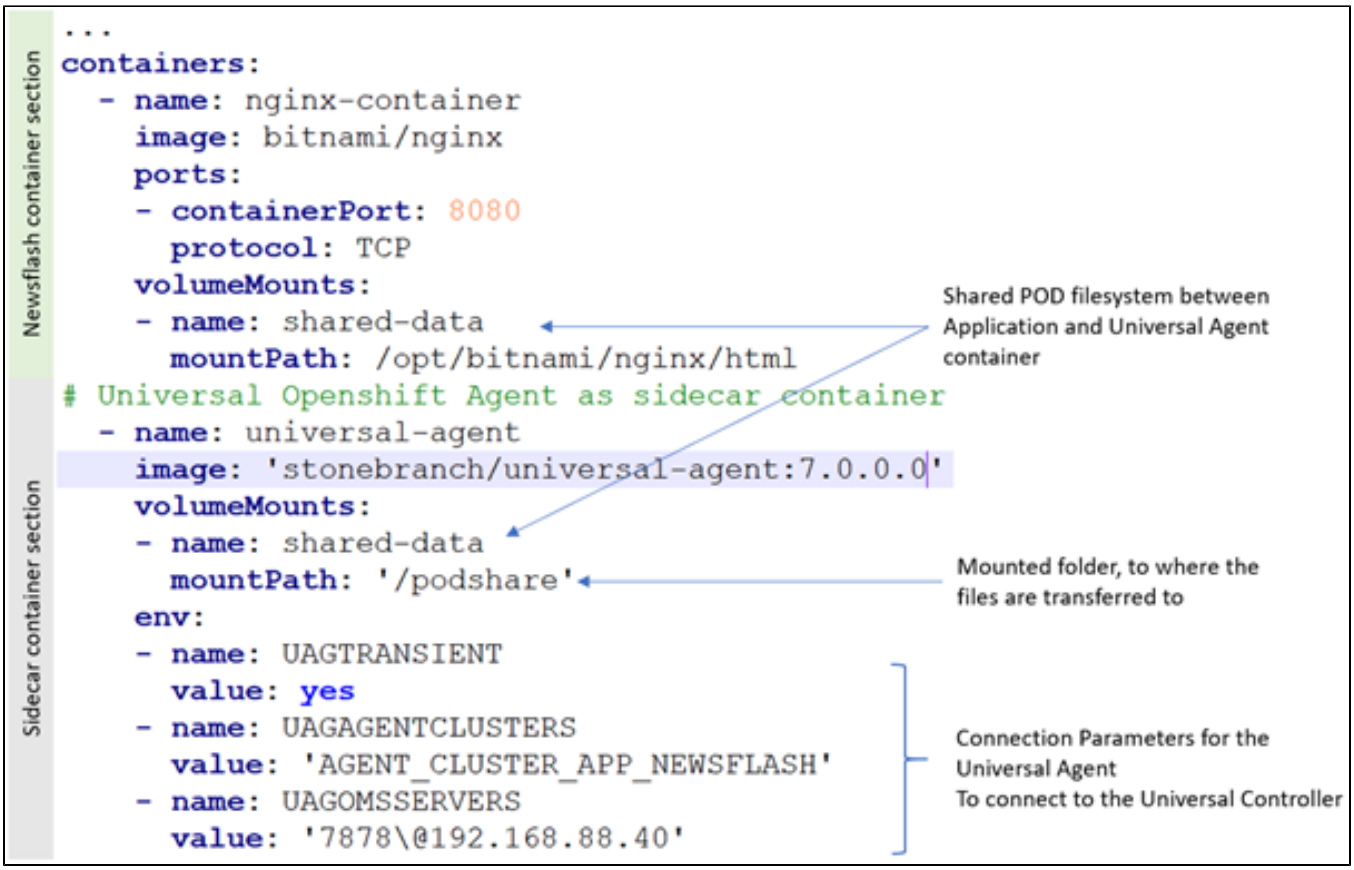
Configure the following three parameters in the deployment file:

Parameter	Description
UAGAENGTCLUSTERS	Name of the Open Shift Application (must be the same name as the Universal Controller Agent cluster configured in 2.3.1).
UAGOMSSERVERS	IP and PORT of the Universal Controller Message Middleware OMS
UAGTRANSIENT	“yes”: Transient means that the agent will be deleted or decommissioned, when it shuts down or goes offline.

*Note: A 30-day demo license key can be obtained from Stonebranch [5].

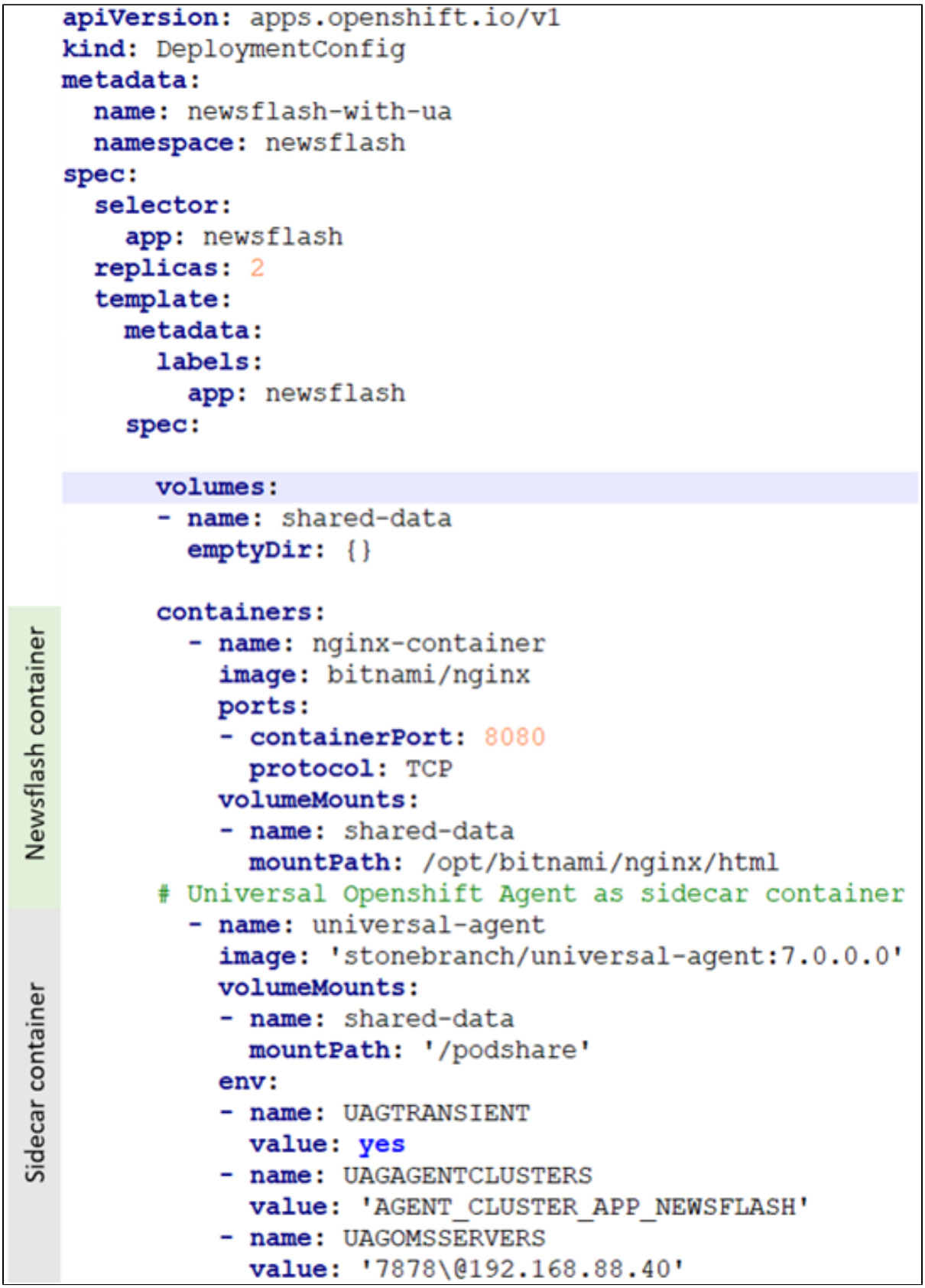
Configure a Shared Folder for Application and Sidecar Containers

To make the files received by the Universal Open Shift Agent available to the application in the POD, you must create a shared folder between the application and the sidecar container with the Universal Open Shift Agent.



Open Shift POD shared folder

The following shows the complete deployment script, consisting of the Open Shift web application based on an NGINX web server and the Universal Agent as a sidecar container.



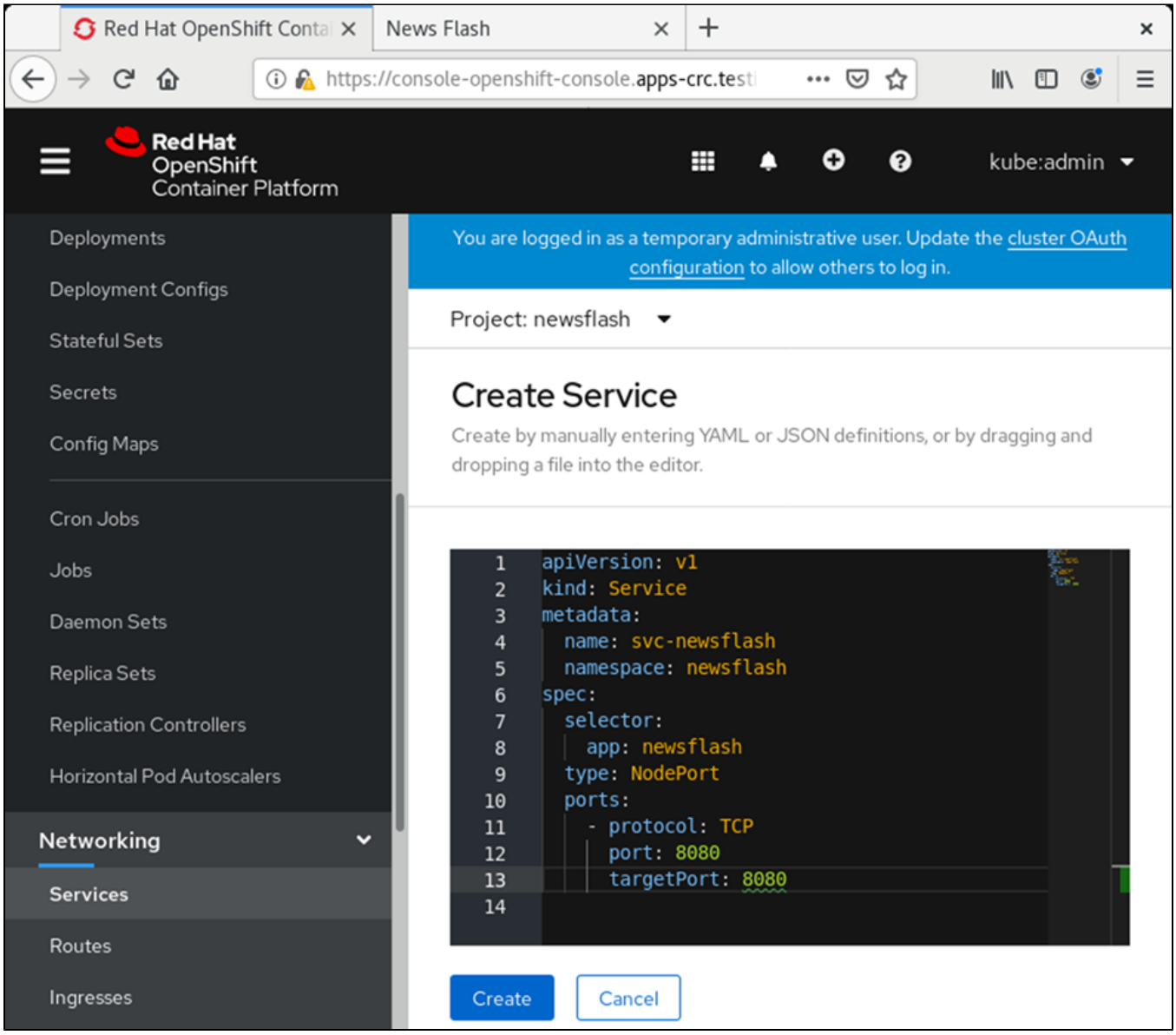

```yaml
apiVersion: apps.openshift.io/v1
kind: DeploymentConfig
metadata:
  name: newsflash-with-ua
  namespace: newsflash
spec:
  selector:
    app: newsflash
  replicas: 2
  template:
    metadata:
      labels:
        app: newsflash
    spec:

      volumes:
      - name: shared-data
        emptyDir: {}

      containers:
        - name: nginx-container
          image: bitnami/nginx
          ports:
          - containerPort: 8080
            protocol: TCP
          volumeMounts:
          - name: shared-data
            mountPath: /opt/bitnami/nginx/html
      # Universal Openshift Agent as sidecar container
        - name: universal-agent
          image: 'stonebranch/universal-agent:7.0.0.0'
          volumeMounts:
          - name: shared-data
            mountPath: '/podshare'
          env:
          - name: UAGTRANSIENT
            value: yes
          - name: UAGAGENTCLUSTERS
            value: 'AGENT_CLUSTER_APP_NEWSFLASH'
          - name: UAGOMSSERVERS
            value: '7878\@192.168.88.40'
```

Deployment YAML

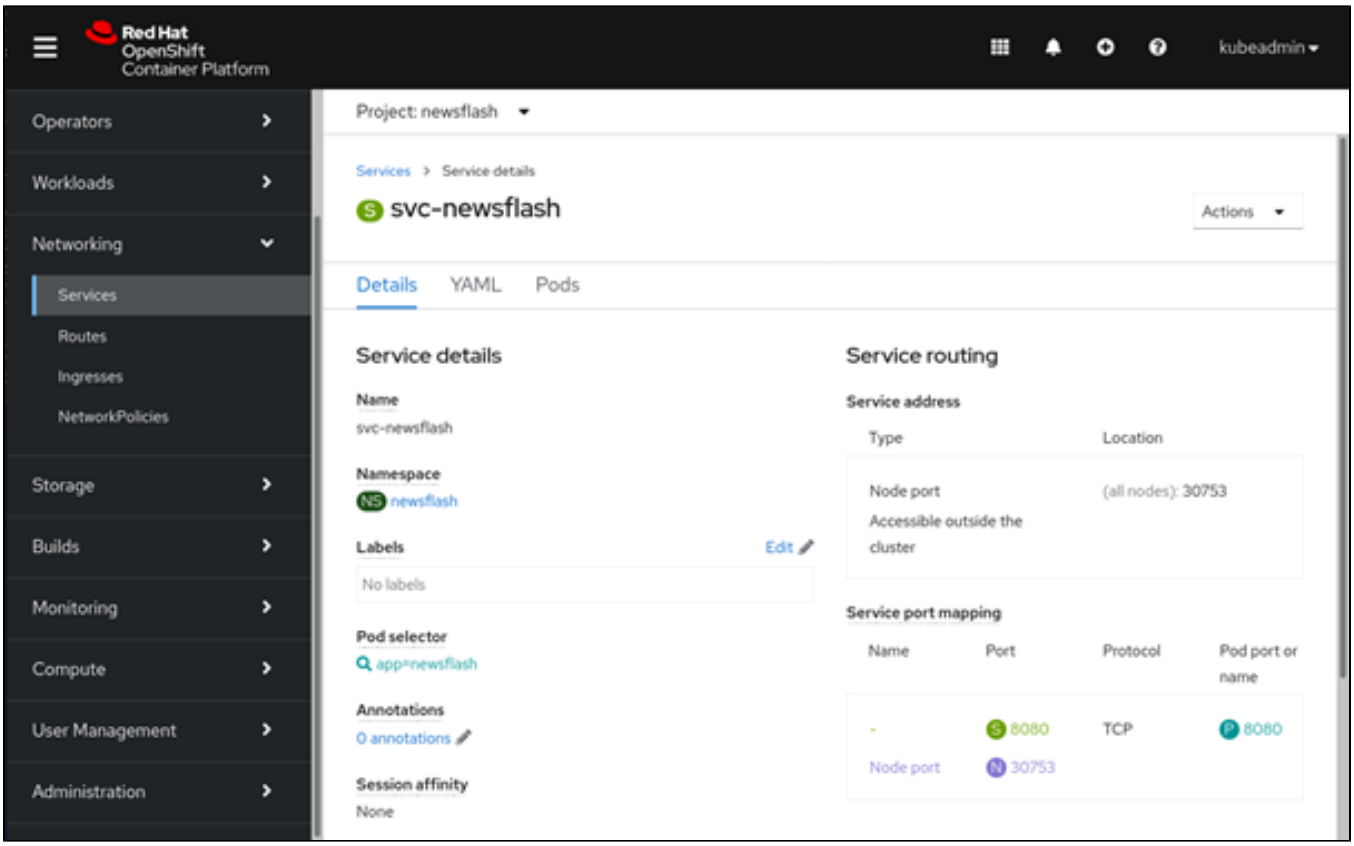
The file can be downloaded from GITHUB [6].

Configure Service to Access the Newsflash Application

To make the newsflash application available outside Open Shift, you must create a Service with a NodePort.



Open Shift Service YAML



Open Shift Node port

The Newsflash application will then be accessible via the following IP:

http://192.168.130.11:30753/

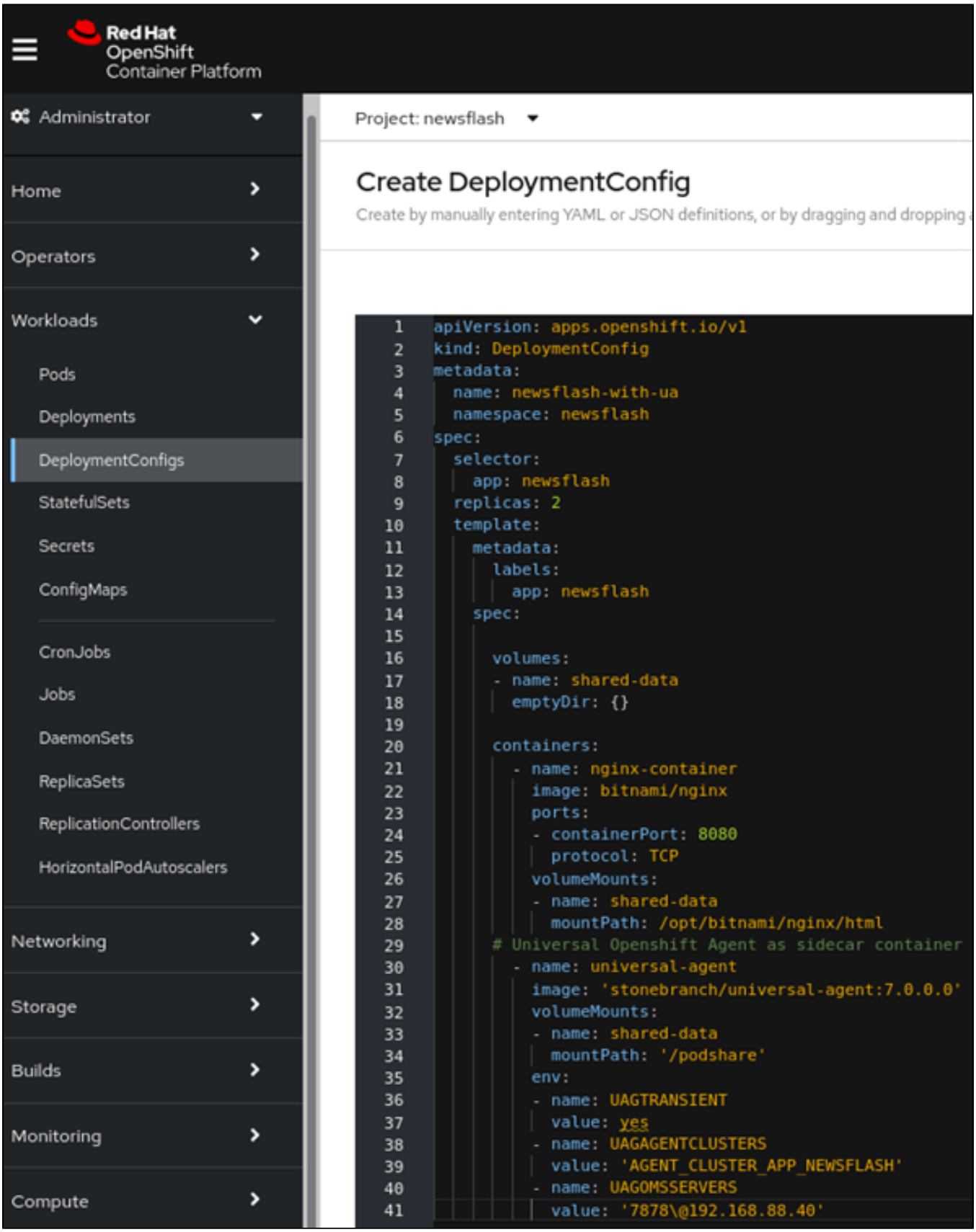
Note:

The host IP can be retrieved on a CRC-based Open Shift install via the command: *crc ip*.

In the described example, it is: *192.168.130.11*

Deploy the Open Shift Application

To deploy the application on Open Shift, you only need to deploy the deployment file in the ‘Deployment Configs’ screen (see below).

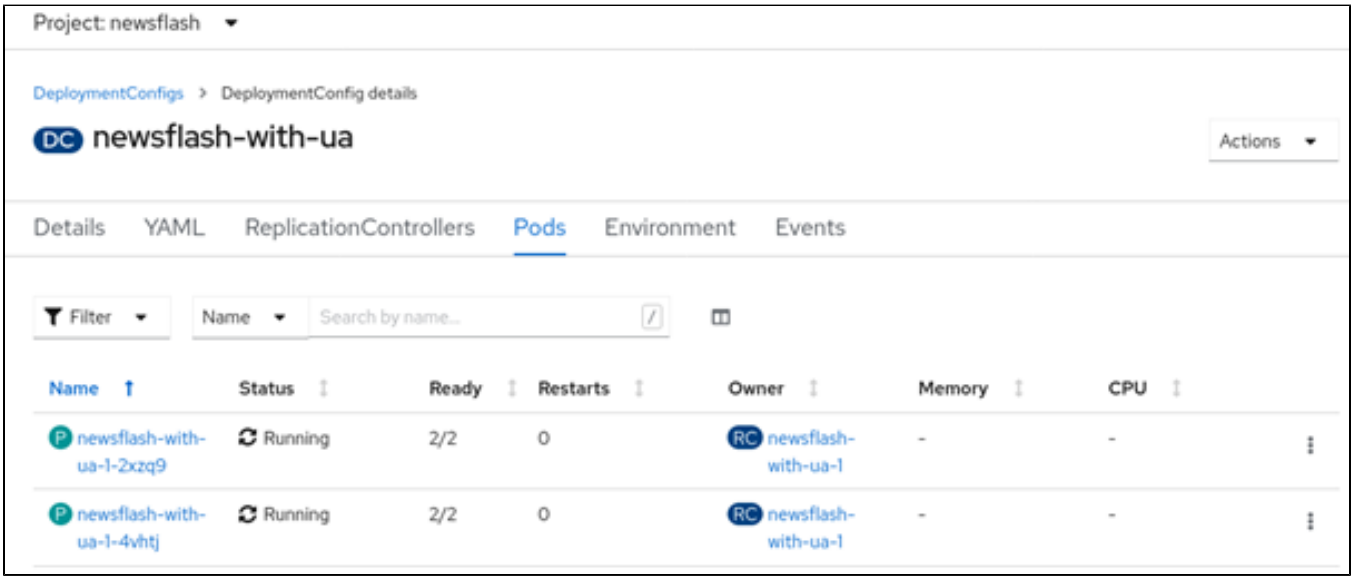


Open Shift Deployment Config YAML

In the deployment config file, we defined three replicas. This means that once we press the ‘Create Deployment Config’ button, three instances of our application will be started (= 3 PODs).

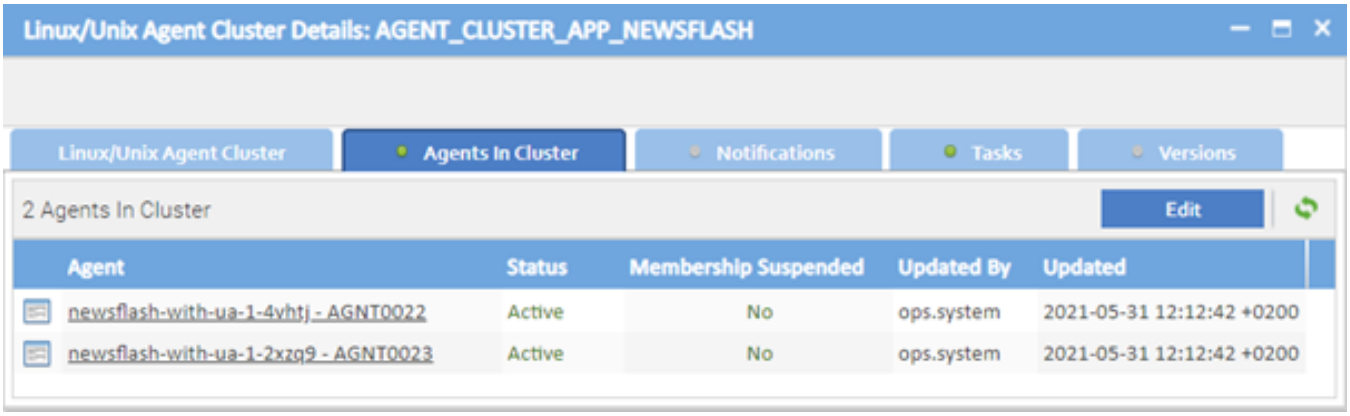
These three Universal Open Shift Agents will connect to the Universal Controller agent cluster, *“AGENT_CLUSTER_APP_NEWSFLASH,”* which was configured in the deployment script as a parameter and during the set-up in Universal Controller.

In the following screenshot, you can see the three started PODs in Open Shift:



Open Shift

…as well as the related Universal Open Shift Agents, which have registered automatically to the Universal Controller agent cluster:



Universal Controller Agent Cluster with Open Shift Agents

Auto Scaling of POD

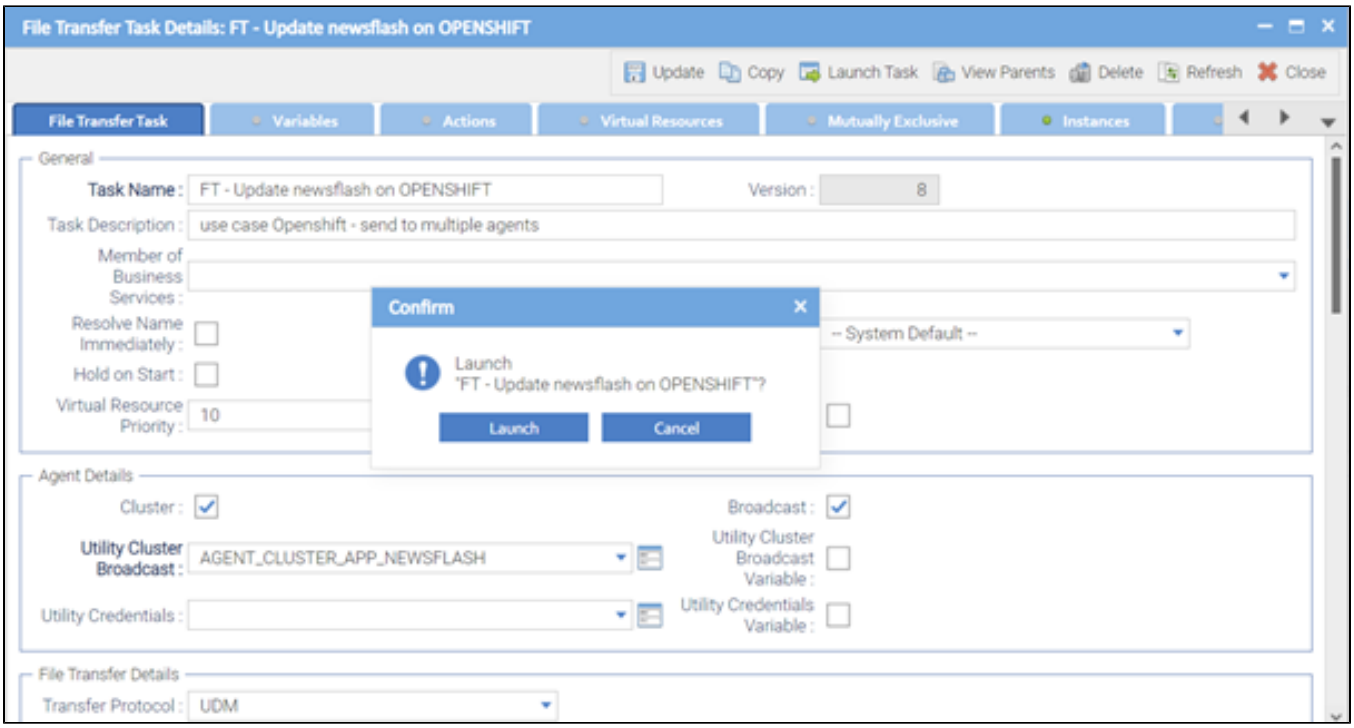
If the number of PODs in Open Shift is scaled up, more agents will automatically register for each new POD. Conversely, if the number of PODs is scaled down, the excess agents will automatically be removed from the Universal Controller agent cluster.

Running the File Transfer

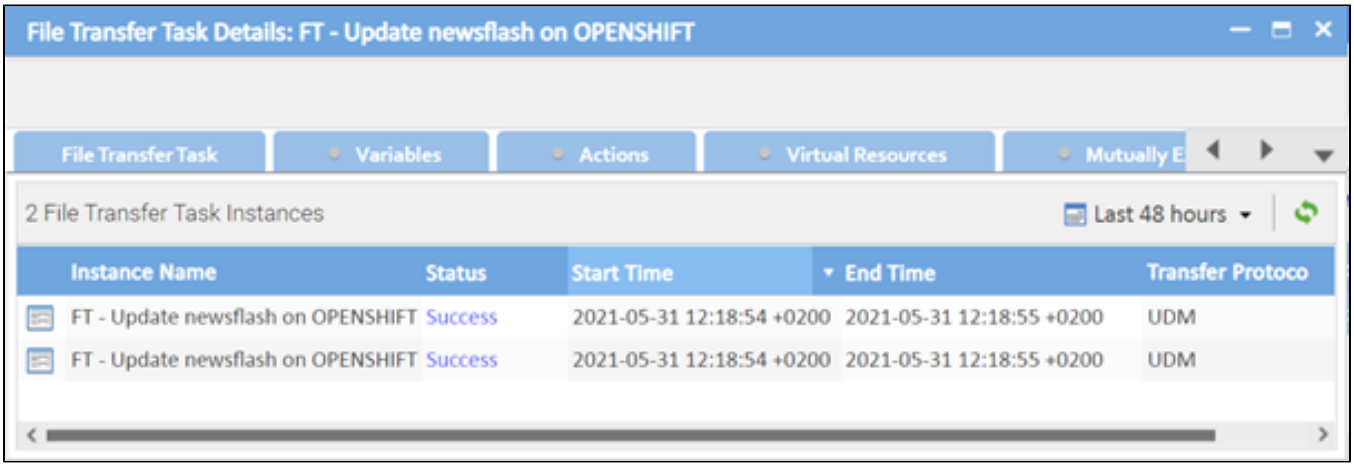
When the Universal Open Shift Agents have registered in the Universal Controller agent cluster, they are ready to send and receive files.

The file transfer task in Universal Controller is then launched, initiating the transfer of homepage HTML files, including related pictures from the on-premise Linux server to all started PODs of the newsflash application.

In this example, we triggered the file transfer manually.



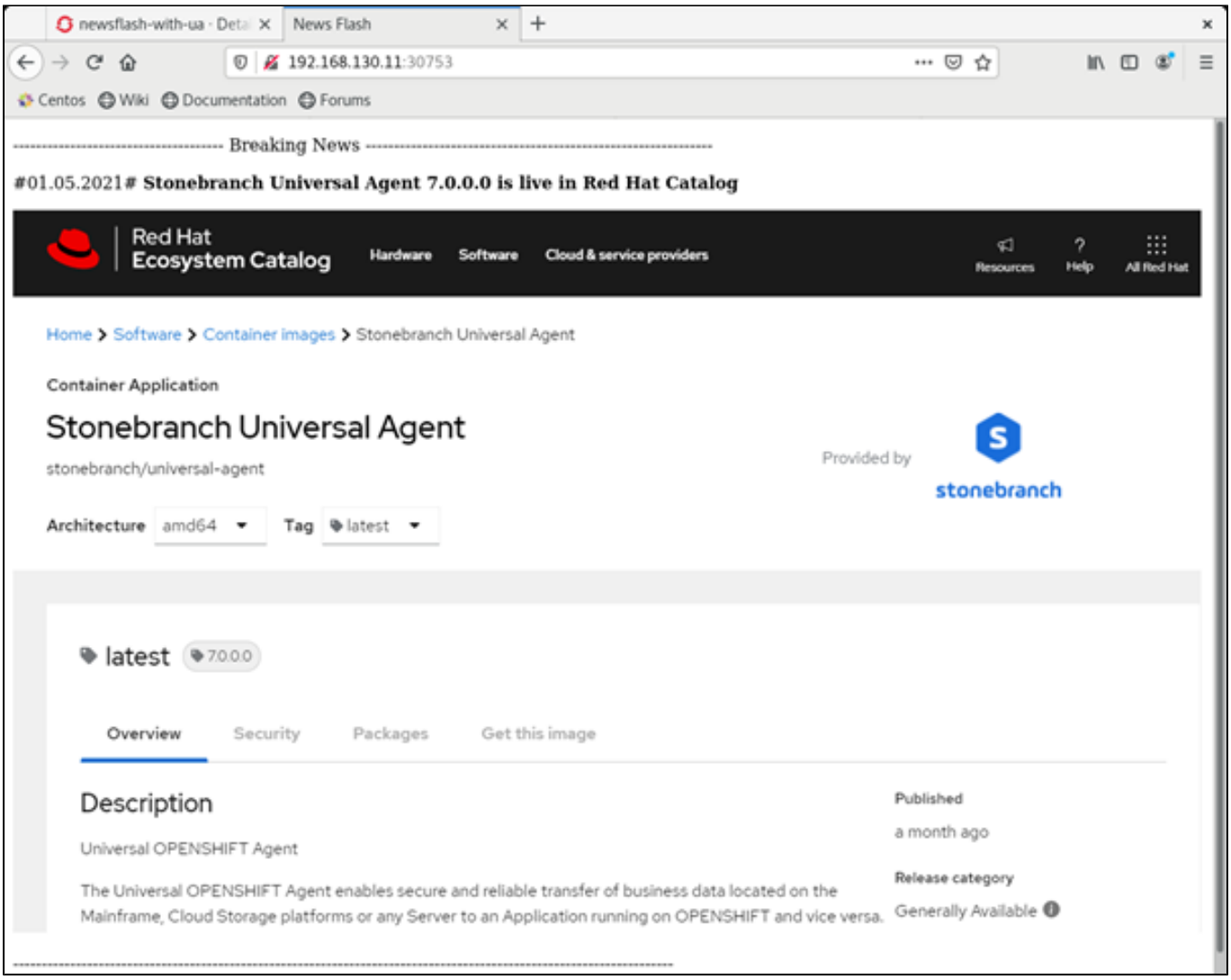
Universal Controller File Transfer Task



File Transfer Task Instances

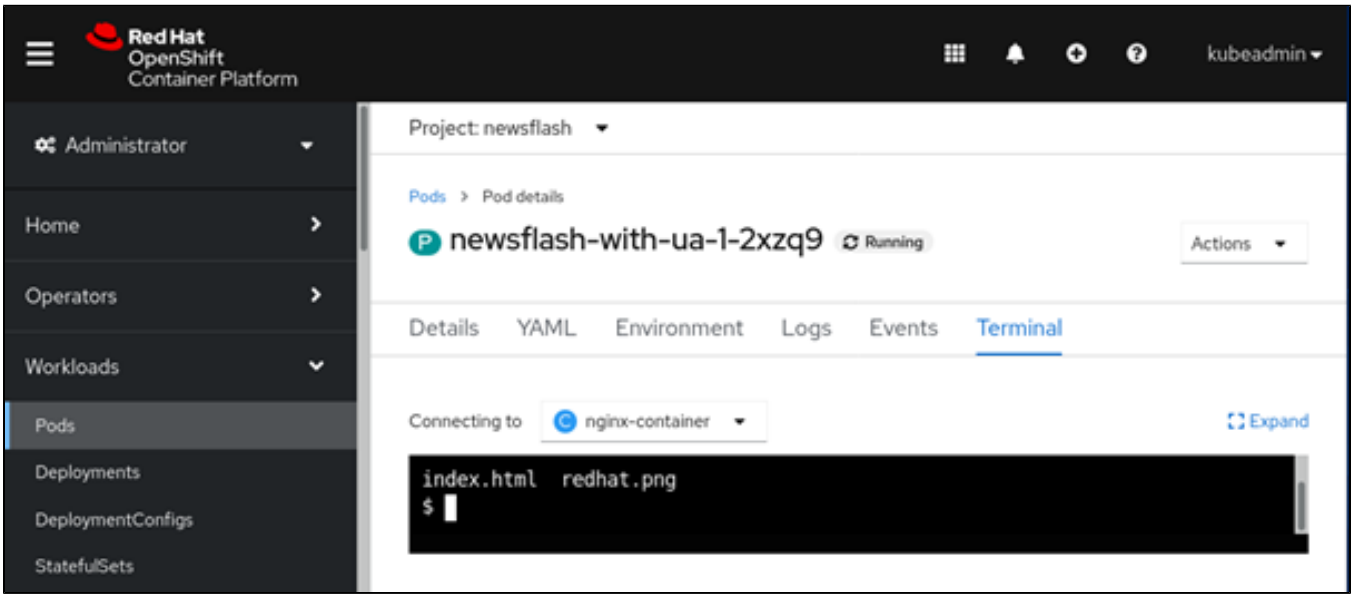
In the screenshot above, you can see that two file transfers were launched. This is because the Universal Controller agent cluster contained two agents, one for each started POD.

As a result, the files are transferred from the Linux server to all assigned PODs and the homepage is updated with the new data:



Newsflash Webpage

The new data can also be viewed in all PODs directly:



Open Shift POD Terminal

Options to Trigger the File Transfer Scenario

In the example above, we triggered the file transfer manually. However, Universal Controller can trigger the file transfers in numerous ways to ensure that the business data that the application requires is always consistent and up to date.

Additional triggers include (but are not limited to):

- File arrival
- Time-based (with support for an internal and external calendar like an SAP calendar)
- Email arrival
- Web services
- Event in a message queue (for example, MQ, JMS)
- SAP Event
- The status of another task/workflow (for example, the start of a new file transfer or if the transfer from last night was successful)

Integration of File Transfer into Microservices Architectures

Any file transfer can be triggered by calling the REST API of Universal Controller.

Essentially, a file transfer workflow can be started from any application, independent of the platform it runs on (for example: virtual server, mainframe, or POD). This single API enables a loosely coupled integration with a microservices architecture.

Security and Auditability

The file transfer protocol outlined above is based on Stonebranch UDM protocol, which encrypts all data and communication channels using TLS1.2 (for example, AES 256 / SHA 384).

Universal Controller's web GUI real-time reporting functionality provides full auditability through detailed information of all data transfers, including log files.

Document References

Ref#	Description
[1] https://docs.stonebranch.com/confluence/display/UA70/Docker+Containers	Open Shift Agent documentation
[2] https://www.stonebranch.com/products/universal-automation-center/	Link to Free Trial for Universal Automation Center
[3] https://developers.redhat.com/products/codeready-containers/overview	RED HAT webpage to retrieve an Open Shift Code Ready Container (CRC)
[4] https://docs.stonebranch.com/confluence/display/UA70/Universal+Agent+for+UNIX+Installation	Universal Agent Installation
[5] https://www.stonebranch.com/contact/	Contact address to get a 30days license key for Universal Data Mover
[6] https://github.com/stonebranch-marketplace/Universal-Open Shift-Agent/tree/master/export	Universal Controller File Transfer task export (xml) The XML files are bundled in a tar archive. 1. Extract the tar file to a directory accessible by your Universal Controller (tar -xvf startup_guide_export.tar). 2. Import the extracted xml files using the Universal Controller list import feature. 3. Adjust the imported file transfer task to your environment.
[7] https://github.com/stonebranch-marketplace/Universal-Open Shift-Agent/tree/master/src	• POD deployment YAML file • Service deployment YAML file • Webpage index.html and redhat.jpg
[8] https://www.stonebranch.com/solutions/hybrid-cloud-file-transfers/	Hybrid Cloud File Transfer Solution Paper

Summary and Benefits

The Universal Automation Center, with the introduction of the newly developed Universal Open Shift Agent, enables the secure and reliable transfer of business data located on the mainframe, on cloud storage platforms, or any server, to an application running on Open Shift (and vice versa).

As this example has shown, only three steps are required to configure a secure file transfer to or from any of your Open Shift applications.

Red Hat OpenShift Operator Start-Up Guide

Introduction

This start-up guide describes how to deploy a Stonebranch Universal Agent to an Openshift namespace using the Universal Agent Operator from the Red Hat Marketplace.

When Universal Agent has been deployed, it can be used to send and receive files from any external server or cloud storage and save it to a shared persistence storage system on the related OpenShift namespace.

Any application in OpenShift that has access to the same shared persistence storage can make use of the received files or place data to the shared persistence storage for further file transfer using Universal Agent.

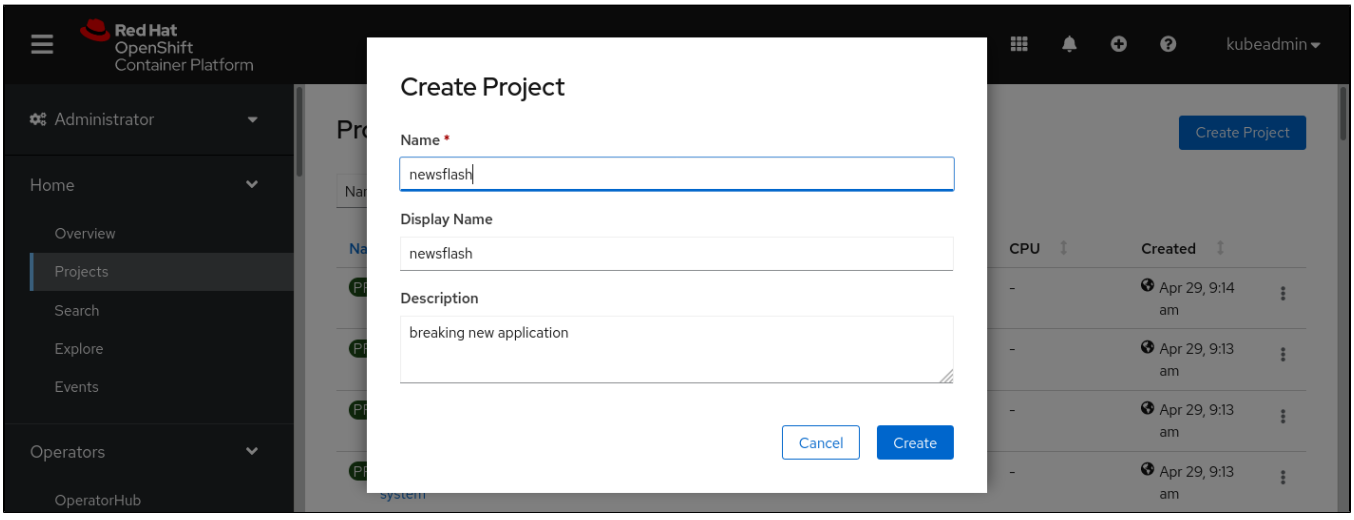
At the end of this tutorial, an example scenario is provided showing how to transfer data from a Linux Server to a shared persistence storage system.

Step 1 Prerequisites

Universal Agent is deployed as POD in a namespace of a Project.

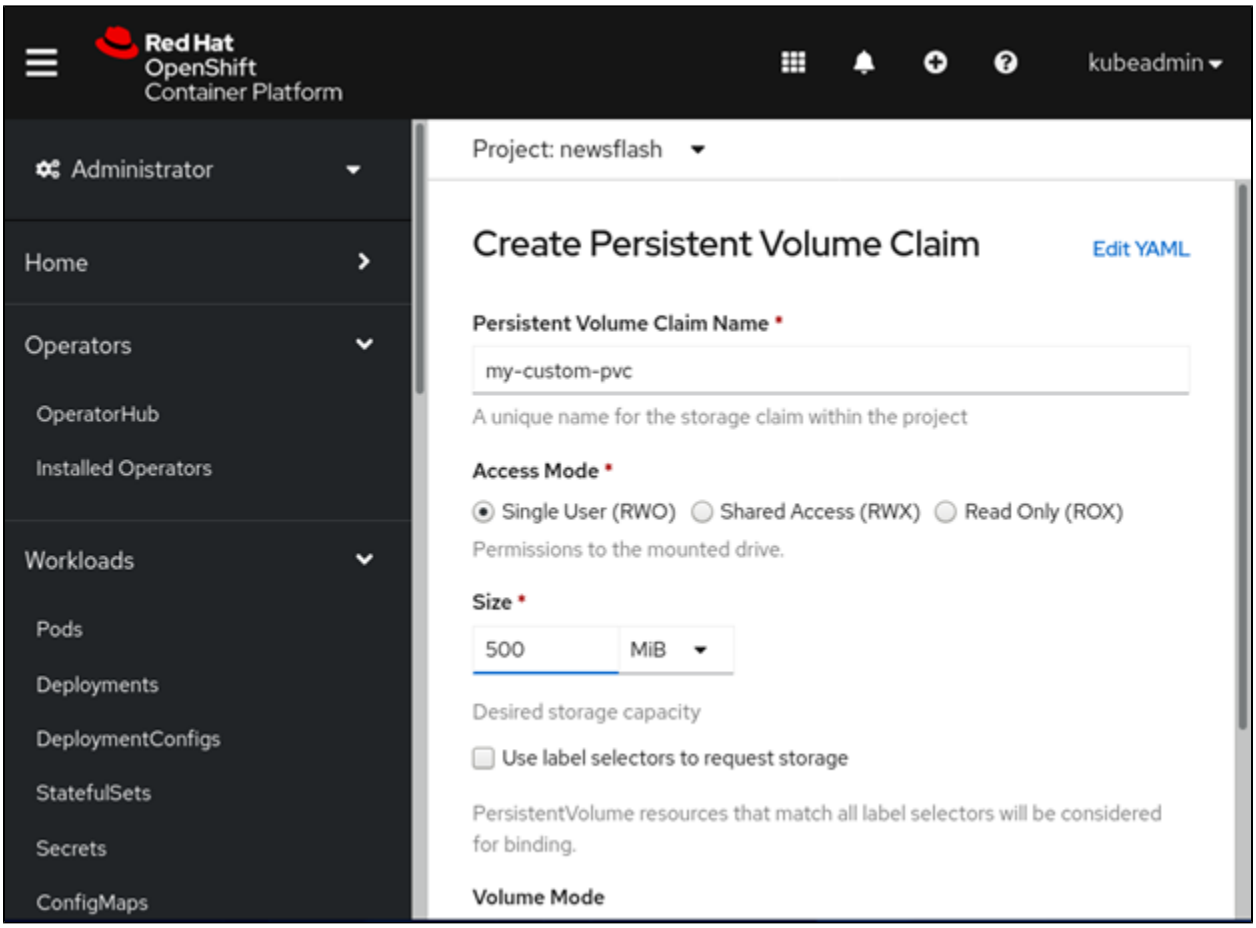
1 Create a Project

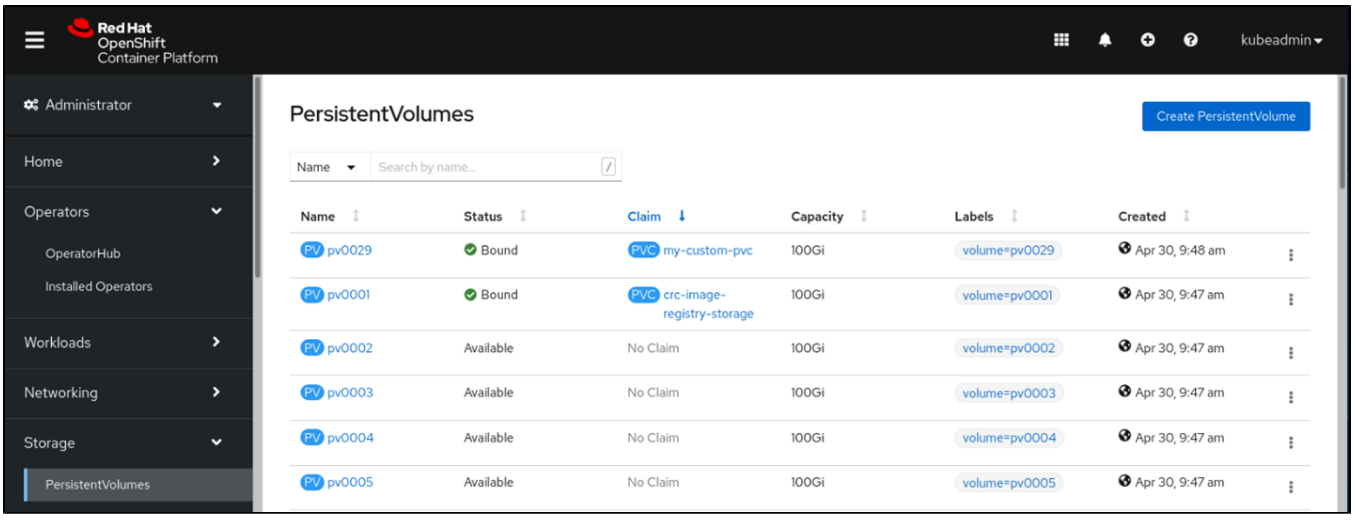
Create a project with namespace **newsflash**. If you already have a project where you would like to deploy Universal Agent, you can skip this step.



2 Create a Persistent Volume Claim

Create Persistent Volume Claim **my-customer-pvc** with size 500MiB. If you already have a Persistent Volume Claim, you can skip this step.





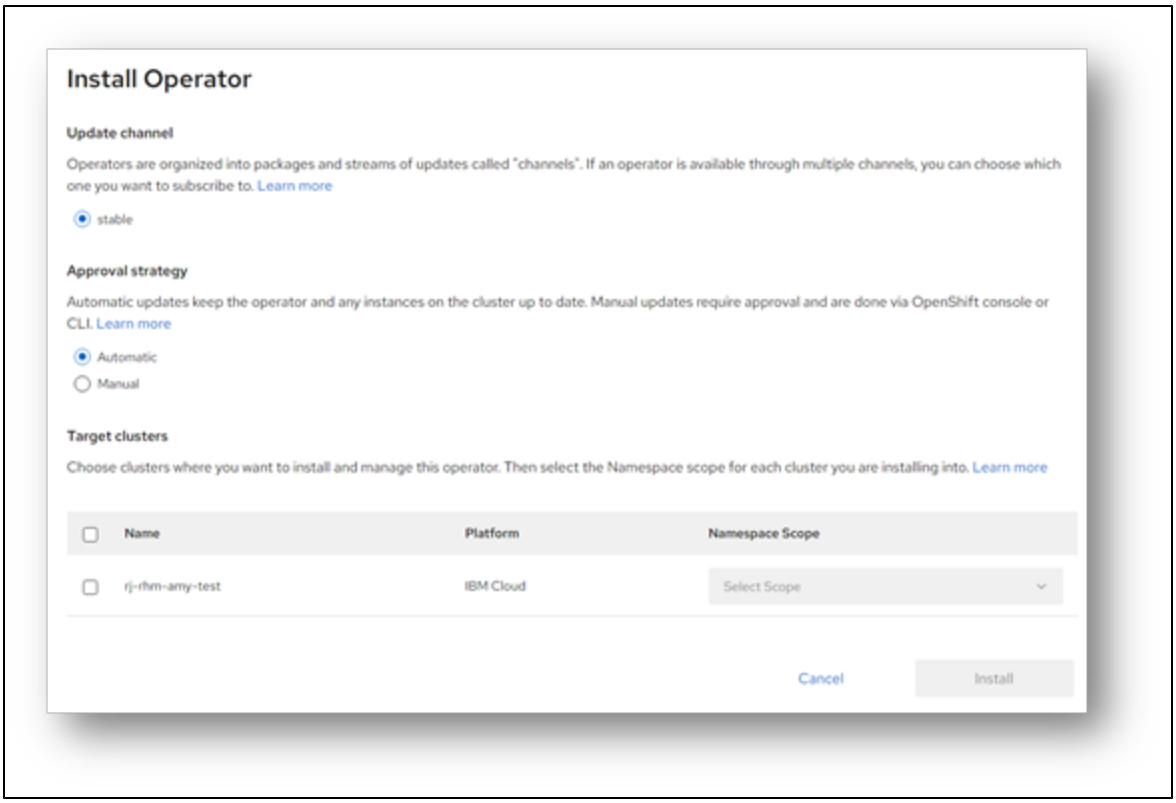
Step 2 Operator Installation

The Operator can be installed either:

- Directly from the Marketplace.
- From within the OpenShift Web Console.

Operator Installation from Red Hat Marketplace

1. For information on registering your cluster and creating a namespace, see Red Hat Marketplace Docs. This must be done prior to operator install.
2. On the main menu, click **Workspace**>**My Software**>**product**>**Install Operator**.
3. On the *Update Channel section*, select an option.
4. On the *Approval Strategy* section, select either *Automatic or Manual*. The approval strategy corresponds to how you want to process operator upgrades.
5. On the *Target Cluster* section:
    - Click the checkbox next to the clusters where you want to install the Operator.
    - For each cluster you selected, under **Namespace Scope**, on the **Select Scope** list, select an option.
6. Click **Install**. It may take several minutes for installation to complete.
7. Once installation is complete, the status will change from **installing** to **Up to date**.
8. For further information, see the Red Hat Marketplace Operator documentation.

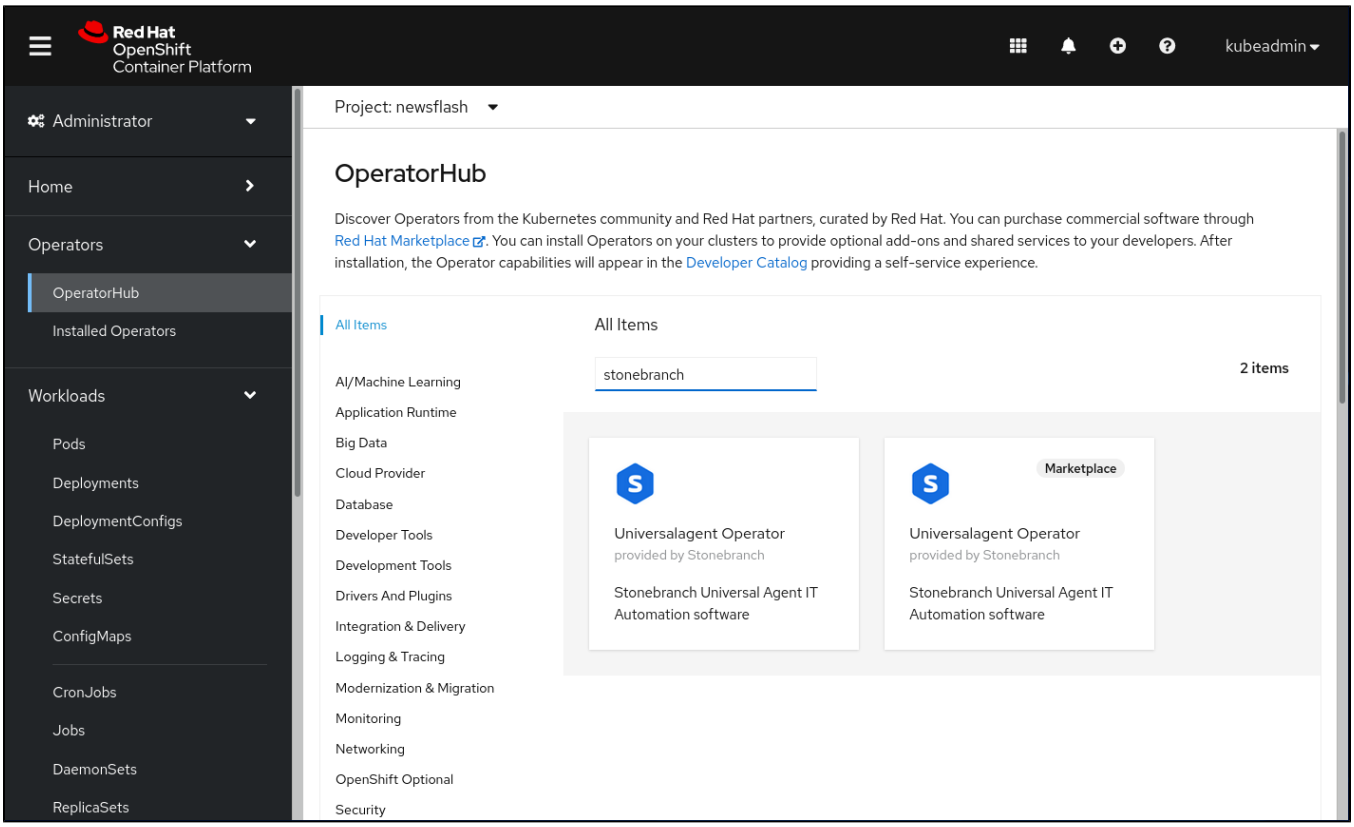


Operator Installation from within the OpenShift Web Console

This section describes how to install the Universal Agent Operator on an OpenShift Cluster within the OpenShift Web Console

1 Deploy the Operator from the Red Hat Marketplace

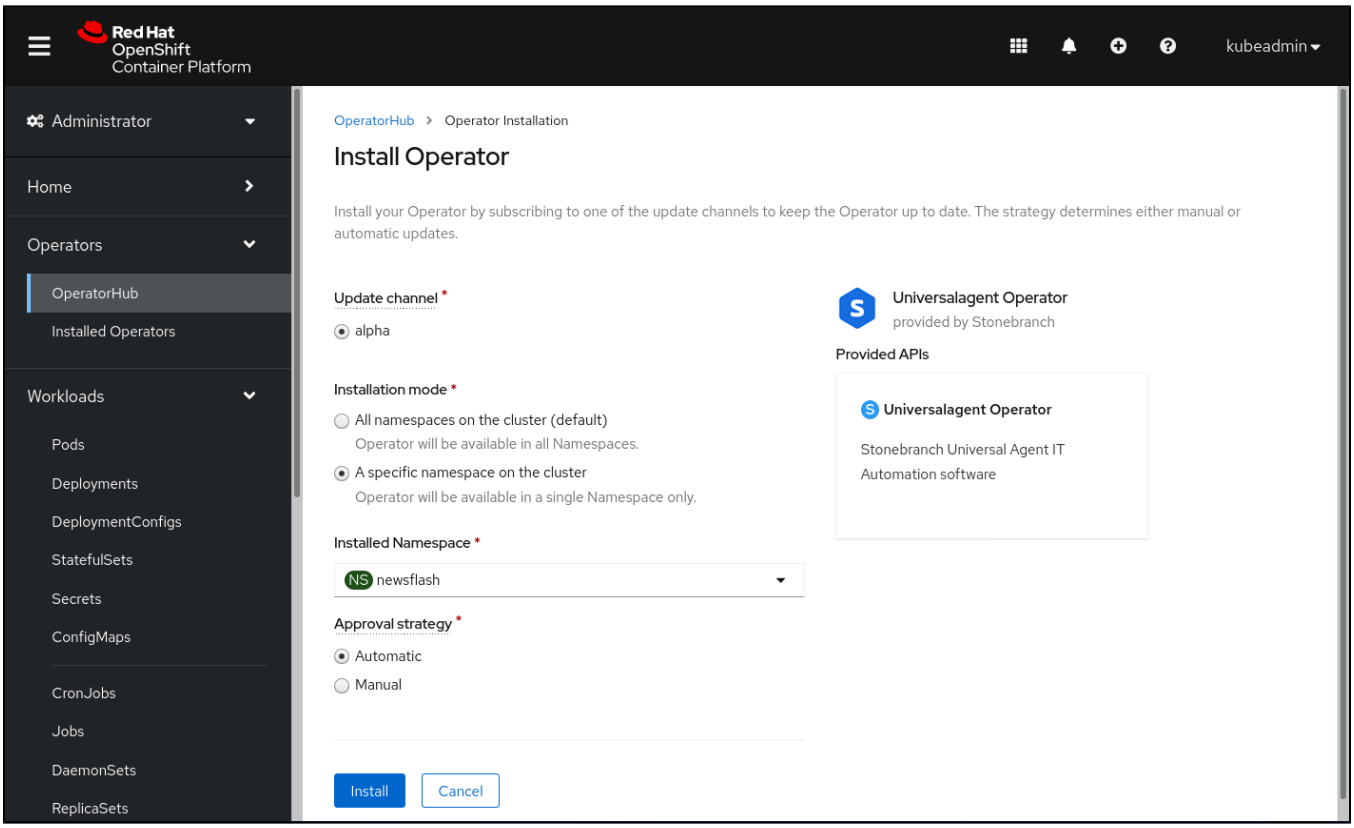
In the OpenShift Web Console, select: Operators -> OperatorHub and search for **stonebranch**, Provider Type: **Marketplace.**



2 Install the Operator

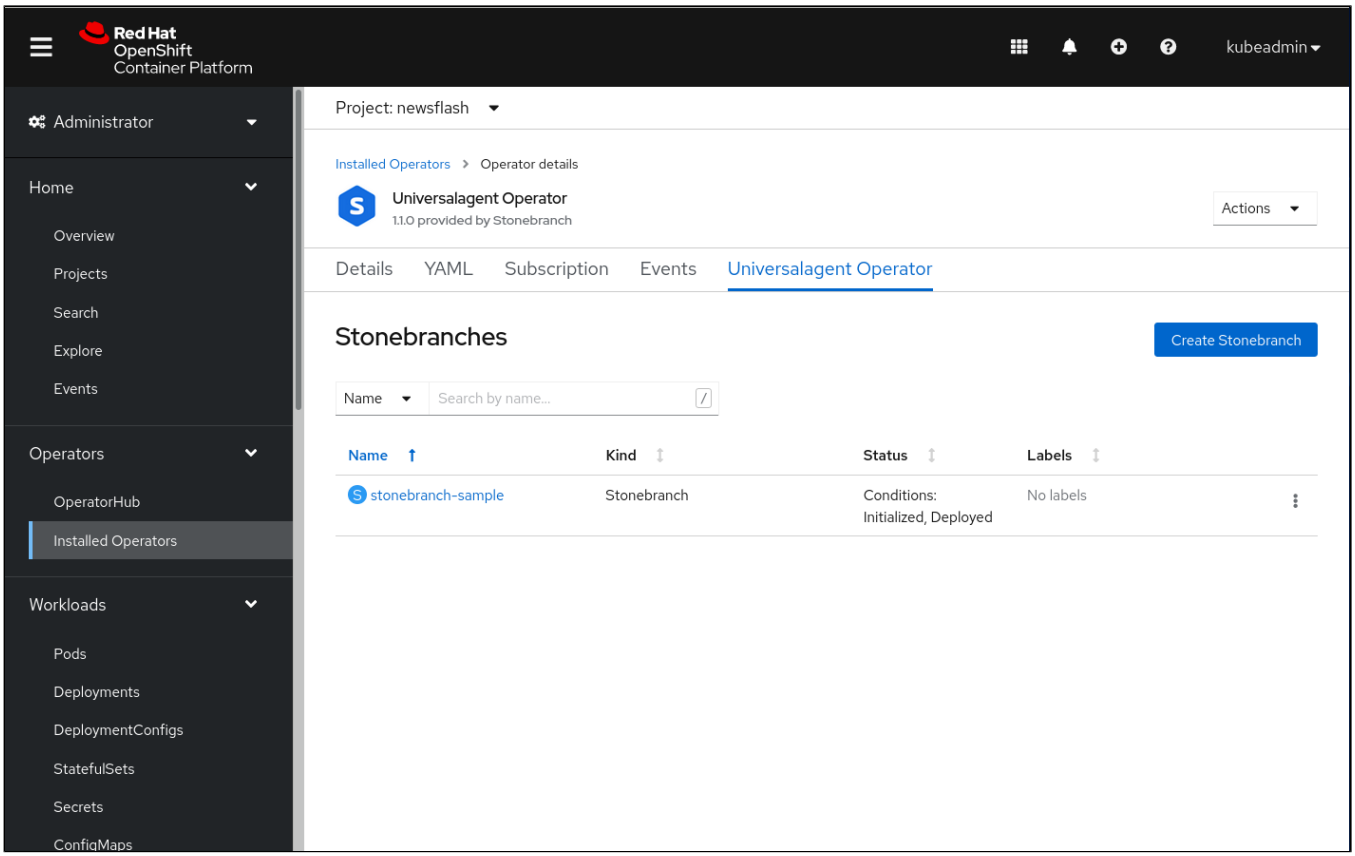
On the *Update Channel* section, select an option.

- On the *Approval Strategy* section, select either *Automatic or Manual*. The approval strategy corresponds to how you want to process operator upgrades.
- Select the namespace of your application e.g.,
- **Click Install**. It may take several minutes for installation to complete.
- Once installation is complete, the status will change from installing to Up to date.
- For further information, see the Red Hat Marketplace Operator documentation.



Step 3 Verification of Operator Installation

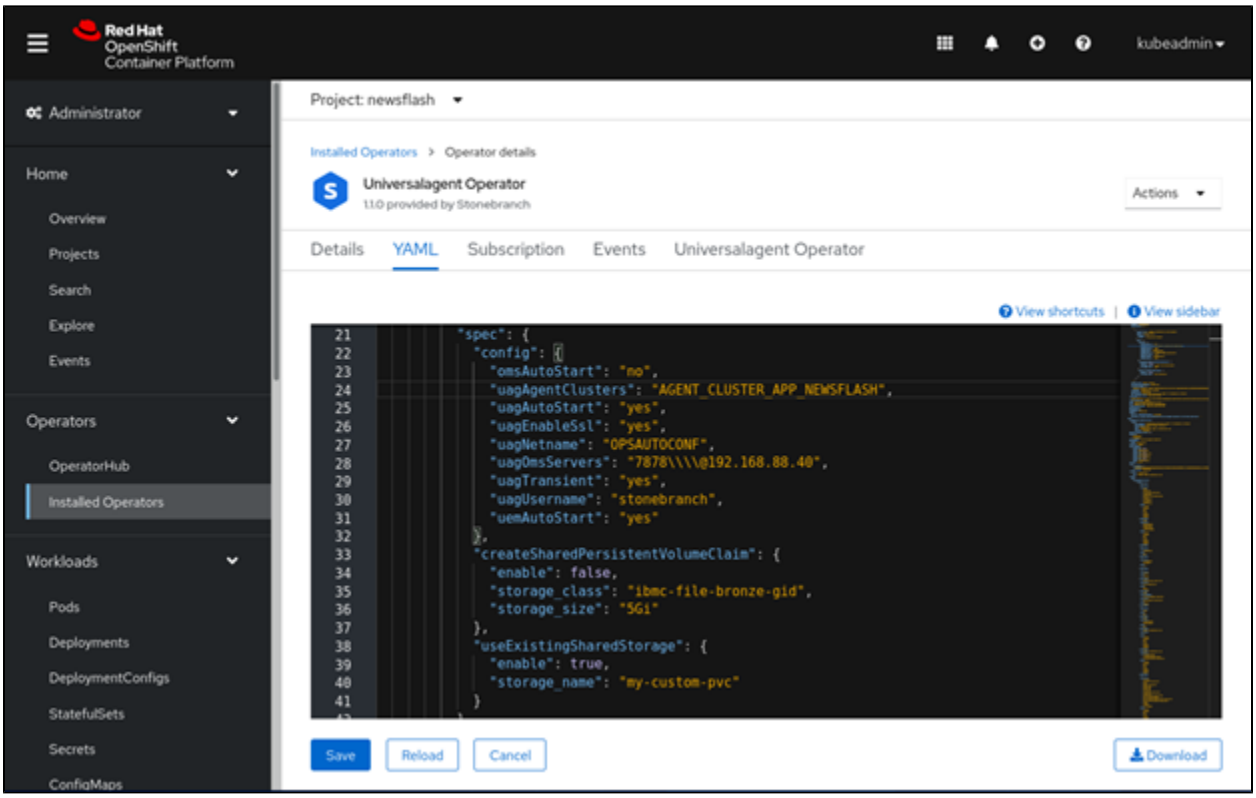
1. When the status changes to Up to date, click the vertical ellipses and select Cluster Console.
2. Open the cluster where you installed the product.
3. Go to **Operators > Installed Operators.**
4. Select the Namespace or Project you installed on.
5. Verify status for product is **Succeeded.**
6. Click the product name to open details.



Step 4 Adjust the Operator YAML (CSV)

When the Operator is installed, it needs to be adjusted to the values according to your Universal Controller landscape.

Any adjustment is done in the YAML file of the Universal Operator.



Universal Controller / OMS Setting

The following tables identifies the Parameters that you must set for Universal Agent to connect to the related Universal Controller OMS.

Parameter	Description
UAGAENGTCLUSTERS	Name of the OPENSHIFT Application (needs to be the same name as the Universal Controller Agent cluster configured in ..)
UAGOMSSERVERS	IP and PORT of the Universal Controller Message Middleware OMS
UAGTRANSIENT	“yes” : Transient means that the agent will be deleted or decommissioned, when the Agent shuts down or goes offline.

In the example shown, Universal Agent would connects to the OMS Port **7878** on the server **192.168.88.40** . The Agent would register to the Universal Controller Agent Cluster: **AGENT_CLUSTER_APP_NEWSFLASH.**

The POD, where the Agent is installed in, would use the Persistent Volume Claim **my-custom-pvc**.

Note

Agent Cluster **AGENT_CLUSTER_NEWSFLASH** must exist in Universal Controller; if it does not, create it manually.

Universal Controller Setting

YAML

```
"omsAutoStart": "no",
"uagAgentClusters": "AGENT_CLUSTER_APP_NEWSFLASH",
"uagAutoStart": "yes",
"uagEnableSsl": "yes",
"uagNetname": "OPSAUTOCONF",
"uagOmsServers": "7878\\\\@192.168.88.40",
"uagTransient": "yes",
"uagUsername": "stonebranch",
"uemAutoStart": "yes"
```

PersistentVolumeClaim Settings

Here you can define if an existing Shared Persistent Volume **("useExistingSharedStorage")** or a new Shared Persistent Volume should be created ("**createSharedPersistentVolumeClaim**").

This example uses the existing Shared Persistent Volume "**my-custom-pvc**", which was created in **Step 1 Prerequisites**.

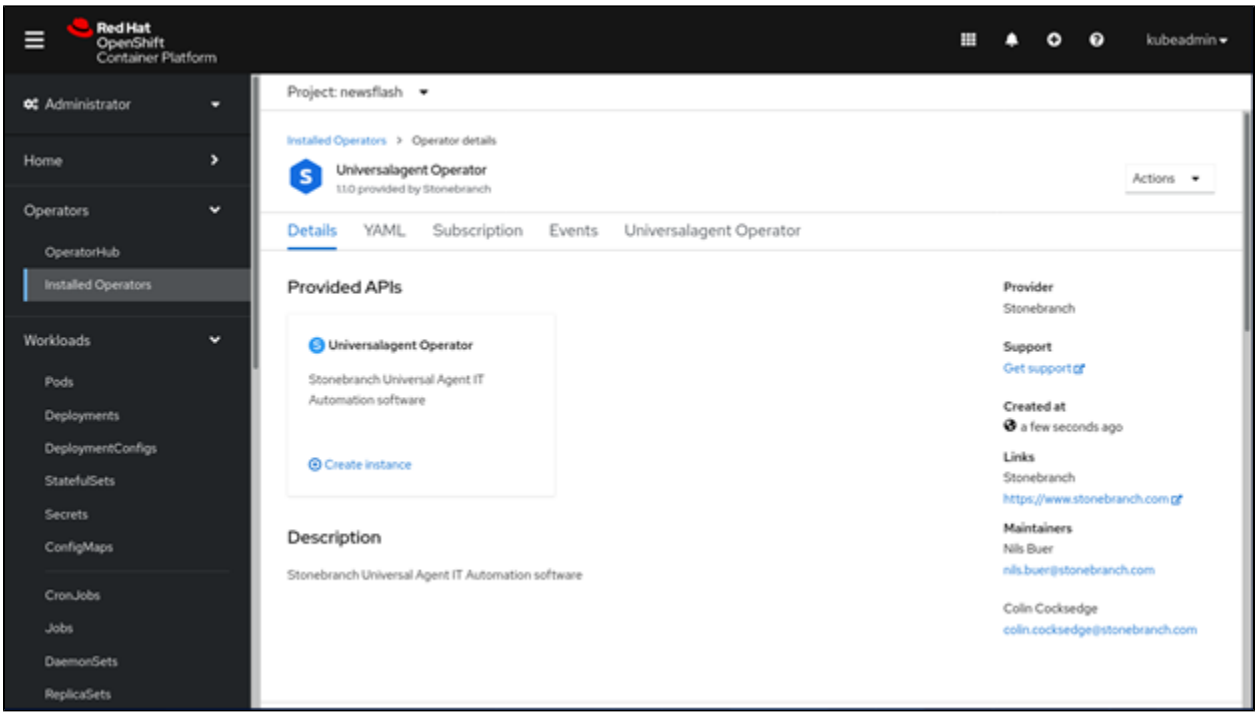
YAML

```
"createSharedPersistentVolumeClaim": {
"enable": false,
"storage_class": "ibmc-file-bronze-gid",
"storage_size": "5Gi"
},
"useExistingSharedStorage": {
"enable": true,
"storage_name": "my-custom-pvc"
```

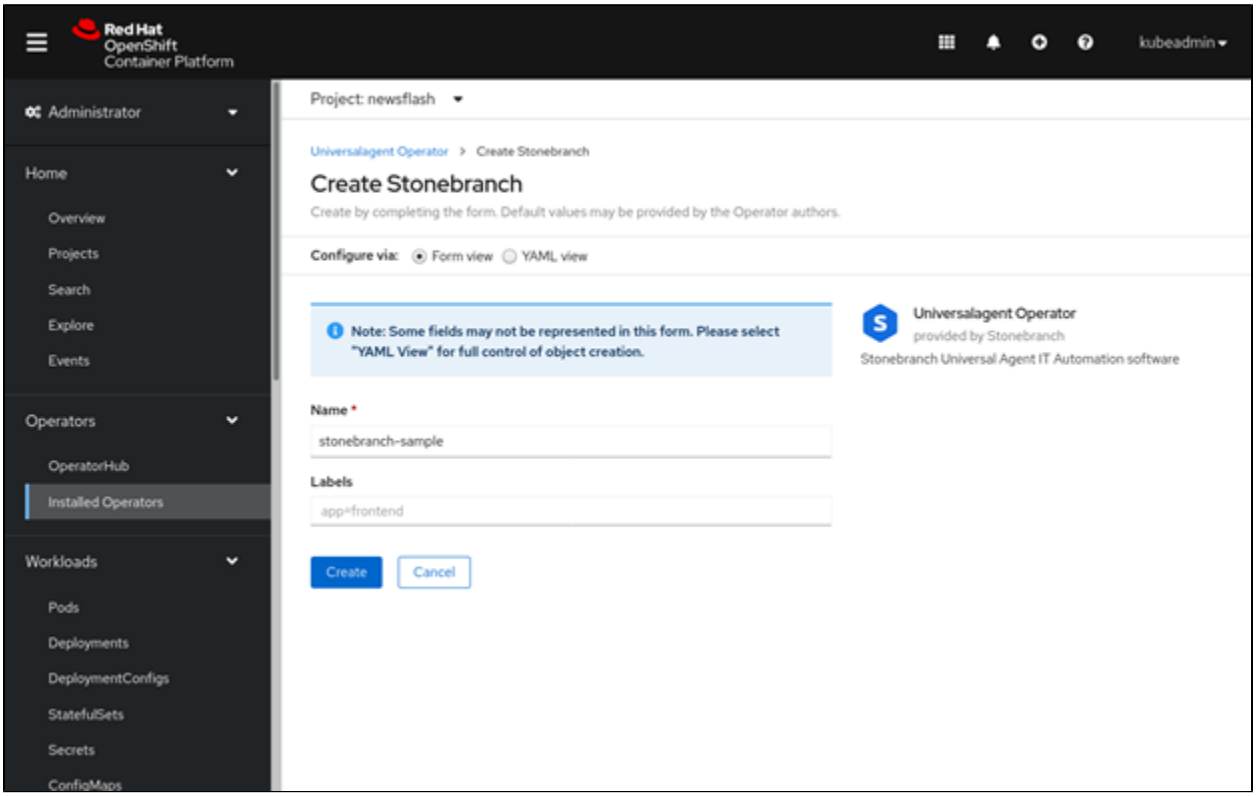
Step 5 Create an Instance

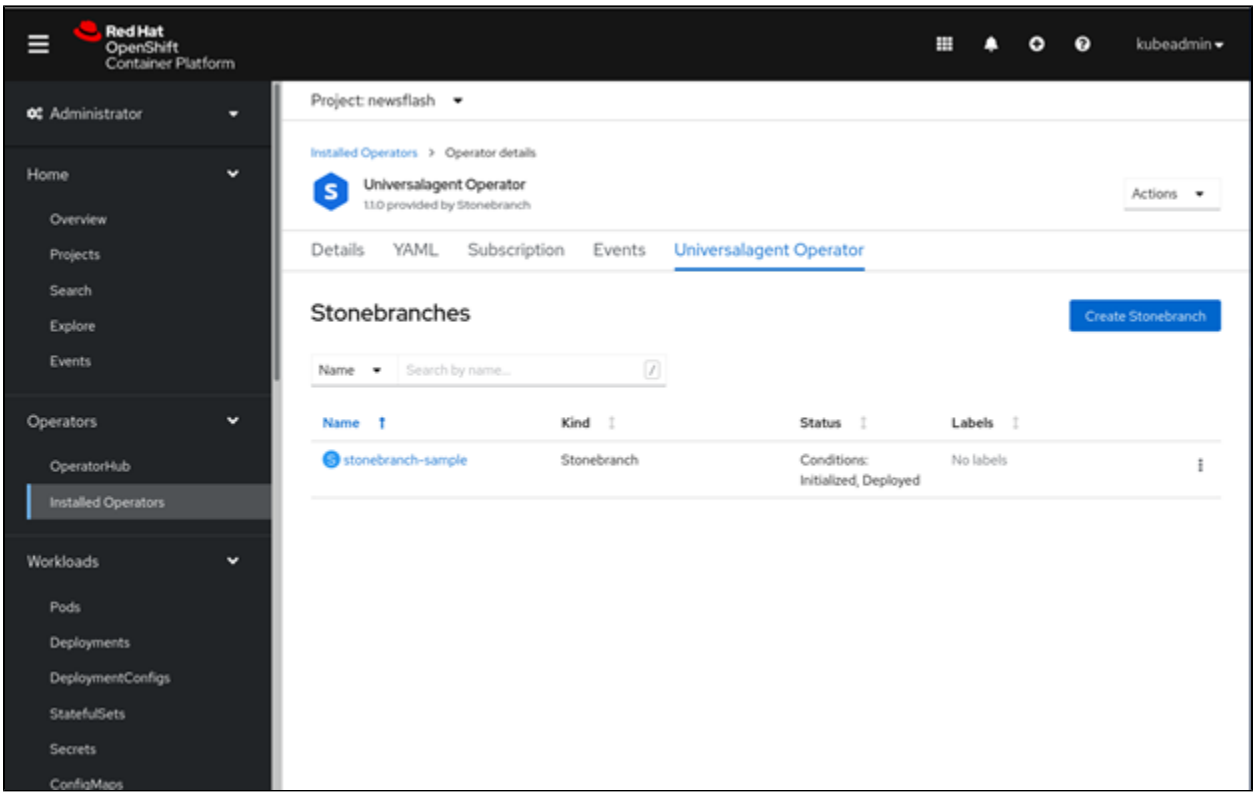
After defining the Universal Agent connection settings in the YAML file, an instance of a POD with a Universal Agent can be created.

In this example, we create an instance named **stonebranch-sample.**



After you click **Create instance**, the **stonebranch-sample-sb-agent** POD in the namespace **newsflash** is created.

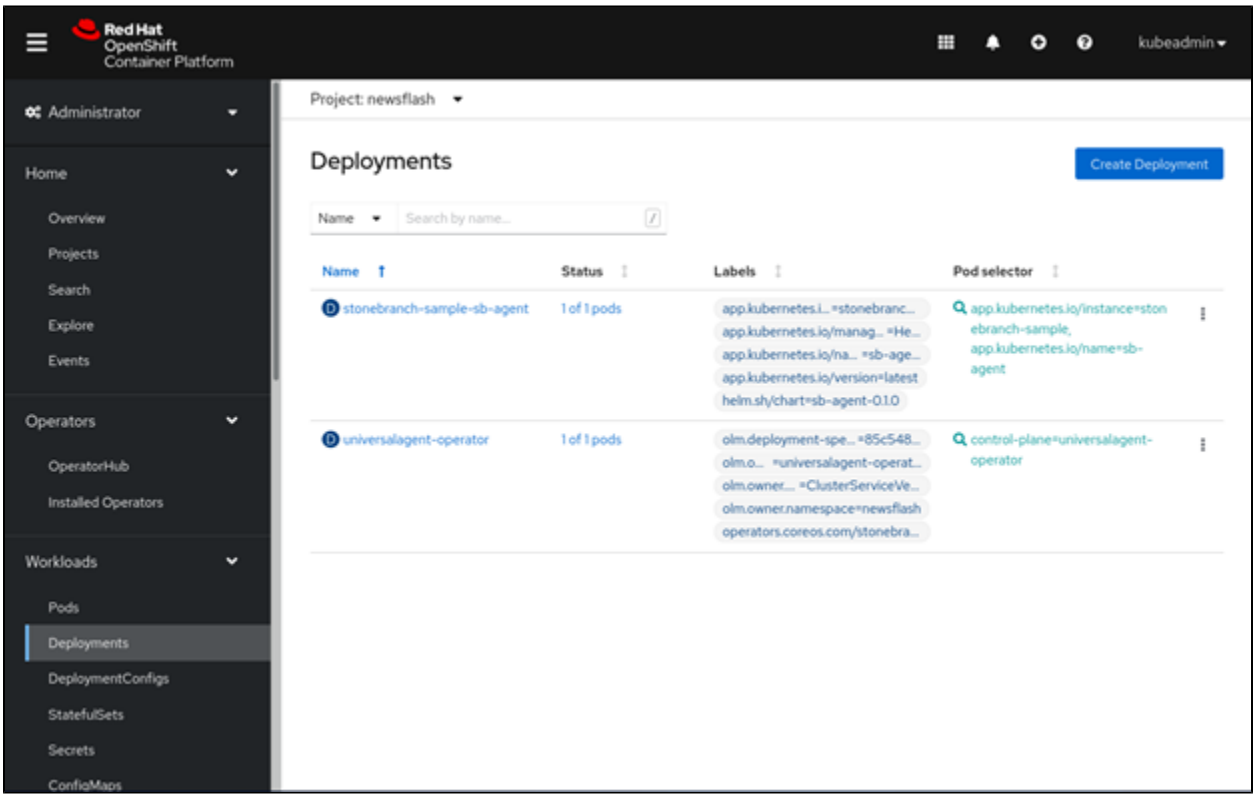




Step 6 Verify the Universal Agent POD

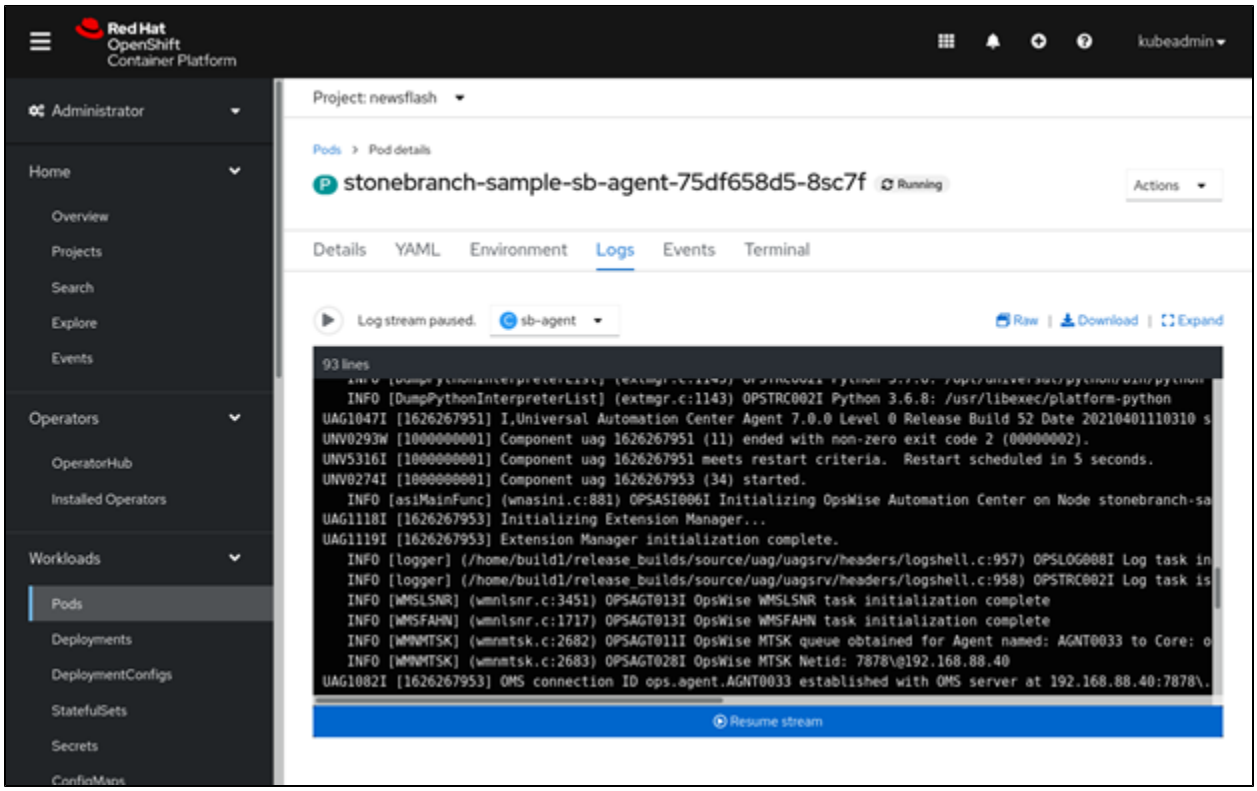
Under **Deployments**, the started **stonebranch-sample-sb-agent** POD in the namespace **newsflash** is shown.

This POD contains a Universal Agent, which can send and receive files.



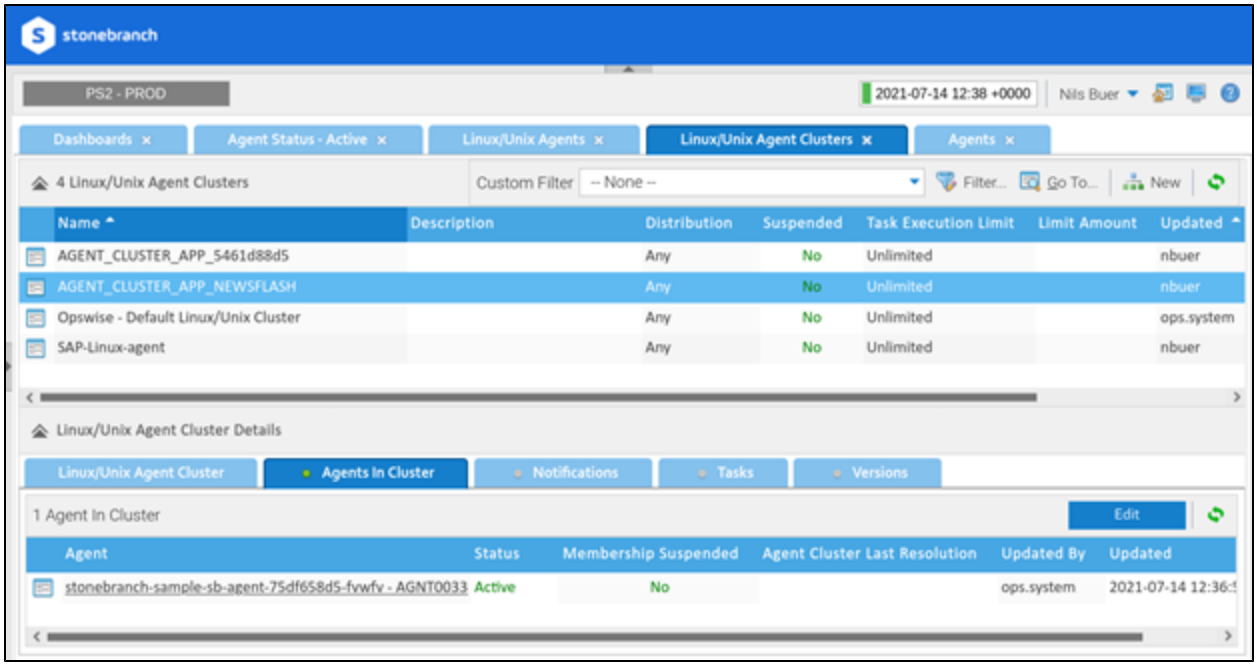
Universal Agent POD Log-file

The log file shows the Agent connected to the OMS of the Universal Controller: *7878@192.168.88.40*



Step 7 Verify that the Universal Agent in OpenShift is Connected to Universal Controller

The Universal Agent should have been automatically registered to the Agent Cluster **AGENT_CLUSTER_NEWSFLASH,** which was defined in the YAML set-up in Step 4.



Note

Agent Cluster **AGENT_CLUSTER_NEWSFLASH** must exist in Universal Controller; if it does not, create it manually.

Summary

Universal Agent has been registered in Agent Cluster **AGENT_CLUSTER_NEWSFLASH.** As a result, it can be used to send and receive files from any external server or cloud storage and save them to the shared persistence storage configured in the YAML set-up in Step 4.

In addition to file transfers, Universal Agent can also trigger any type of API (REST, JDBC, SAP RFC, ...) for scheduling any type of Application installed in OpenShift.

Step 8 Sample Scenario - File Transfer

The following sample scenario shows how a basic file transfer to a shared persistence storage can be set-up, and how the transferred data is accessible by any application having access to the shared persistence storage in OpenShift.

As sample application 2 NGNIX Webserver will be deployed in the newsflash project, which was set-up in Step 1.

The NGNIX Webserver will have access to the shared persistence storage, meaning that if a file has been received by the Universal Agent and saved on a folder in the shared persistence storage, it should also be accessible by the 2 NGNIX webserver for further processing.

Configuration Steps

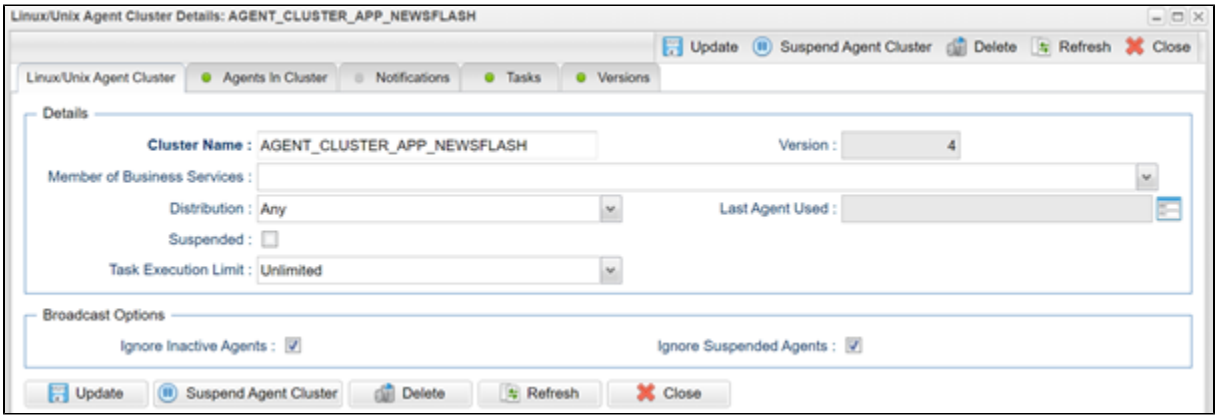
The installation consists of three steps:

1. Define an Agent Cluster in Universal Controller
2. Configure the file transfer workflow using the Universal Controller web GUI.
3. Deploy two sample NGNIX webservers in OpenShift within the newsflash namespace.

1 Define a new Agent Cluster for the OpenShift application newsflash

An agent cluster must be configured in Universal Controller for each Project in OpenShift. When a POD is started in OpenShift, the Universal OpenShift Agent will automatically register with the Universal Controller agent cluster defined in the CSV YAML file in Step 4.

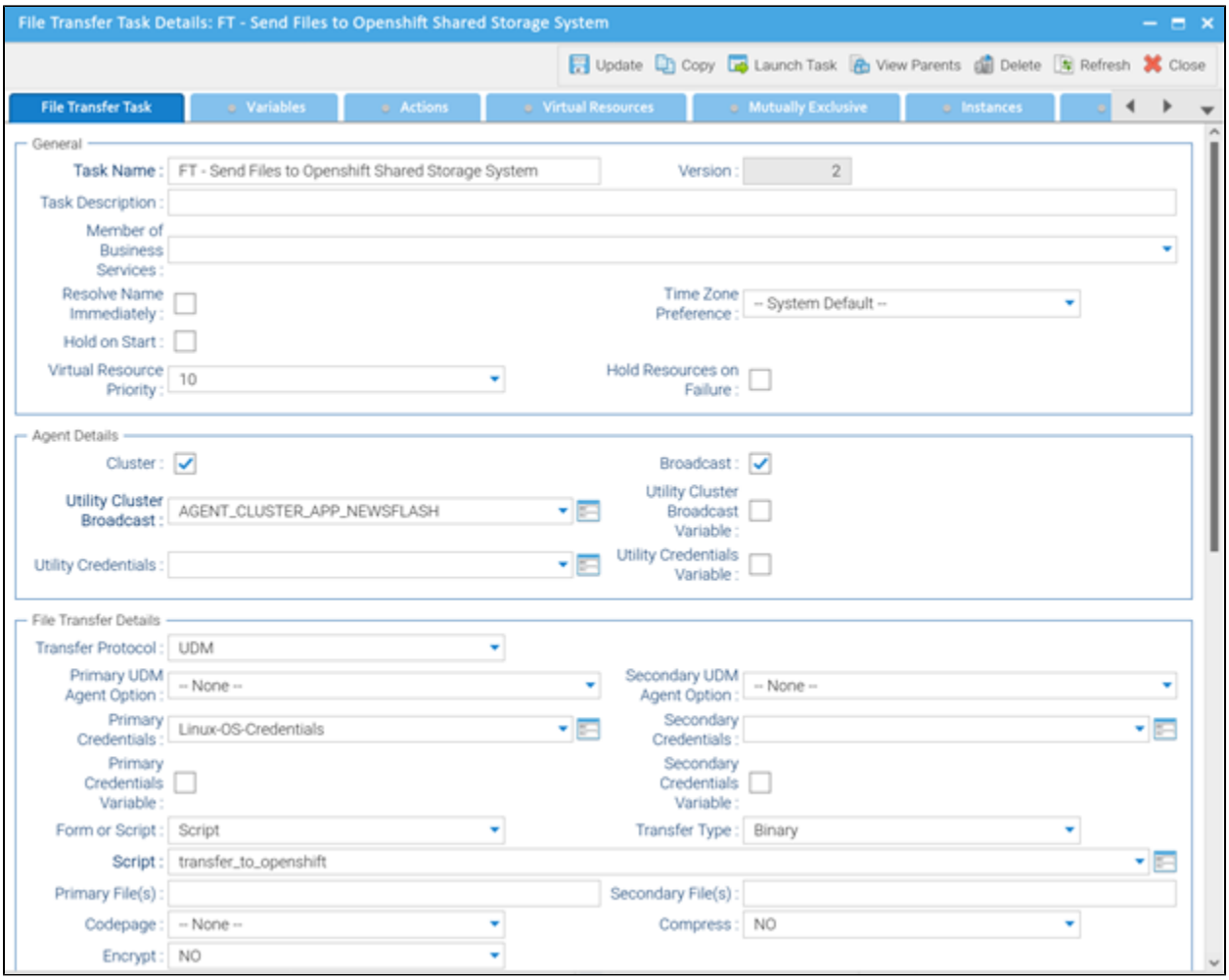
The agent cluster in this example is named *AGENT_CLUSTER_APP_NEWSFLASH.*



2 Universal Controller File Transfer Task

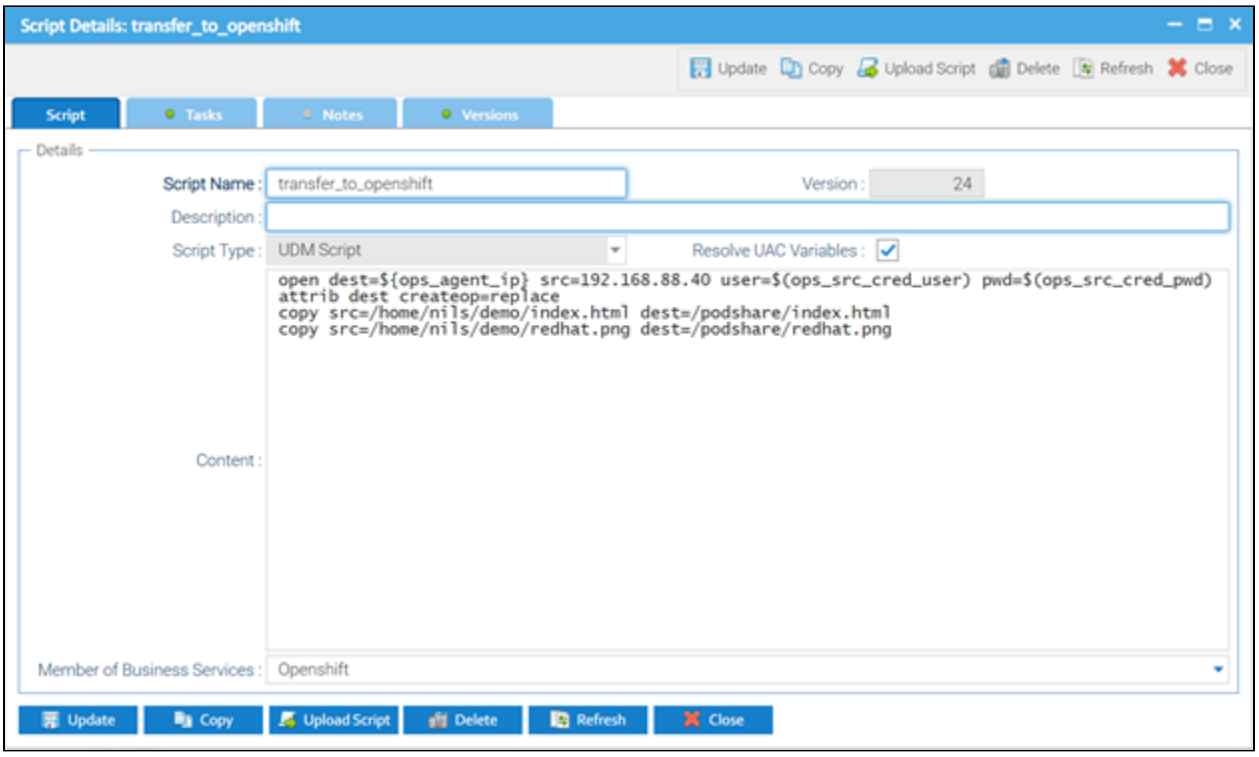
A standard UDM file transfer task can be used to send and retrieve data from the Universal Agent in Openshift. The following File Transfer task transfers two files from an on-premise Linux Server to the OpenShift share folder: **/podshare** (persistence volume claim: my-custom-pvc):

- copy src=/home/nils/demo/index.html dest=/podshare/index.html
- copy src=/home/nils/demo/redhat.png dest=/podshare/redhat.png



File Transfer UDM Script

The following file transfer script is used.




```
open dest=${ops_agent_ip} src=192.168.88.40 user=$(ops_src_cred_user) pwd=$(ops_src_cred_pwd)
attrib dest createop=replace
copy src=/home/nils/demo/index.html dest=/podshare/index.html
copy src=/home/nils/demo/redhat.png dest=/podshare/redhat.png
```

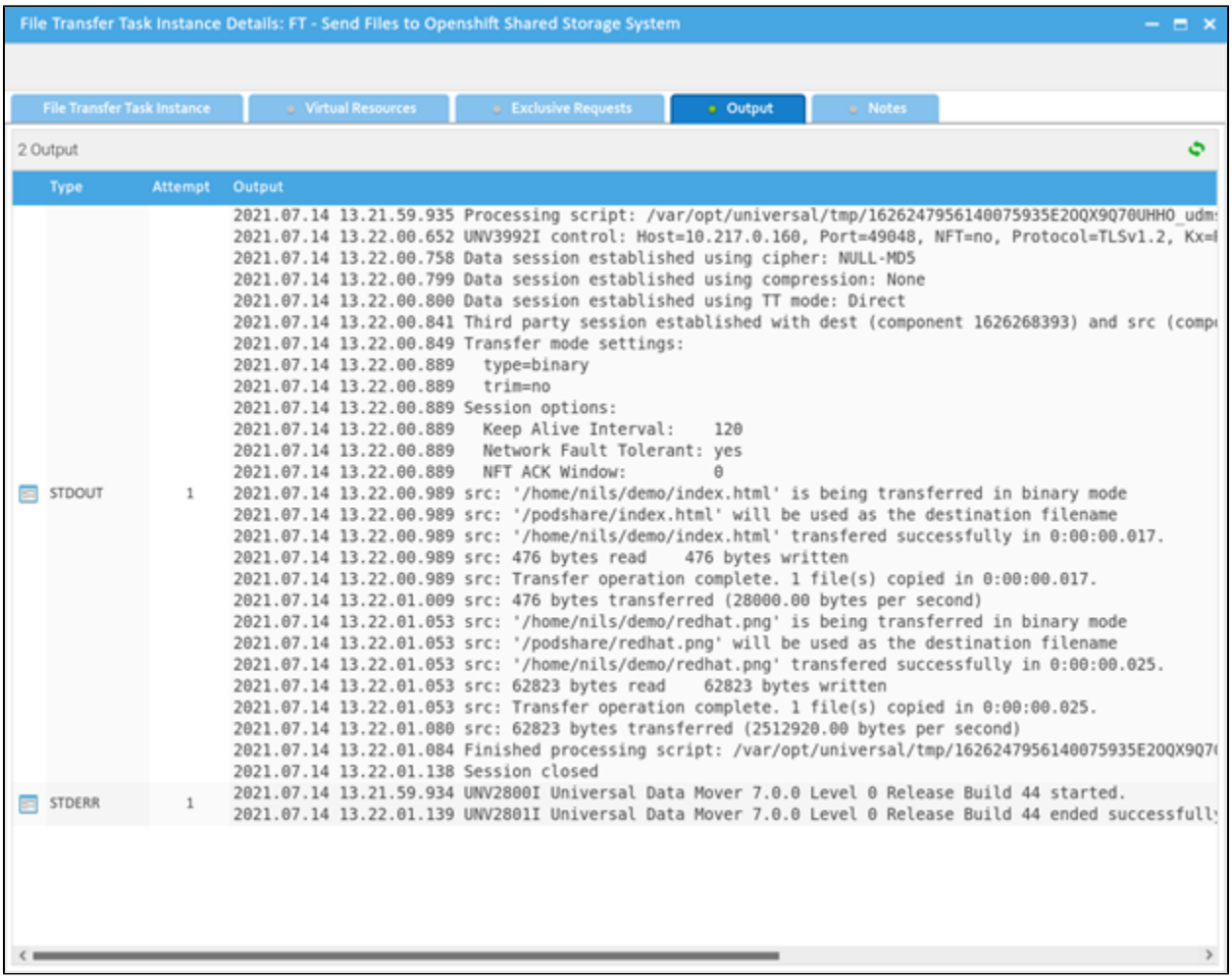
Description

- Source Credentials: Linux-OS-Credentials (adjust according to your server credentials)
- Source Linux Server: 192.168.88.40 (adjust according to your server)
- Source Folder Linux Server: /home/nils/demo/out (adjust according to your server)
- Files to Transfer: index.html, RedHat.png
- Destination Agent Cluster: AGENT_CLUSTER_APP_NEWSFLASH
- Destination Folder in the POD: /podshare/ (this is the mounted POD directory)

The export files of the file transfer task can be found here filetransfer task.

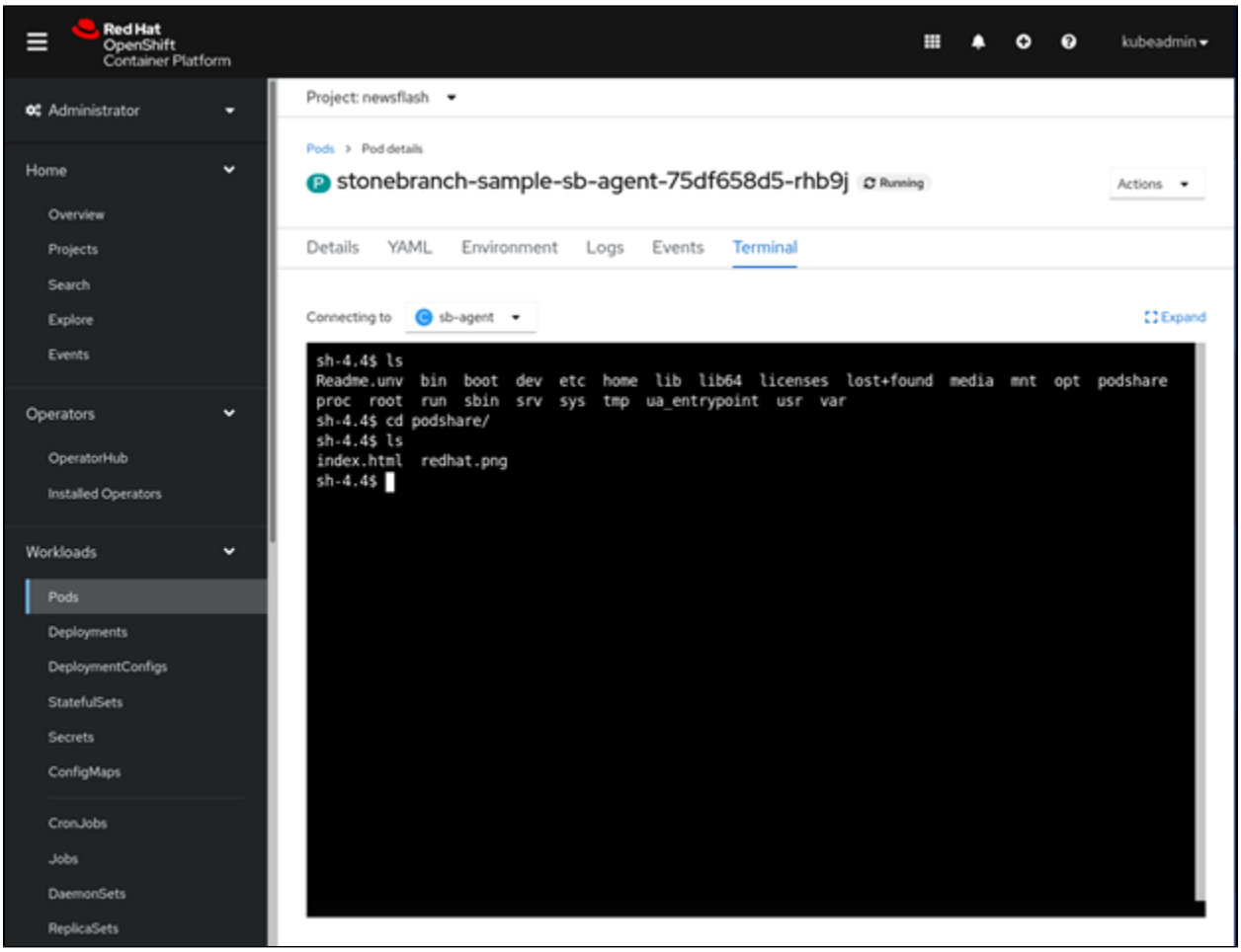
File Transfer Log File

The log file shows that the two files have been transferred to the /podshare/ folder.



Check File Arrival in the POD Using the Openshift Web Console

Both transferred files are available in the folder `/podshare/`.

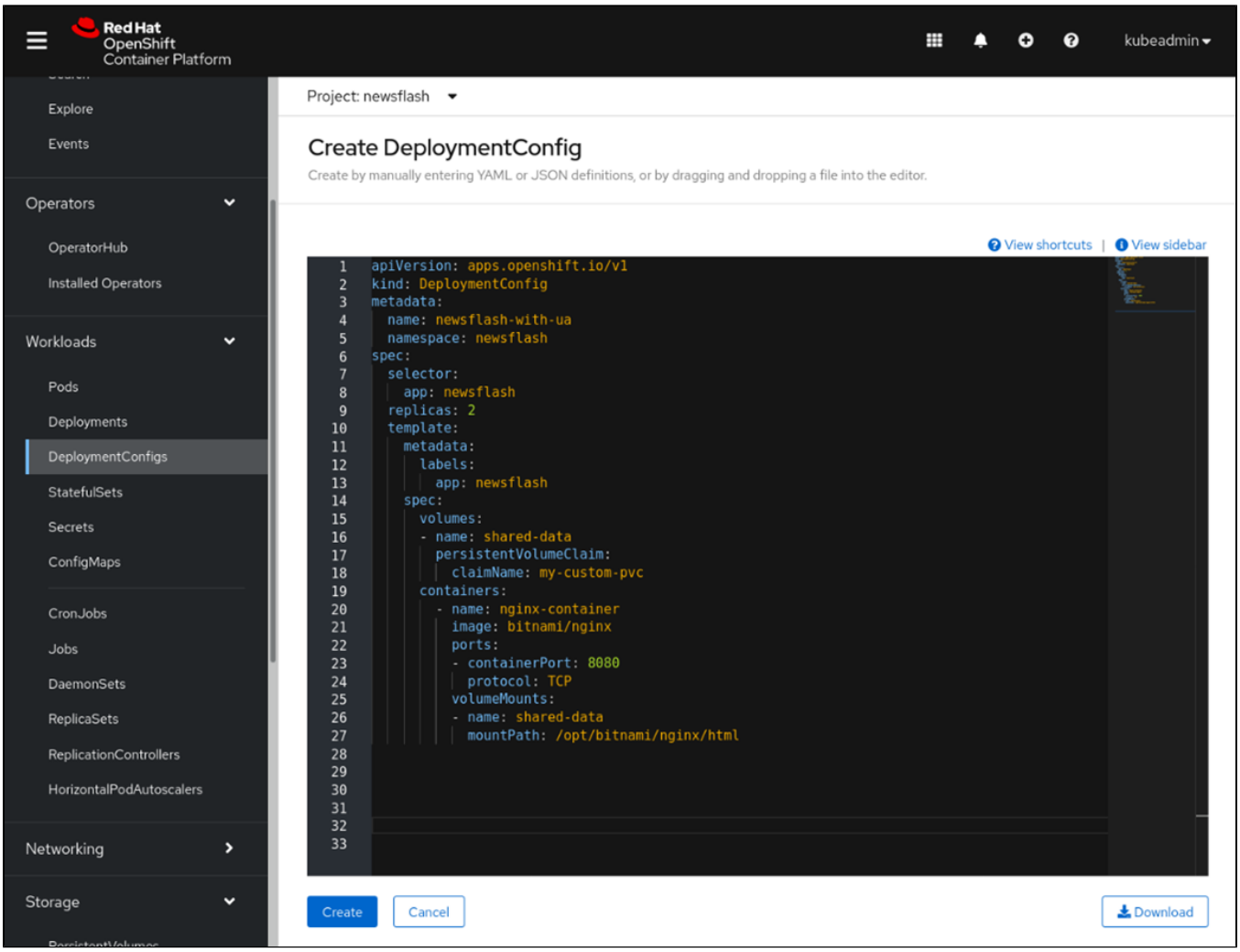


3 Deploy an application, with access to the transferred data

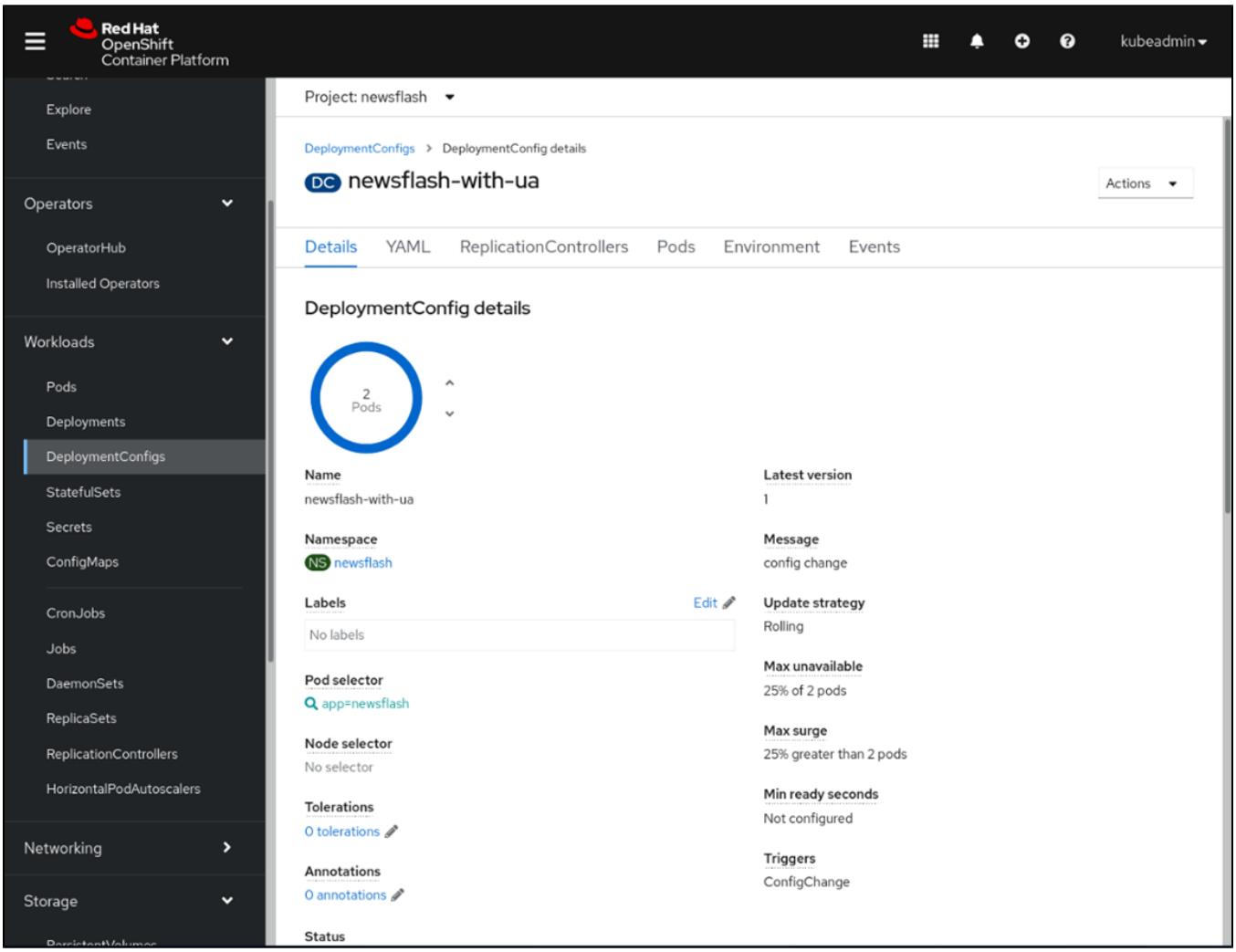
The following deployment YAML deploys 2 PODS with a nginx webserver.

The nginx webserver uses the same persistent Volume Claim as the Universal Agent used for the file transfer.

Persistent Volume Claim: **my-custom-pvc**

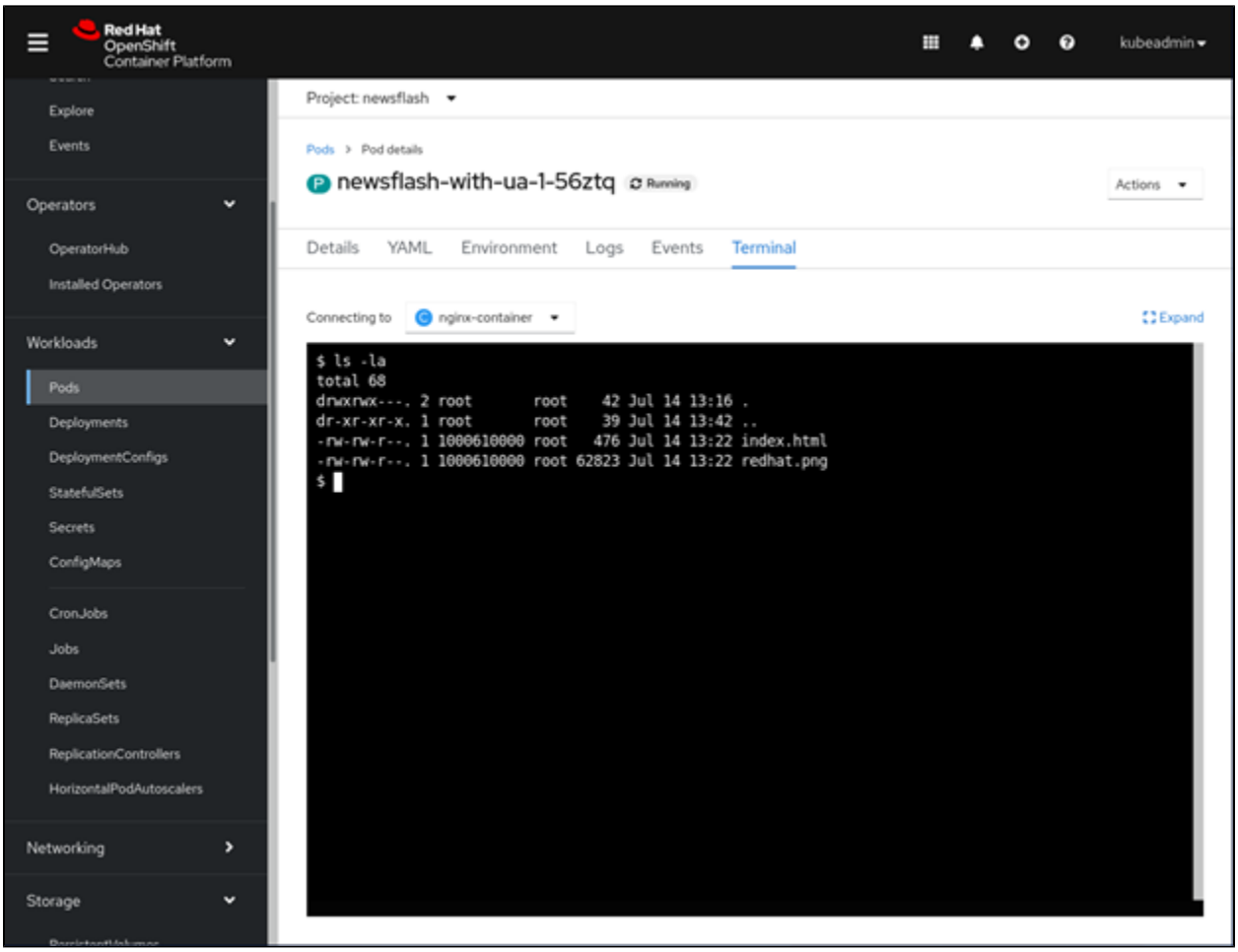


After clicking Create, two nginx webserver PODs have been started based on the given deployment.



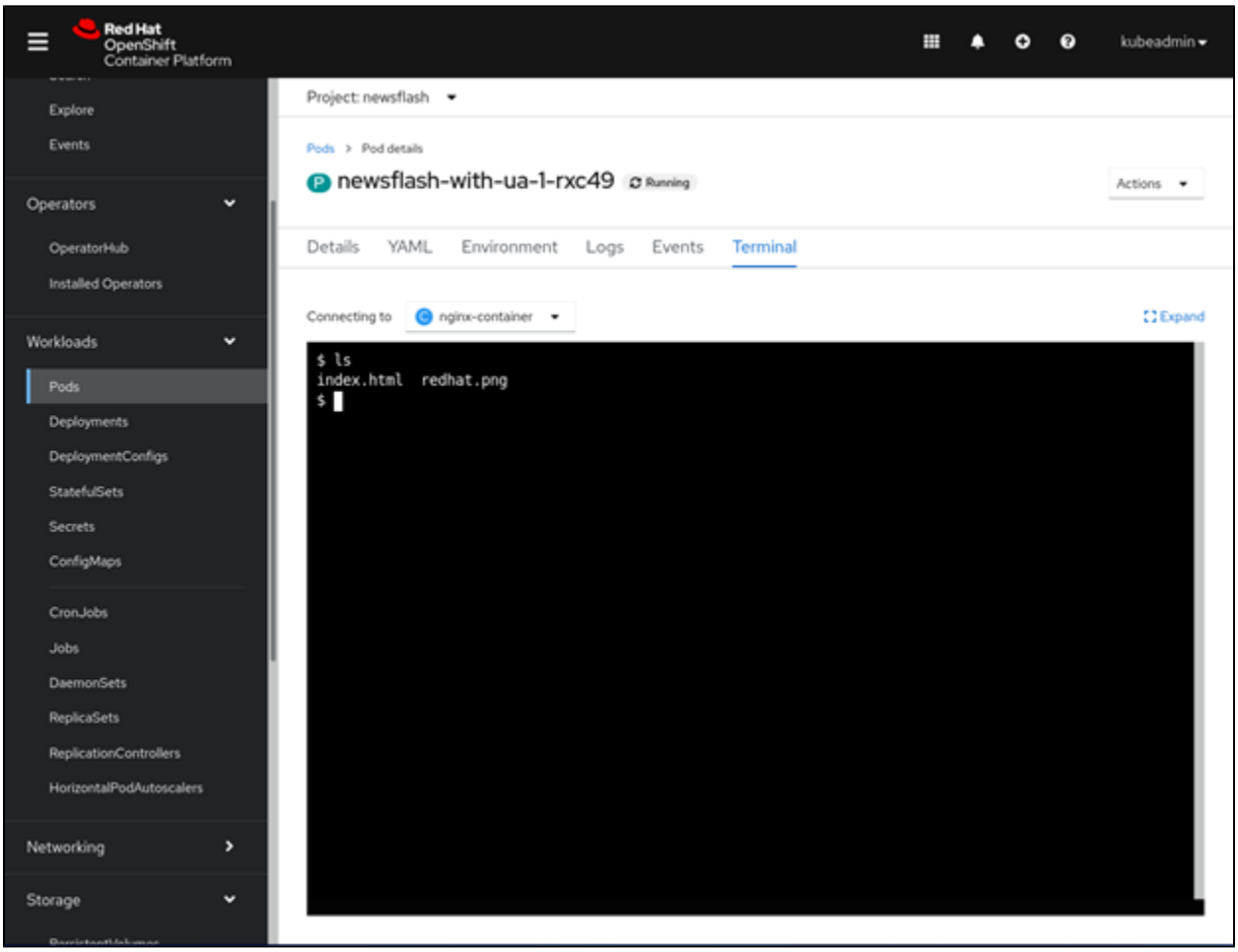
Both PODS can access the two files transferred to the folder /podshare/ because all PODs are using the persistentVolumeClaim: **my-custom-pvc**

First nginx webserver Pod with access to the transferred files



As you can see both files, index.html and redhat.png are available in the first nginx POD

Second nginx webserver Pod with access to the transferred files



As you can see both files index.html and redhat.png are available also in the second nginx POD

The use case showed how to transfer data from a Linux Server to a persistent Volume Claim in an Openshift Cluster. As a result, the transferred data is accessible by any application having access to the shared persistence storage. In our case the two started nginx webserver could access that file transferred from the Linux Server to OpenShift

Summary

The Universal Automation Center, with the introduction of the newly developed Operator for Universal OPENSHIFT Agent, enables the secure and reliable transfer of business data located on the mainframe, on cloud storage platforms, or any server, to any shared persistence storage connect to your OPENSHIFT cluster (and vice versa).