



**stonebranch**

Universal Agent 7.4.x

Universal Agent 7.4.x Container Operations

© 2023 by Stonebranch, Inc. All Rights Reserved.

1. Container Operations .....	3
1.1 Docker Containers .....	3
1.2 Red Hat OpenShift Start-Up Guide .....	9
1.3 Red Hat OpenShift Operator Start-Up Guide .....	26

# Container Operations



Docker Containers

[Downloading a Universal Agent Docker Image](#)

[Docker Environment Variables](#)

[Docker Container Ports](#)

[Universal Agent Logs](#)



Red Hat OpenShift Start-Up Guide

[Introduction](#)

[How to Get Started](#)



Red Hat OpenShift Operator Start-Up Guide

[Introduction](#)

[Prerequisites](#)

[Installation](#)

[Sample](#)



The information on these pages also is located in [Universal Agent 7.3.x User Guide.pdf](#).

## Docker Containers

- [Downloading a Universal Agent Docker Image](#)
- [Docker Environment Variables](#)
- [Docker Container Ports](#)
- [Universal Agent Logs](#)

### 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
```

These images were created with the following

▼ [RHEL Dockerfile](#)

```

# Set the base image to RedHat UBI
FROM redhat/ubi9:latest

# Install libcrypt-compat and procps (for pgrep and pkill)
RUN yum -y install libxcrypt-compat procps

# create label
LABEL name="universal-agent"
LABEL vendor="Stonebranch"
LABEL version="{local_ua_version}"
LABEL release="GA"
LABEL summary="Universal Agent installed in a UBI9 base Image"
LABEL description="Universal Agent can be used for Scheduling and File Transfer"
LABEL author="Stonebranch, Inc."

# create license directory and add product license
RUN mkdir /licenses
ADD /licenses/Terms_and_Conditions.pdf /licenses

# Install ua
ARG ua_version
ADD /ua_install/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 --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

# Setup UBI Python
COPY ./ua_requirements /
RUN mkdir /opt/universal/python3.6 && mkdir /opt/universal/python3.6/bin && ln -s /bin/python3 /opt
/universal/python3.6/bin/python3 && yum -y update && yum -y install python3-pip
RUN /bin/python3 -m pip install -r ua_requirements && rm ./ua_requirements

# Update path
ENV PATH "$PATH:/opt/universal/bin"

# Set Default userid
USER 10010

# Add entrypoint script
COPY ./ua_entrypoint /

# Set entrypoint
ENTRYPOINT [ "./ua_entrypoint" ]

```

▼ [Debian Dockerfile](#)

```

# Set the base image to Debian
FROM debian:latest

# create label
LABEL name="universal-agent"
LABEL vendor="Stonebranch"
LABEL version="{local_ua_version}"
LABEL release="GA"
LABEL summary="Universal Agent installed in a Debian based Image"
LABEL description="Universal Agent can be used for Scheduling and File Transfer"
LABEL author="Stonebranch, Inc."

# create license directory and add product license
RUN mkdir /licenses
ADD /licenses/Terms_and_Conditions.pdf /licenses
# Install procps (for pgrep and pkill)
RUN apt-get update && apt-get install -y procps
# Install ua
ARG ua_version
ADD /ua_install/sb-{$ua_version}-linux-3-x86_64-deb.tar.Z /tmp
RUN zcat /tmp/sb-{$ua_version}-linux-3-x86_64-deb.tar.Z | tar xvf - && ./unvinst --network_provider oms
--oms_servers 7878@localhost --oms_port 7878 --security inherit --ubroker_start no && rm unvinst *.deb
*.tar upimerge.sh upimerge.log usrmode.inc install.log /tmp/sb-{$ua_version}-linux-3-x86_64-deb.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

# Setup Python
COPY ./ua_requirements /
RUN mkdir /opt/universal/python3.6 && mkdir /opt/universal/python3.6/bin && ln -s /usr/bin/python3 /opt
/universal/python3.6/bin/python3 && apt-get install -y python3-pip
RUN python3 -m pip install -r ua_requirements && rm ./ua_requirements

# Update path
ENV PATH "$PATH:/opt/universal/bin"

# Set Default userid
USER 10010

# Add entrypoint script
COPY ./ua_entrypoint /

# Set entrypoint
ENTRYPOINT [ "./ua_entrypoint" ]

```

#### ▼ [Entrypoint Script](#)

```

#!/bin/bash

echo "Entrypoint Version 1.8"

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

# Set UBroker Message Level
if [ ! -z "$UBRMSGLEVEL" ]; then
sed -ri "s/^message_level.*/message_level $UBRMSGLEVEL/g" /etc/universal/ubroker.conf
fi

# Set UAG Server Message Level
if [ ! -z "$UAGMSGLEVEL" ]; then
sed -ri "s/^message_level.*/message_level $UAGMSGLEVEL/g" /etc/universal/uags.conf
fi

# Set UDM Server Message Level
if [ ! -z "$UDMMSGLEVEL" ]; then
sed -ri "s/^message_level.*/message_level $UDMMSGLEVEL/g" /etc/universal/udms.conf
fi

# Set UCMD Server Message Level
if [ ! -z "$UCMMSGLEVEL" ]; then
sed -ri "s/^message_level.*/message_level $UCMMSGLEVEL/g" /etc/universal/ucmds.conf
fi

# Set UAG Server Log Level
if [ ! -z "$UAGLOGLEVEL" ]; then
sed -ri "s/^loglvl.*/loglvl $UAGLOGLEVEL/g" /etc/universal/uags.conf
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
OMS AUT OST ART	Specifies whether the Universal Broker starts an OMS server. Default = no	OMSAUTOSTART=yes

UAG AGE NTC LUS TERS	List of Universal Controller Agent Clusters to join automatically.  Default = 'Opwise - Default Linux/Unix Cluster, Opwise - Default Windows Cluster'	UAGAGENTCL USTERS='Agen t Cluster 1, Agent Cluster 2'
UAG AUT OST ART	Specifies whether the Universal Broker starts a UAG server.  Default = yes	UAGAUTOSTA RT=no
UAG BUSI NES SSE RVIC ES	Specify the list of UC Business Services that this Agent will join when registering with the Universal Controller.	UAGBUSINESS SERVICES: ='Business Service 1, Business Service 2'
UAG ENA BLE SSL	Specifies whether the SSL/TLS protocol is used for network communication between UAG and OMS.  Default = no	UAGENABLES SL=yes
UA GNE TNA ME	Sets the Agent ID to be used when the Universal Agent registers / connects to a Universal Controller Instance.  Default = OPSAUTOCONF	UAGNETNAME =UADKR001
UAG OMS SER VERS	Specifies one or more OMS server addresses.  Default = 7878@localhost	UAGOMSSERV ERS=7878\@o msserver1, 7878\@omsser ver2
UAG TRA NSIE NT	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.   <b>Note</b> 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:  <ul style="list-style-type: none"> <li>• <b>yes</b> Agent is registered as a transient Agent.</li> <li>• <b>no</b> Agent is registered as a regular, persistent Agent.</li> </ul> Default = no.	UAGTRANSIEN T=yes
UEM AUT OST ART	Specifies whether the Universal Broker starts a UEM server.  Default = yes	UEMAUTOSTA RT=no
UBR MSG LEV EL	Specify the Universal Broker message level.  Default = INFO	UBRMSGLEVE L=TRACE

UAG MSG LEV EL or UAG LEV EL	Specify the UAG Server message level. Default = INFO	UAGMSGLEVE L=TRACE
UDM MSG LEV EL	Specify the UDM Server message level. Default = INFO	UDMMSGLEVE L=TRACE
UCM MSG LEV EL	Specify the UCMD Server message level. Default = INFO	UCMMSGLEVE L=TRACE
UAG LOG LEV EL or UAG LOG LVL	Specify the UAG Server Log Level. Default = I	UAGLOGLEVE L=T

## 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 latest version 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:latest
```

## Red Hat OpenShift Start-Up Guide

- [Introduction](#)
  - [Use Case Description](#)
  - [Key Features](#)
  - [Solution Architecture](#)
  - [General Description of File Transfer Process](#)
- [How to Get Started](#)
  - [USE CASE: “News-Flash Application”](#)
  - [Prerequisites](#)
  - [Configuration Steps](#)
  - [Set Up a File Transfer Task in Universal Controller](#)
  - [Add the Universal Sidecar Container to the Application Deployment Script](#)
  - [Deploy the Open Shift Application](#)
  - [Running the File Transfer](#)
  - [Options to Trigger the File Transfer Scenario](#)
  - [Integration of File Transfer into Microservices Architectures](#)
  - [Security and Auditability](#)
- [Document References](#)
- [Summary and Benefits](#)

### 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:

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

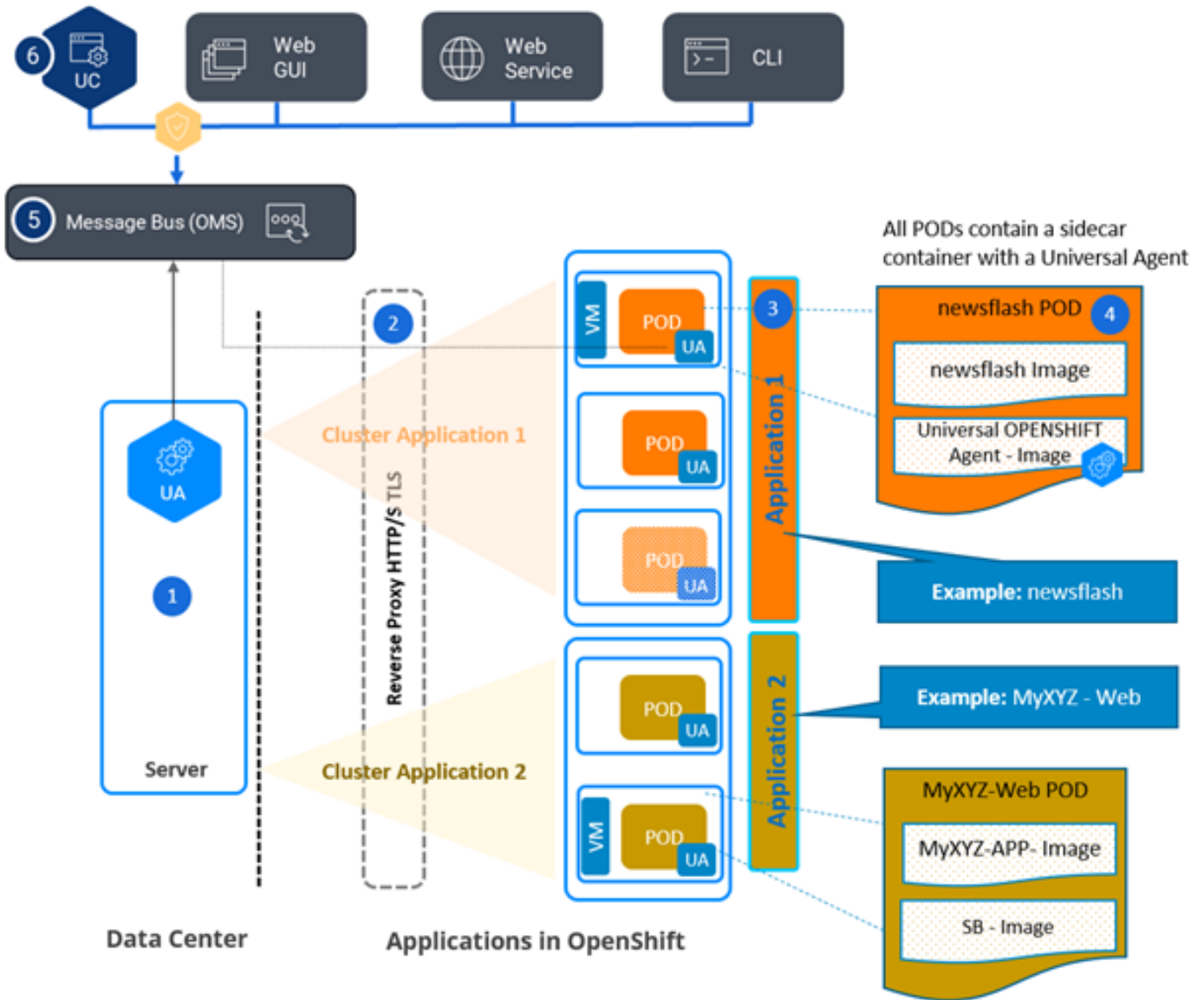
A Universal Agent specifically built for the Open Shift environment is called a Universal Open Shift Agent.

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.

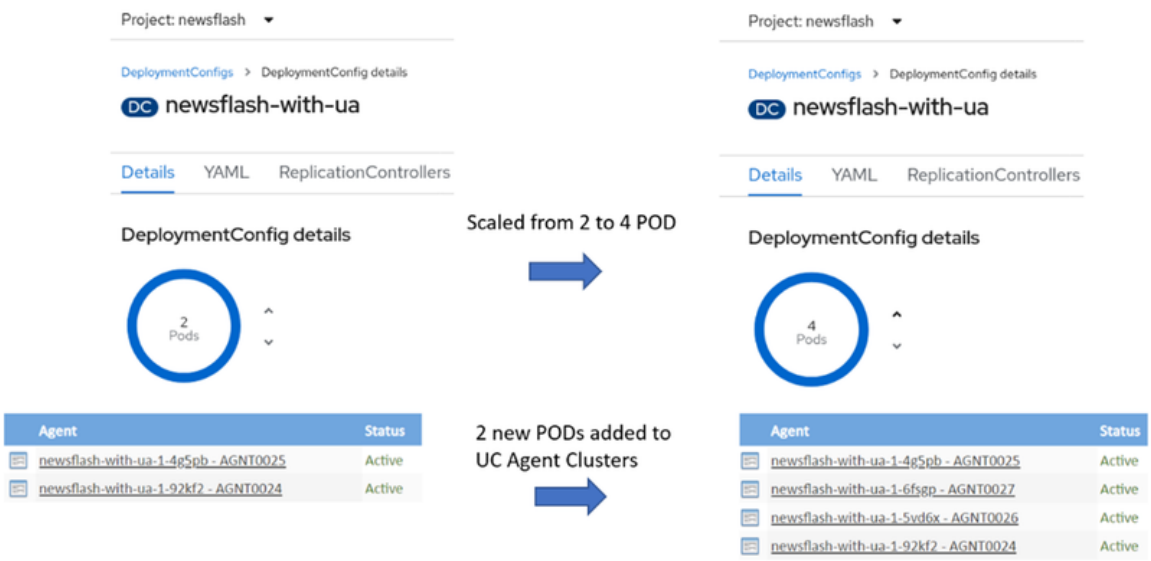


*Provide business data from an on-premise server to an application running in a POD.*

### File Transfer Process Steps

Step	Component	Description
<b>Customer Data Centre</b>		
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 installed in the Data Center, and it can also connect to any public cloud storage for file transfer.

## Open Shift PaaS

<p>S t e p 2</p>	<p>R e v e r s e P r o x y</p> <p>Due to security regulations, all communication from and to Open Shift should be sent via reverse HTTPS proxy.</p>
<p>S t e p 3</p> <p>A p p l i c a t i o n I n s t a n c e c l u s t e r</p>	<p>An application is deployed in a POD.</p> <p>If the application load <b>increases</b>, the Open Shift orchestration platform allows users to dynamically <b>scale up</b> the number of application in ir POD.</p>  <p style="text-align: center;"><b>Scaling Up from 2 to 4 Application Instances in Open Shift and Automatically in Universal Controller</b></p> <p>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 to the Universal Controller agent cluster related to the application.</p> <p>If the load decreases, application instances, (that is, PODs) can be stopped in Open Shift.</p>
<p>S t e p 4</p> <p>D e p l o y m e n t C o n t a i n e r</p>	<p>P O D</p> <p>All PODs contain a sidecar container with a Universal Open Shift Agent. Each sidecar container is based on the Red Hat UBI image with installed inside. The latest version of the image can be retrieved from the docker registry or the Red Hat <a href="#">catalog</a>.</p> <p>Detailed documentation of the image can be found <a href="#">here</a> [1].</p> <p>When a POD (and, respectively, the sidecar container) is started, the Universal Open Shift Agent of the container automatically registers to the cluster dedicated to the application.</p> <p>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 if a dedicated Universal Agent cluster is created, containing the Universal Open Shift agents of all started instances of the application.</p> <p>The example in <a href="#">General Description of File Transfer Process</a>, above, shows two applications:</p> <ol style="list-style-type: none"> <li>1. newflash</li> <li>2. MyXYZ – APP</li> </ol> <p>Each instance of an application is represented by one POD.</p> <p>As soon as the Universal Open Shift Agent of the sidecar container is assigned to the related Universal Controller agent cluster, all related files from/to any other Universal Agent installed on any server. In addition, the application running in the POD can be scheduled like any other application and be included in any automated business process.</p> <p>The Universal Agent cluster supports file transfers to just one agent (POD), or to all agents (that is, started PODs related to an application).</p>

Steps 5	OMS	<p>OMS (Controller message bus).</p> <p>Universal Agent in the POD connects on start-up automatically to the Controller message bus. The IP address of OMS is provided in the POD for the sidecar container.</p>
Steps 6	Universal Controller	<p>Universal Controller is the web-based workflow, reporting, and orchestration engine.</p> <p>For each application, one Universal Controller Agent Cluster must be configured.</p> <p>When the Universal Agent in the POD is started, it registers automatically to the UC Agent Cluster, which is configured in the deployment sidecar container.</p>

## 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	<p>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 <a href="#">here</a> [2].</p> <p>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.</p> <p>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.</p>
Universal Agent 7.x or Above	<p>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.</p> <p>For the installation of a Universal Agent on a Linux server, please refer to the Universal Agent installation guide found <a href="#">here</a> [4].</p>
Open Shift 4.x	<p>This scenario uses Open Shift 4 deployed via a <a href="#">Red Hat CodeReady Container</a> (CRC) [3]. A CRC enables Open Shift to run on a local laptop or server. However, any Open Shift 4 deployment could be used.</p>
Linux Server	<p>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.</p> <p>For information on the installation of a Universal Agent on a Linux server, please refer to the Universal Agent installation guide found <a href="#">here</a> [4].</p>

## 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.

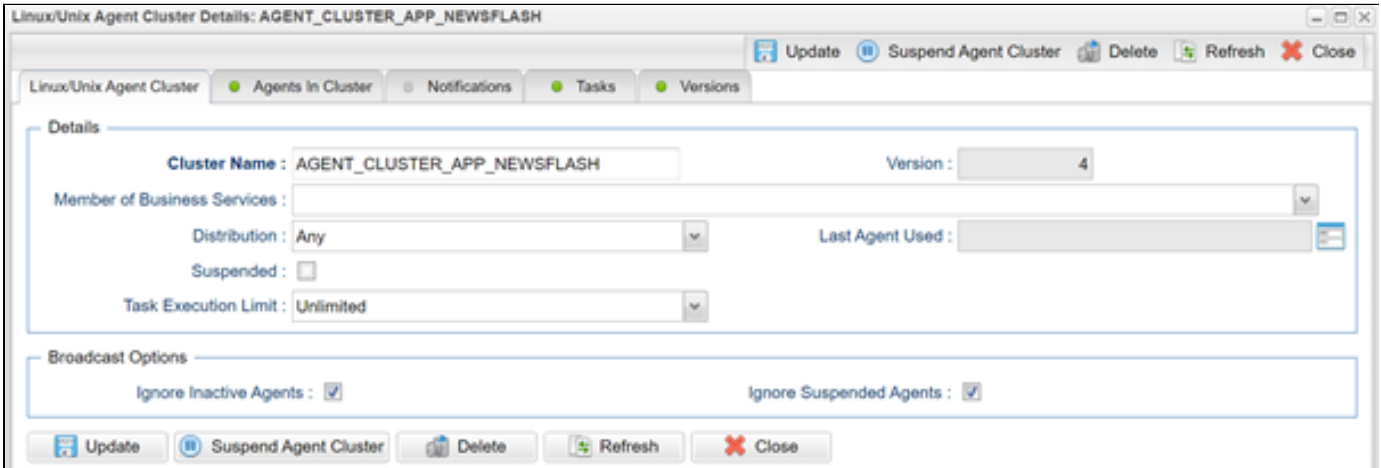
Setting up the file transfer task in Universal Controller requires two steps:

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.

**File Transfer Task Details: FT - Update newflash on OPENSIFT**

Update Copy Launch Task View Parents Delete Refresh Close

File Transfer Task Variables Actions Virtual Resources Mutually Exclusive Instances

**General**

Task Name: FT - Update newflash on OPENSIFT Version: 8

Task Description: use case Openshift - send to multiple agents

Member of Business: [Dropdown]

Services: [Dropdown]

Resolve Name Immediately:  Time Zone Preference: -- System Default --

Hold on Start:

Virtual Resource Priority: 10 Hold Resources on Failure:

**Agent Details**

Cluster:  Broadcast:

Utility Cluster Broadcast: AGENT\_CLUSTER\_APP\_NEWSFLASH Utility Cluster Broadcast Variable:

Utility Credentials: [Dropdown] Utility Credentials Variable:

**File Transfer Details**

Transfer Protocol: UDM

Primary UDM Agent Option: -- None -- Secondary UDM Agent Option: -- None --

Primary Credentials: Linux-OS-Credentials Secondary Credentials: [Dropdown]

Primary Credentials Variable:  Secondary Credentials Variable:

Form or Script: Script Transfer Type: Binary

Script: transfer\_to\_openshift

**Universal Controller File Transfer Task**

**Script Details: transfer\_to\_openshift**

Update Copy Upload Script Delete Refresh Close

Script Tasks Notes Versions

**Details**

Script Name: transfer\_to\_openshift Version: 24

Description: [Empty]

Script Type: UDM Script Resolve UAC Variables:

```
open dest=${ops_agent_ip} src=192.168.88.40 user=${ops_src_cred_user} pnd=${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
```

Update Copy Upload Script Delete Refresh Close

**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 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'
```

Sidecar container section

← Image location e.g., Dockerhub or Red Hat Catalog

← file system of the POD accessible by sidecar and appl. container

← Transient means that the agent will be deleted or decommissioned when the Agent shuts down or goes offline.

← Agent Cluster This Agent connects to

← OMS Server this Agent connects to

**Universal Open Shift Agent deployment YAML**

Configure the following three parameters in the deployment file:

Parameter	Description
UAGAENGTCLESTERS	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.

```

...
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'

```

Newsflash container section

Sidecar container section

Shared POD filesystem between Application and Universal Agent container

Mounted folder, to where the files are transferred to

Connection Parameters for the Universal Agent  
To connect to the Universal Controller

**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.

```

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'

```

Newsflash container

Sidecar container

#### 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.

Red Hat OpenShift Container Platform

You are logged in as a temporary administrative user. Update the [cluster OAuth configuration](#) to allow others to log in.

Project: newsflash

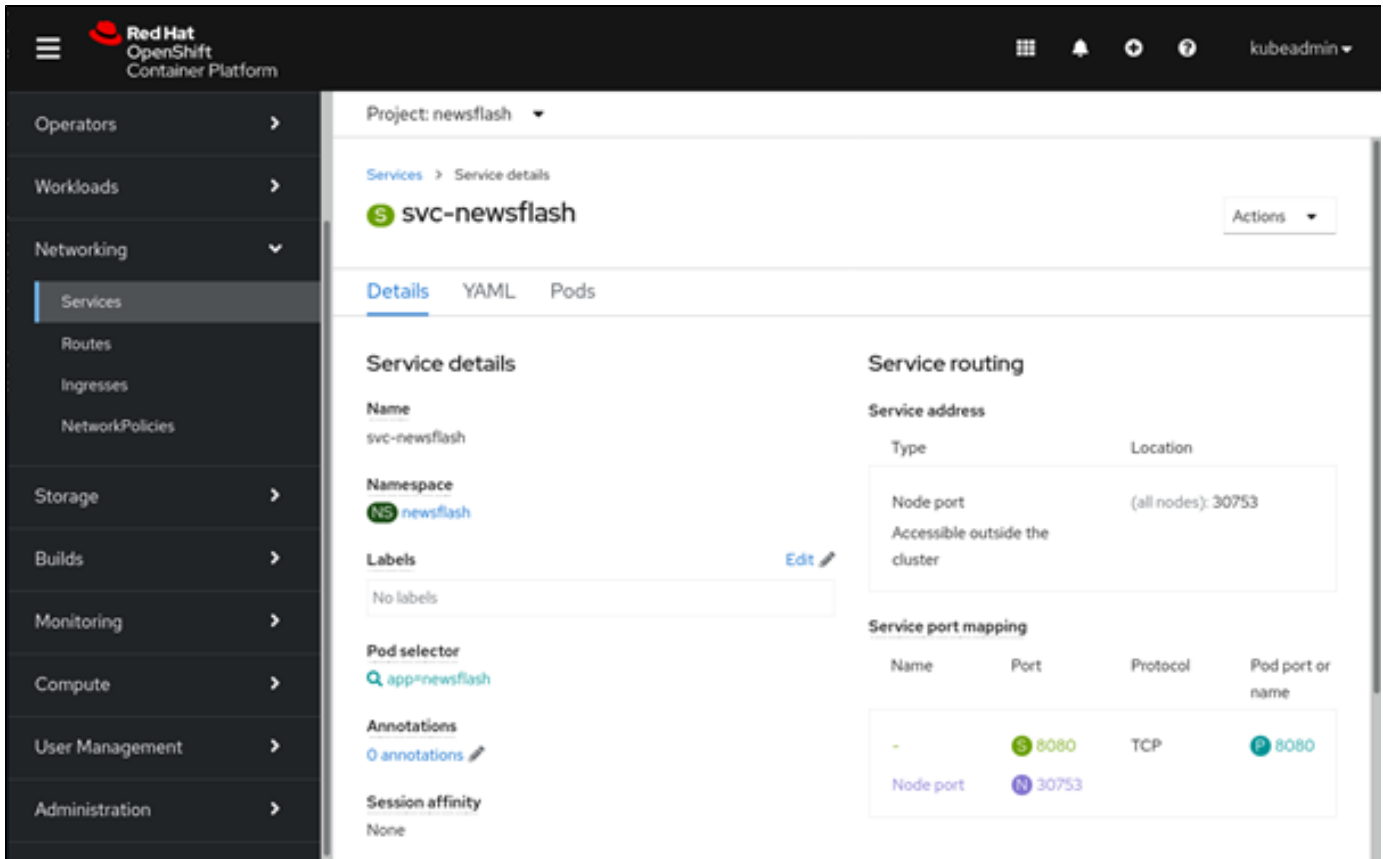
### Create Service

Create by manually entering YAML or JSON definitions, or by dragging and dropping a file into the editor.

```
1  apiVersion: v1
2  kind: Service
3  metadata:
4    name: svc-newsflash
5    namespace: newsflash
6  spec:
7    selector:
8      app: newsflash
9    type: NodePort
10   ports:
11     - protocol: TCP
12       port: 8080
13       targetPort: 8080
14
```

Create Cancel

*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).

## Create DeploymentConfig

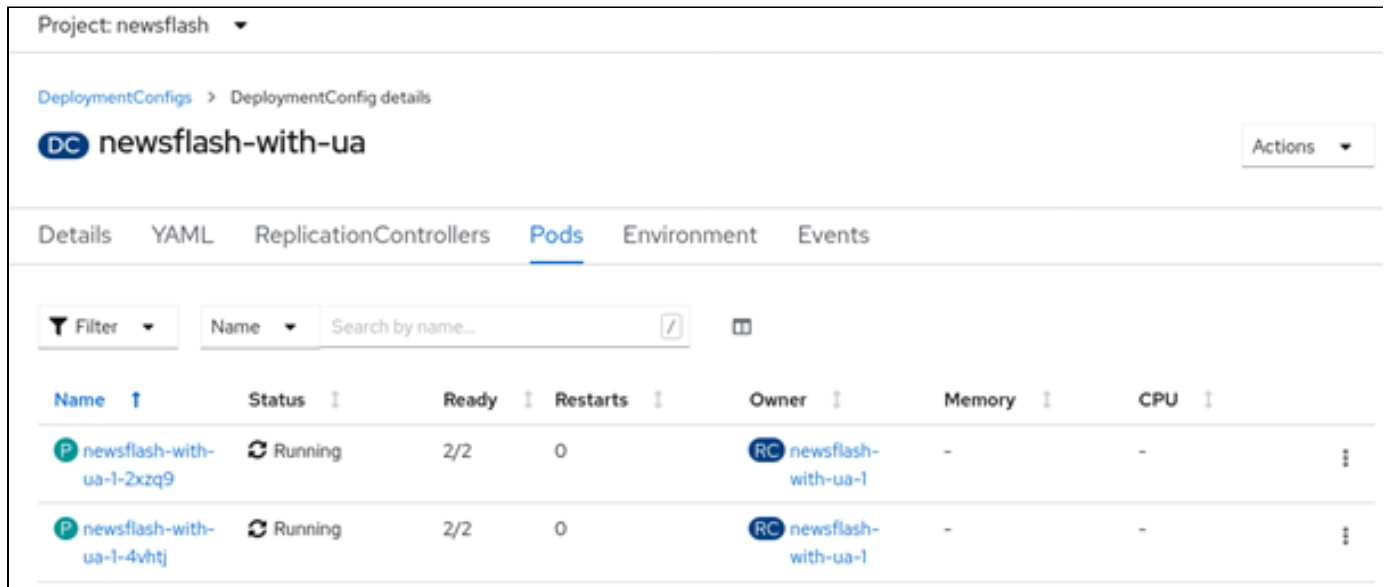
Create by manually entering YAML or JSON definitions, or by dragging and dropping.

```
1  apiVersion: apps.openshift.io/v1
2  kind: DeploymentConfig
3  metadata:
4    name: newsflash-with-ua
5    namespace: newsflash
6  spec:
7    selector:
8      app: newsflash
9    replicas: 2
10   template:
11     metadata:
12       labels:
13         app: newsflash
14     spec:
15       volumes:
16         - name: shared-data
17           emptyDir: {}
18       containers:
19         - name: nginx-container
20           image: bitnami/nginx
21           ports:
22             - containerPort: 8080
23               protocol: TCP
24           volumeMounts:
25             - name: shared-data
26               mountPath: /opt/bitnami/nginx/html
27         # Universal Openshift Agent as sidecar container
28         - name: universal-agent
29           image: 'stonebranch/universal-agent:7.0.0.0'
30           volumeMounts:
31             - name: shared-data
32               mountPath: '/podshare'
33           env:
34             - name: UAGTRANSIENT
35               value: yes
36             - name: UAGAGENTCLUSTERS
37               value: 'AGENT_CLUSTER_APP_NEWSFLASH'
38             - name: UAGOMSSERVERS
39               value: '7878\@192.168.88.40'
```

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).

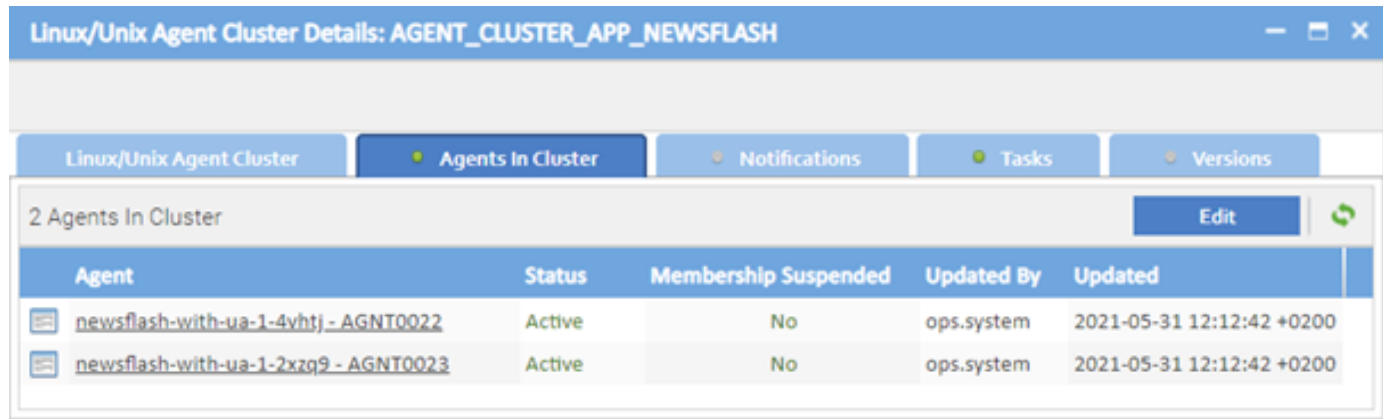
These three Universal Open Shift Agents will connect to the Universal Controller agent cluster, "AGENT\_CLUSTER\_APP\_NEWSFLASH," which was configured in the deployment script as a parameter and during the set-up in Universal Controller.

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

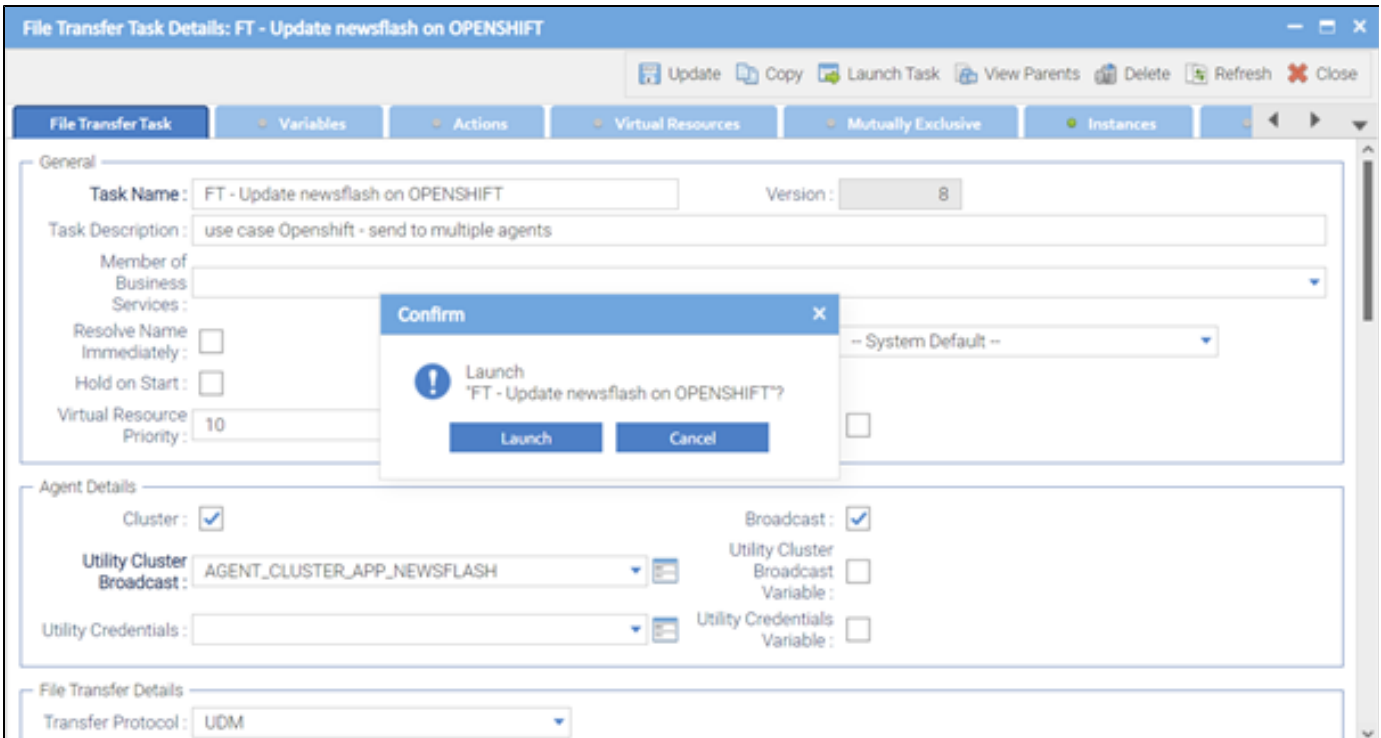
If the number of PODs in Open Shift is scaled up, more agents will automatically register for each new POD. Conversely, if the number of PODs is scaled down, the excess agents will automatically be removed from the Universal Controller agent cluster.

## Running the File Transfer

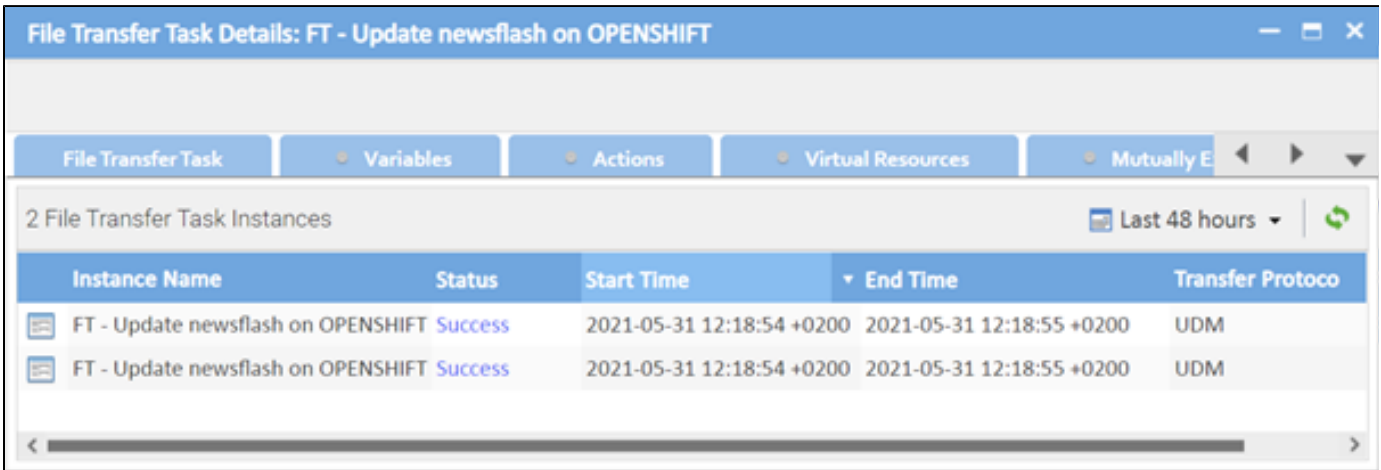
When the Universal Open Shift Agents have registered in the Universal Controller agent cluster, they are ready to send and receive files.

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.



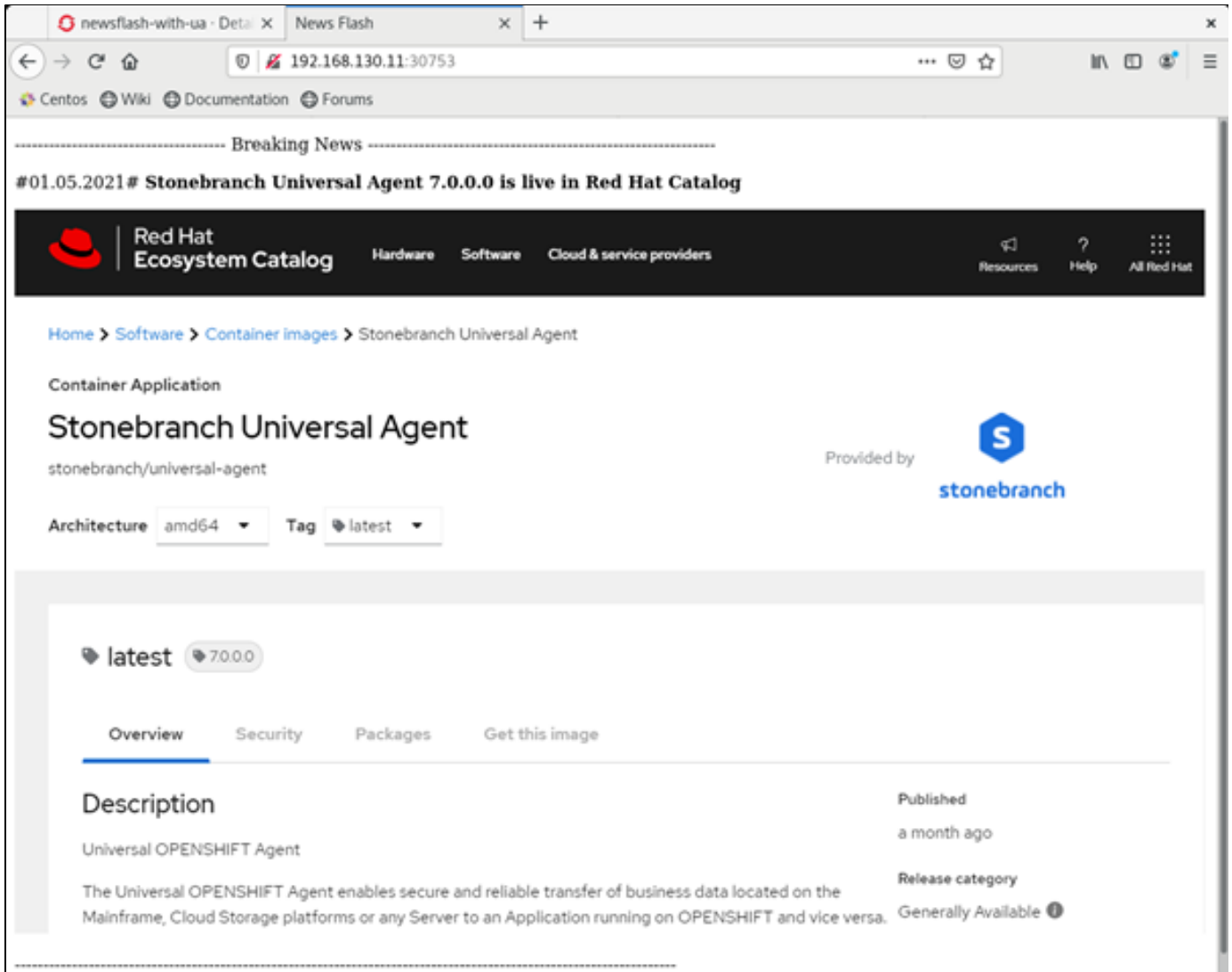
**Universal Controller File Transfer Task**



**File Transfer Task Instances**

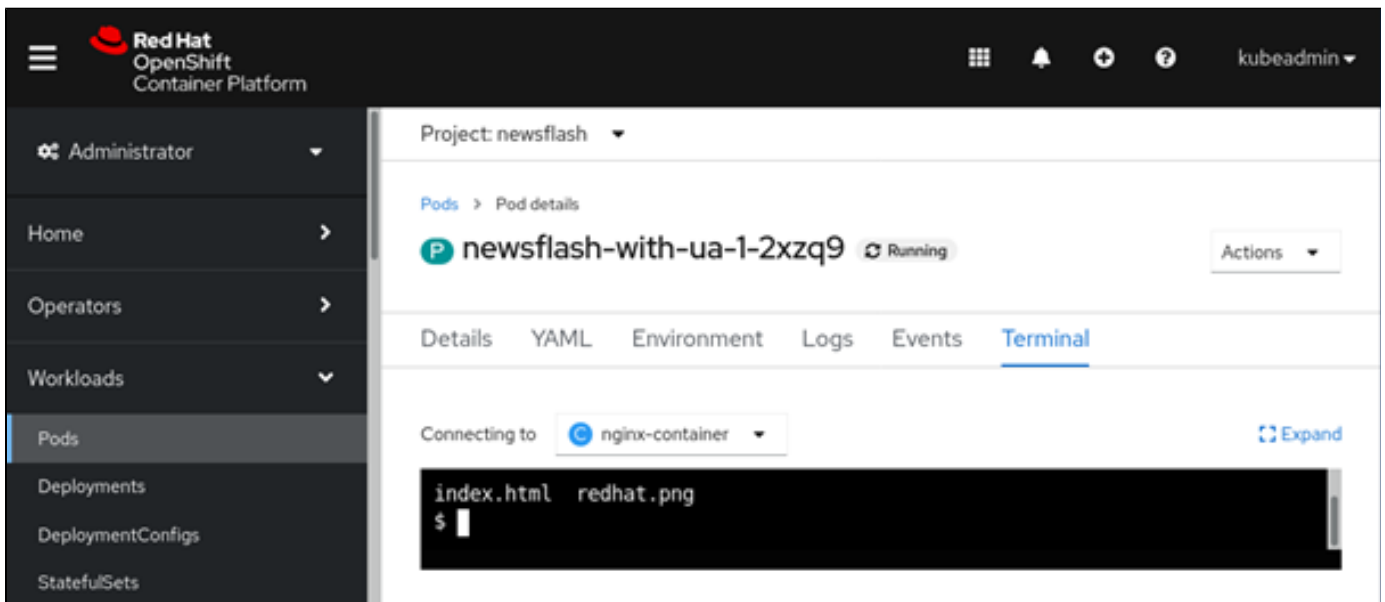
In the screenshot above, you can see that two file transfers were launched. This is because the Universal Controller agent cluster contained two agents, one for each started POD.

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] <a href="https://docs.stonebranch.com/confluence/display/UA70/Docker+Containers">https://docs.stonebranch.com/confluence/display/UA70/Docker+Containers</a>	Open Shift Agent documentation
[2] <a href="https://www.stonebranch.com/products/universal-automation-center/">https://www.stonebranch.com/products/universal-automation-center/</a>	Link to Free Trial for Universal Automation Center
[3] <a href="https://developers.redhat.com/products/codeready-containers/overview">https://developers.redhat.com/products/codeready-containers/overview</a>	RED HAT webpage to retrieve an Open Shift Code Ready Container (CRC)
[4] <a href="https://docs.stonebranch.com/confluence/display/UA70/Universal+Agent+for+UNIX+Installation">https://docs.stonebranch.com/confluence/display/UA70/Universal+Agent+for+UNIX+Installation</a>	Universal Agent Installation
[5] <a href="https://www.stonebranch.com/contact/">https://www.stonebranch.com/contact/</a>	Contact address to get a 30days license key for Universal Data Mover
[6] <a href="https://github.com/stonebranch-marketplace/Universal-Open-Shift-Agent/tree/master/export">https://github.com/stonebranch-marketplace/Universal-Open-Shift-Agent/tree/master/export</a>	Universal Controller File Transfer task export (xml) The XML files are bundled in a tar archive. <ol style="list-style-type: none"><li>1. Extract the tar file to a directory accessible by your Universal Controller ( tar -xvf startup_guide_export.tar ).</li><li>2. Import the extracted xml files using the Universal Controller list import feature.</li><li>3. Adjust the imported file transfer task to your environment.</li></ol>
[7] <a href="https://github.com/stonebranch-marketplace/Universal-Open-Shift-Agent/tree/master/src">https://github.com/stonebranch-marketplace/Universal-Open-Shift-Agent/tree/master/src</a>	<ul style="list-style-type: none"><li>• POD deployment YAML file</li><li>• Service deployment YAML file</li><li>• Webpage index.html and redhat.jpg</li></ul>
[8] <a href="https://www.stonebranch.com/solutions/hybrid-cloud-file-transfers/">https://www.stonebranch.com/solutions/hybrid-cloud-file-transfers/</a>	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](#)
- [Step 1 Prerequisites](#)
- [Step 2 Operator Installation](#)
  - [Operator Installation from Red Hat Marketplace](#)
  - [Operator Installation from within the OpenShift Web Console](#)
- [Step 3 Verification of Operator Installation](#)
- [Step 4 Adjust the Operator YAML \(CSV\)](#)
  - [Universal Controller / OMS Setting](#)
  - [Universal Controller Setting](#)
  - [PersistentVolumeClaim Settings](#)
- [Step 5 Create an Instance](#)
- [Step 6 Verify the Universal Agent POD](#)
  - [Universal Agent POD Log-file](#)
- [Step 7 Verify that the Universal Agent in OpenShift is Connected to Universal Controller](#)
  - [Summary](#)
- [Step 8 Sample Scenario - File Transfer](#)
  - [Configuration Steps](#)
- [Summary](#)

### 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.

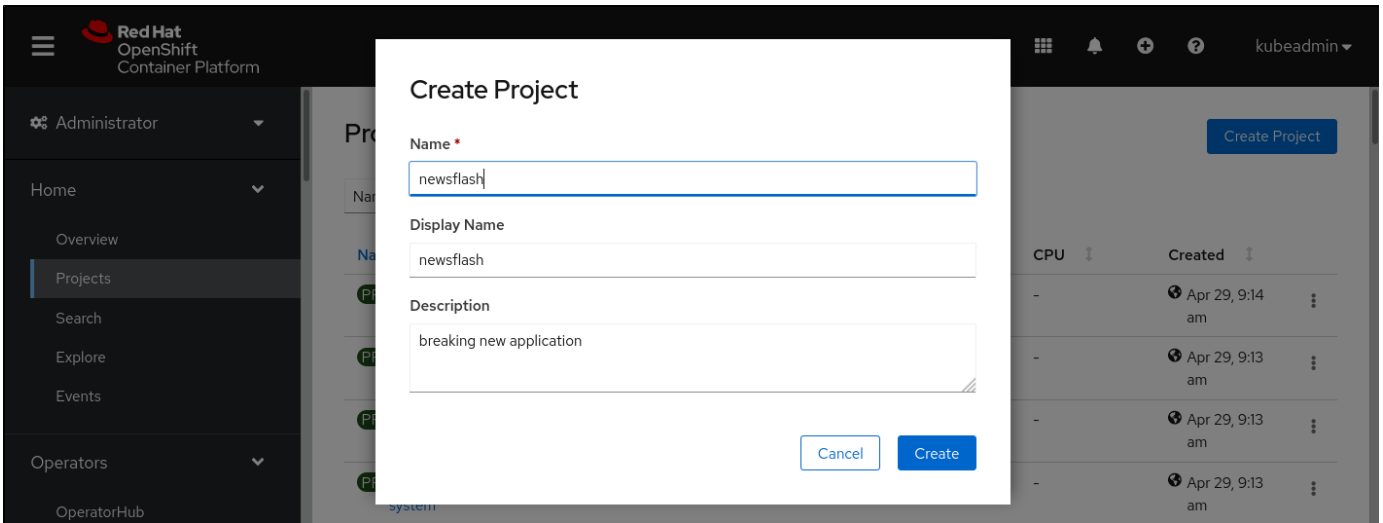
At the end of this tutorial, an example scenario is provided showing how to transfer data from a Linux Server to a shared persistence storage system.

### Step 1 Prerequisites

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

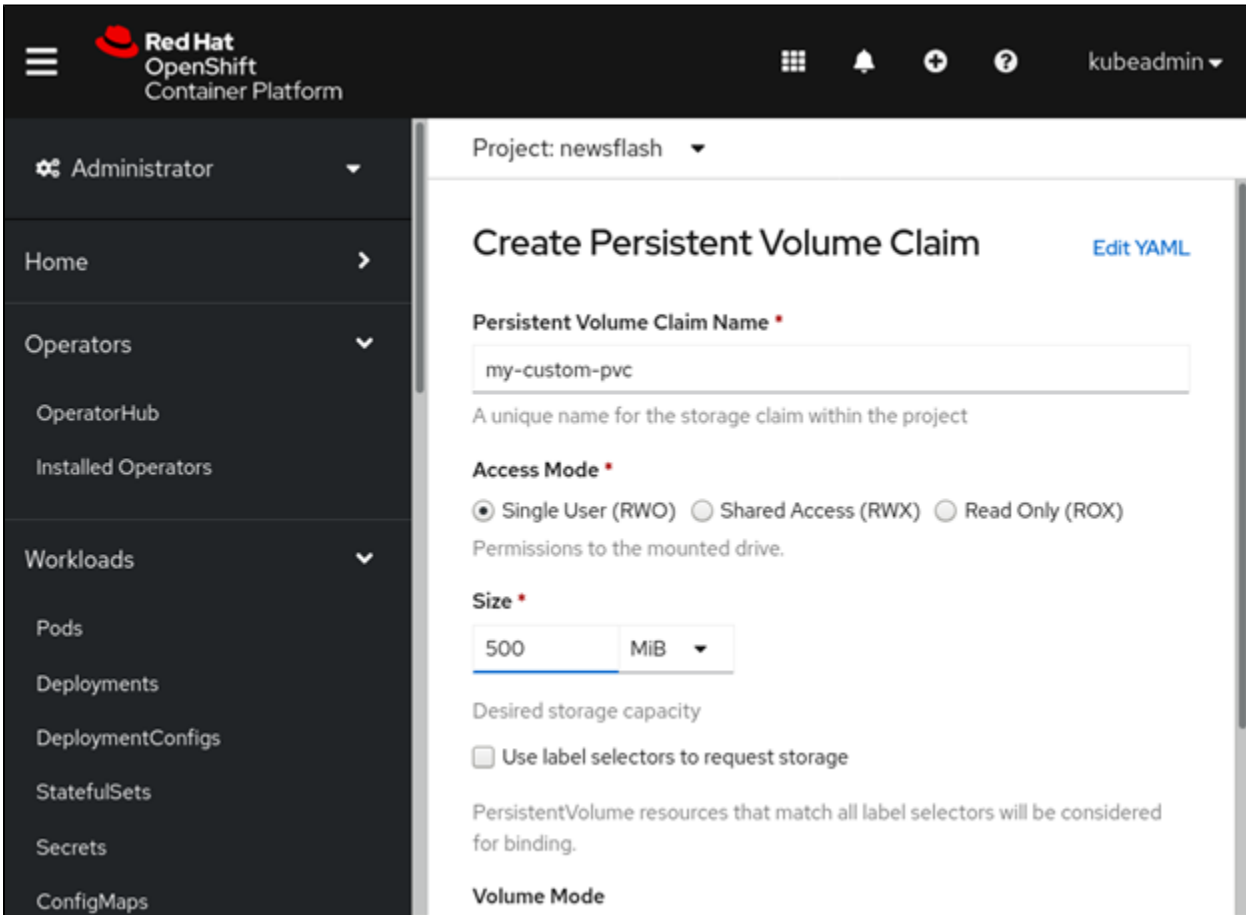
#### 1 Create a Project

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.



Name	Status	Claim	Capacity	Labels	Created
pv0029	Bound	my-custom-pvc	100Gi	volume=pv0029	Apr 30, 9:48 am
pv0001	Bound	crc-image-registry-storage	100Gi	volume=pv0001	Apr 30, 9:47 am
pv0002	Available	No Claim	100Gi	volume=pv0002	Apr 30, 9:47 am
pv0003	Available	No Claim	100Gi	volume=pv0003	Apr 30, 9:47 am
pv0004	Available	No Claim	100Gi	volume=pv0004	Apr 30, 9:47 am
pv0005	Available	No Claim	100Gi	volume=pv0005	Apr 30, 9:47 am

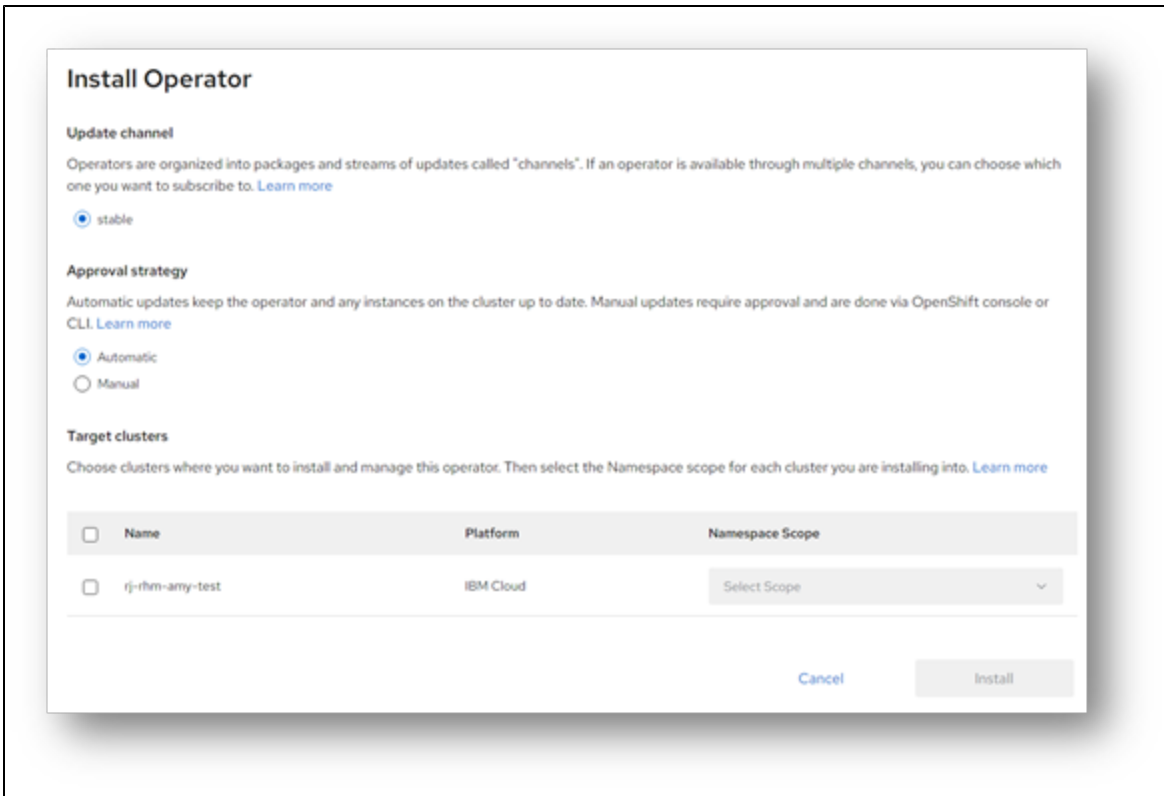
## Step 2 Operator Installation

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**.

The screenshot shows the Red Hat OpenShift OperatorHub interface. The top navigation bar includes the Red Hat logo, 'OpenShift Container Platform', and user information 'kubeadmin'. The left sidebar has a menu with 'Administrator', 'Home', 'Operators', 'OperatorHub' (selected), 'Installed Operators', and 'Workloads'. The main content area is titled 'OperatorHub' and includes a search bar with 'stonebranch' entered. Below the search bar, there are two operator cards for 'Universalagent Operator' provided by Stonebranch. The first card is labeled 'Marketplace' and the second is labeled 'Operator'. Both cards describe the operator as 'Stonebranch Universal Agent IT Automation software'. A list of categories is visible on the left side of the main content area.

## 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](#).

The screenshot shows the Red Hat OpenShift Container Platform interface. The left sidebar contains navigation menus for Administrator, Home, Operators, Workloads, and Installed Operators. The main content area is titled 'OperatorHub > Operator Installation' and 'Install Operator'. It provides instructions on updating operators and offers configuration options:

- Update channel:**  alpha
- Installation mode:**
  - All namespaces on the cluster (default)  
Operator will be available in all Namespaces.
  - A specific namespace on the cluster  
Operator will be available in a single Namespace only.
- Installed Namespace:** NS newsflash
- Approval strategy:**
  - Automatic
  - Manual

On the right, the 'Universalagent Operator' is listed, provided by Stonebranch. Under 'Provided APIs', the 'Universalagent Operator' is shown with the description 'Stonebranch Universal Agent IT Automation software'. At the bottom, there are 'Install' and 'Cancel' buttons.

### 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.

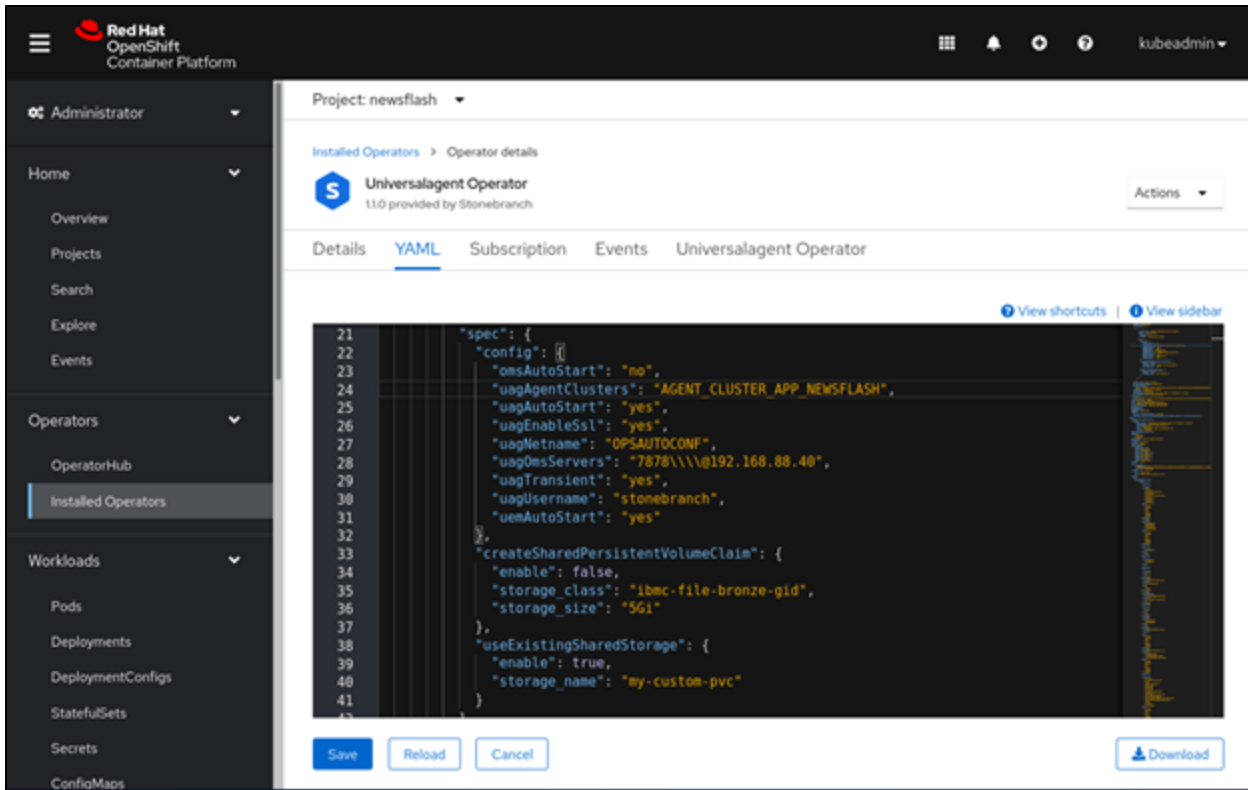
The screenshot shows the Red Hat OpenShift Container Platform administrator interface. The top navigation bar includes the Red Hat logo, the text "Red Hat OpenShift Container Platform", and user information "kubeadmin". The left sidebar contains a menu with categories: Administrator, Home, Operators, and Workloads. The "Operators" section is expanded, showing "OperatorHub" and "Installed Operators". The main content area displays the details for the "Universalagent Operator" (version 1.1.0 provided by Stonebranch). Below this, there is a section titled "Stonebranches" with a "Create Stonebranch" button and a search bar. A table lists the installed stonebranches:

Name	Kind	Status	Labels
stonebranch-sample	Stonebranch	Conditions: Initialized, Deployed	No labels

#### 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 OPENSIFT 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**.

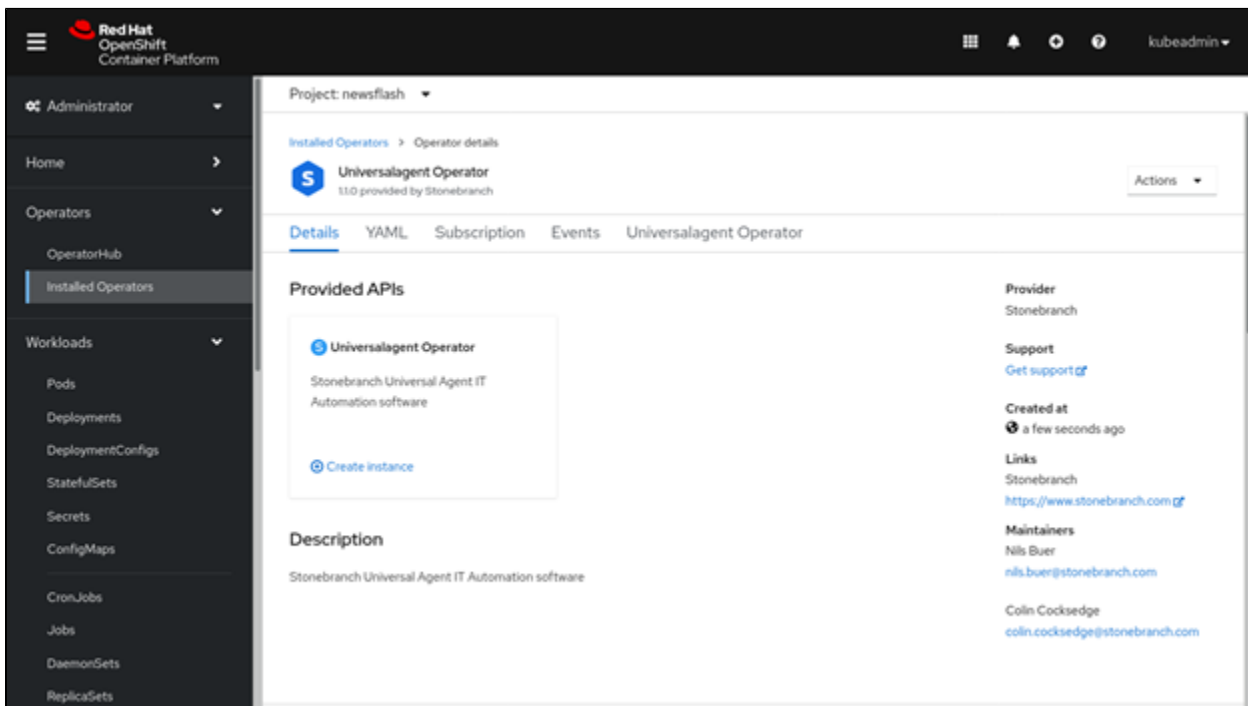
### YAML

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

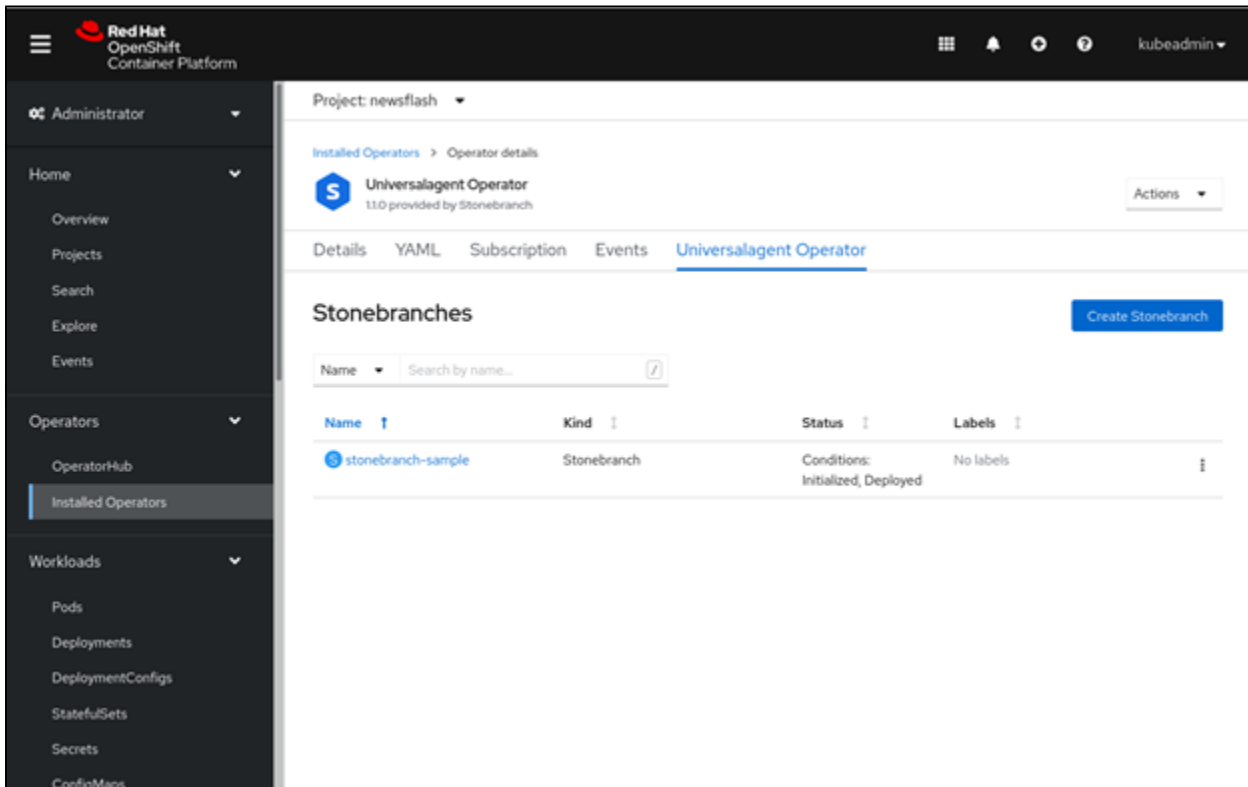
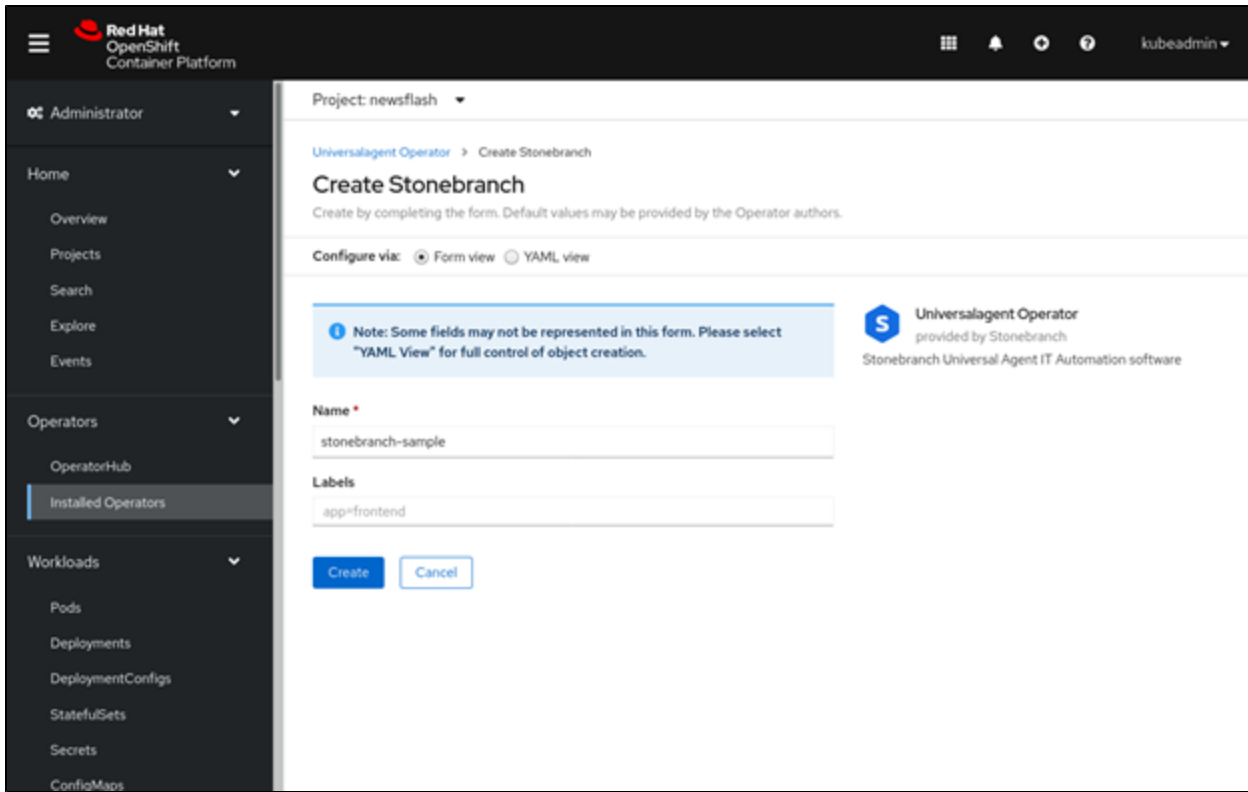
### Step 5 Create an Instance

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**.



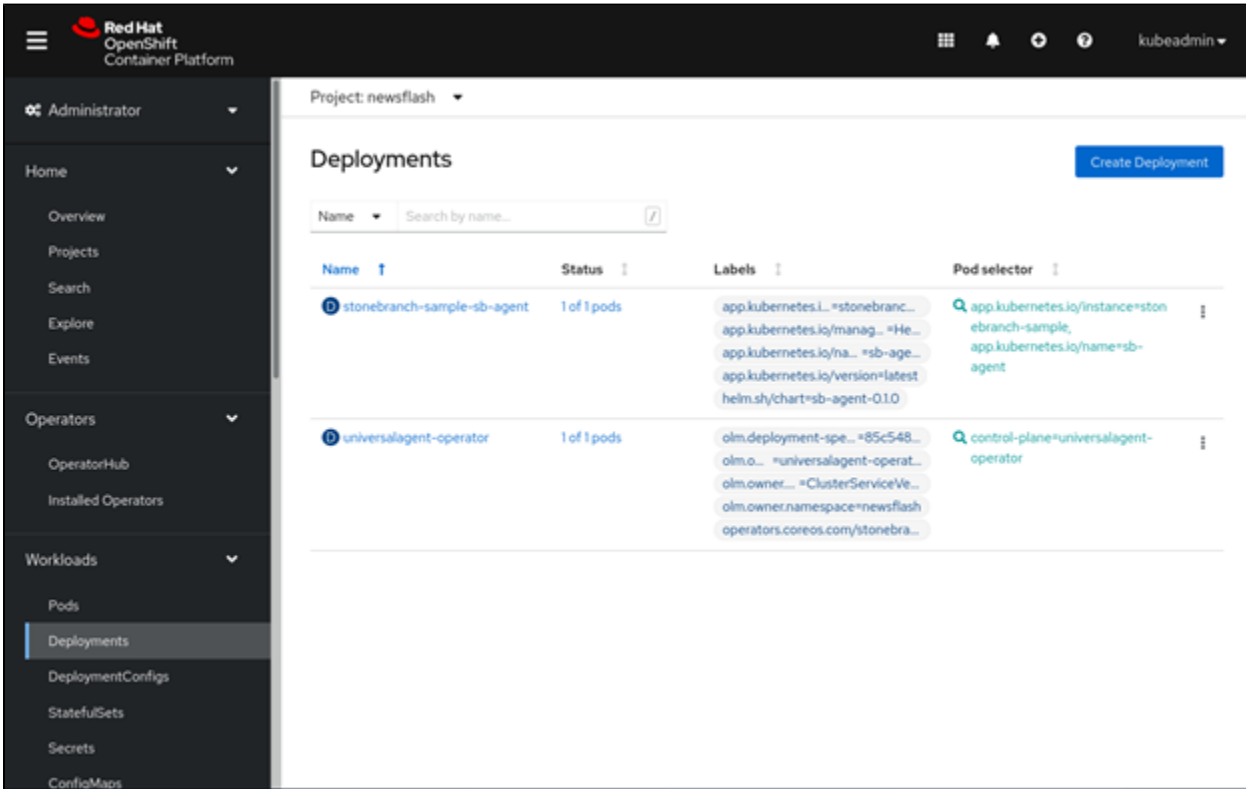
After you click **Create instance**, the **stonebranch-sample-sb-agent** POD in the namespace **newsflash** is created.



Step 6 Verify the Universal Agent POD

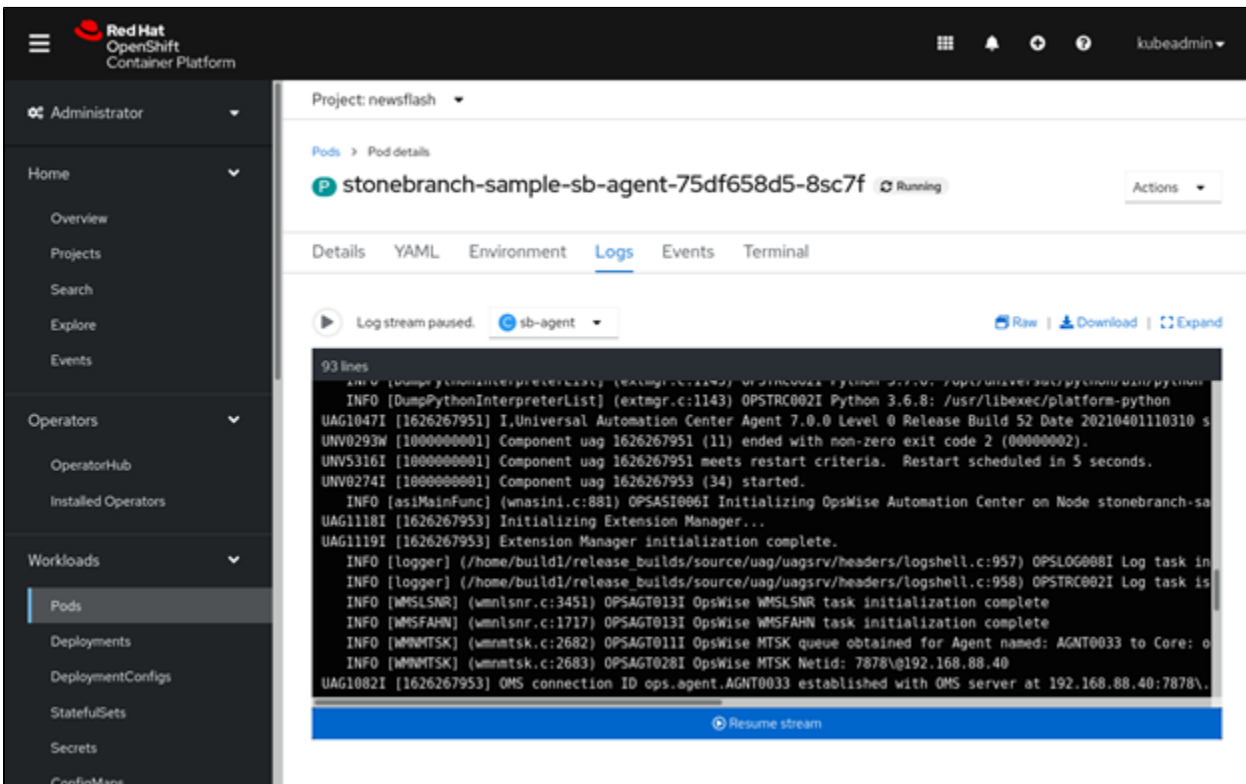
Under **Deployments**, the started **stonebranch-sample-sb-agent** POD in the namespace **newsflash** is shown.

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](https://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.

The screenshot shows the Stonebranch Universal Controller web interface. The top navigation bar includes the Stonebranch logo, the environment name 'PS2 - PROD', the current time '2021-07-14 12:38 +0000', and the user 'Nils Buer'. The main content area is titled 'Linux/Unix Agent Clusters' and displays a table of clusters. Below this, the 'Linux/Unix Agent Cluster Details' section shows 'Agents In Cluster' with one agent listed.

Name	Description	Distribution	Suspended	Task Execution Limit	Limit Amount	Updated
AGENT_CLUSTER_APP_5461d88d5		Any	No	Unlimited		nbuer
AGENT_CLUSTER_APP_NEWSFLASH		Any	No	Unlimited		nbuer
Opswise - Default Linux/Unix Cluster		Any	No	Unlimited		ops.system
SAP-Linux-agent		Any	No	Unlimited		nbuer

Agent	Status	Membership Suspended	Agent Cluster	Last Resolution	Updated By	Updated
stonebranch-sample-sb-agent-75df658d5-fvwfv - AGNT0033	Active	No			ops.system	2021-07-14 12:36:1



#### 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 NGINX Webserver will be deployed in the newsflash project, which was set-up in Step 1.

The NGINX 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 NGINX webserver for further processing.

### Configuration Steps

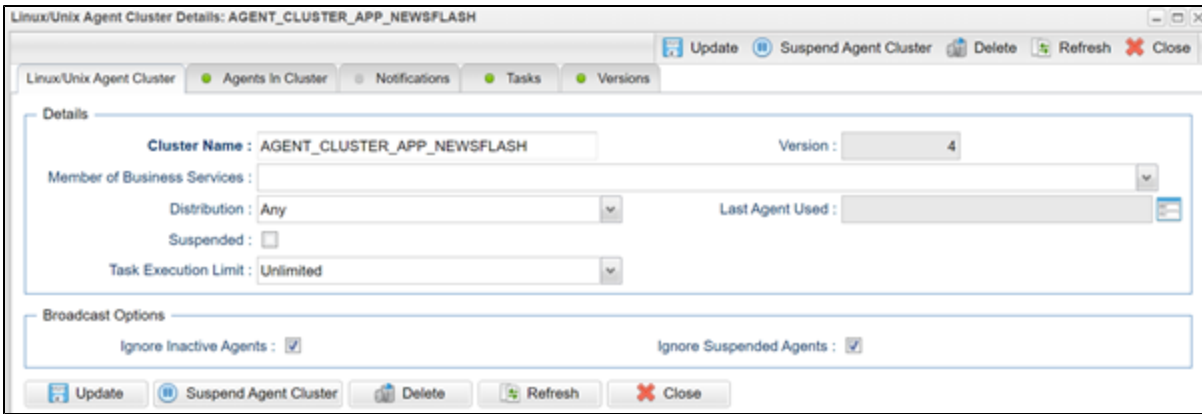
The installation consists of three steps:

1. Define an Agent Cluster in Universal Controller
2. Configure the file transfer workflow using the Universal Controller web GUI.
3. Deploy two sample NGINX webserver 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.

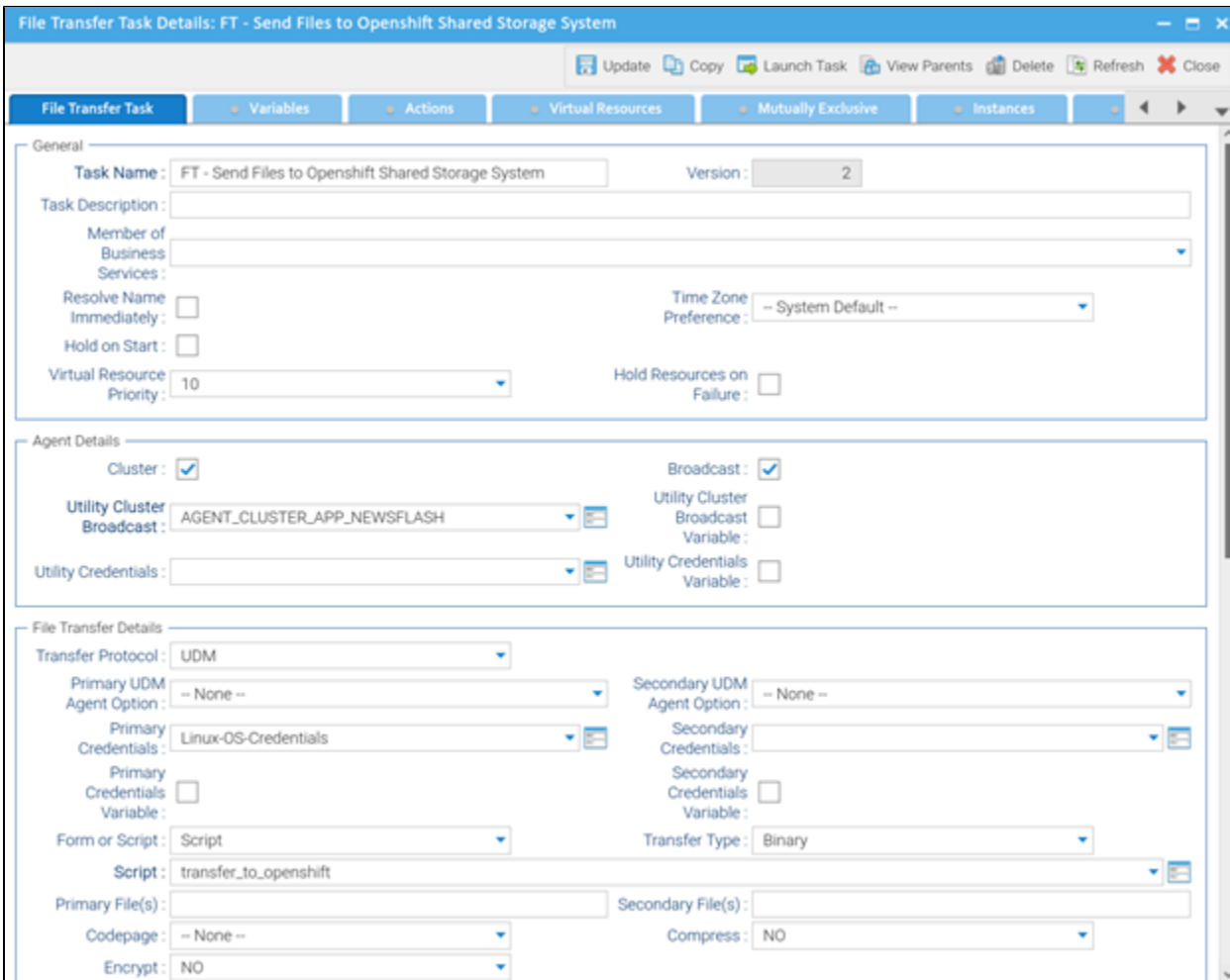
The agent cluster in this example is named *AGENT\_CLUSTER\_APP\_NEWSFLASH*.



## 2 Universal Controller File Transfer Task

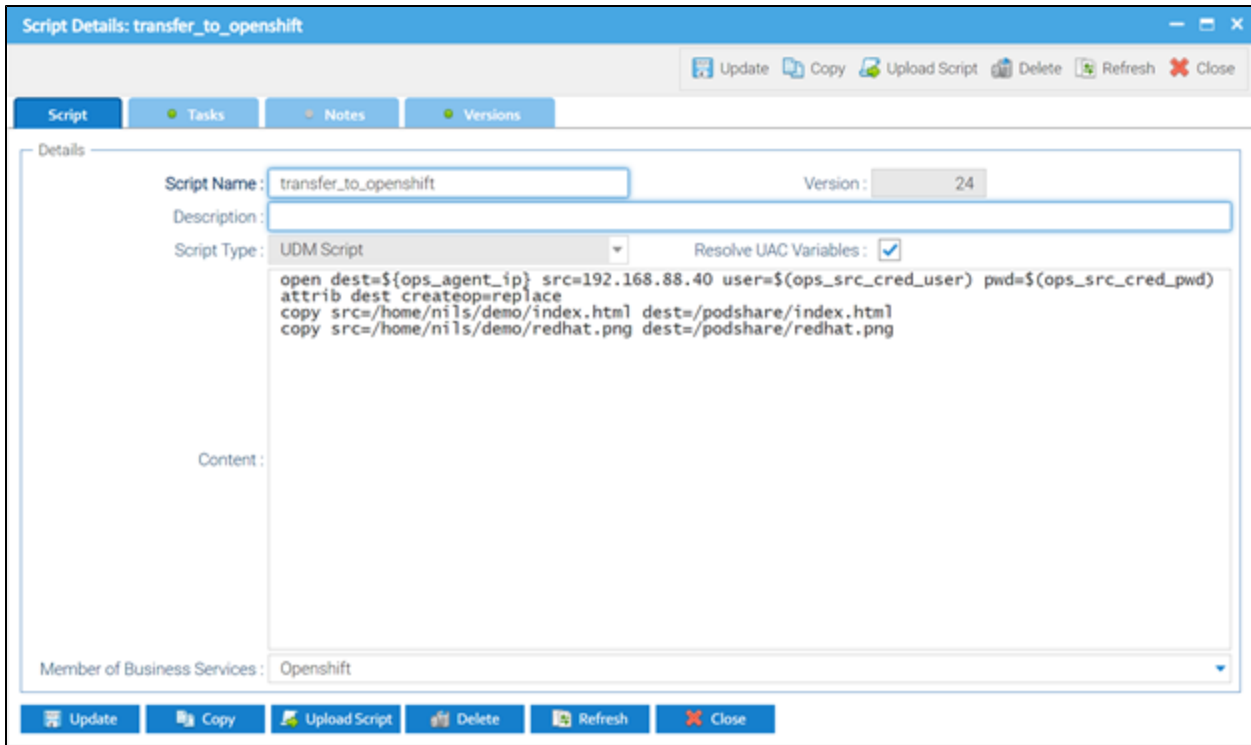
A standard UDM file transfer task can be used to send and retrieve data from the Universal Agent in Openshift. The following File Transfer task transfers two files from an on-premise Linux Server to the OpenShift share folder: **/podshare** (persistence volume claim: my-custom-pvc):

- 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.

File Transfer Task Instance Details: FT - Send Files to Openshift Shared Storage System

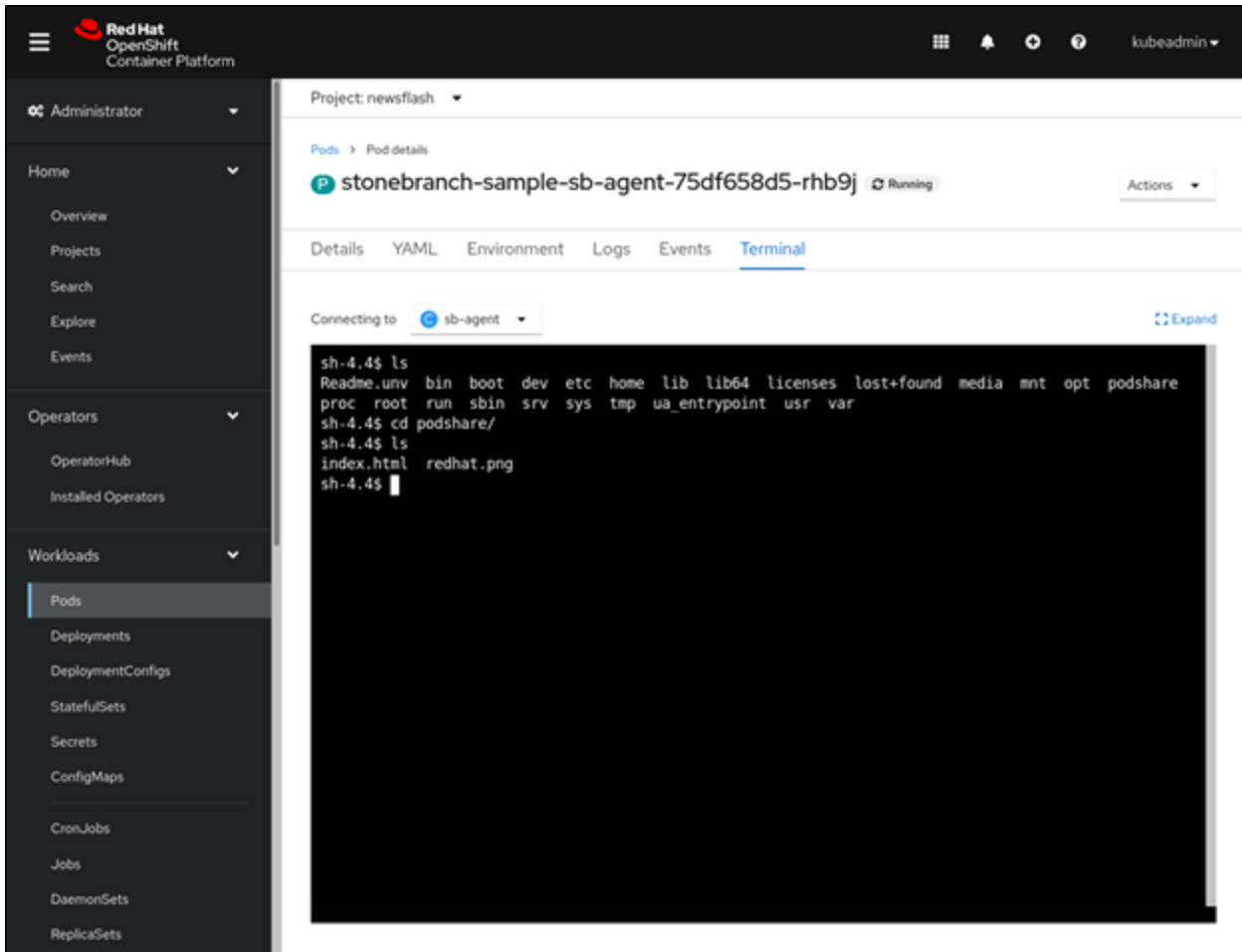
File Transfer Task Instance | Virtual Resources | Exclusive Requests | **Output** | Notes

2 Output

Type	Attempt	Output
		2021.07.14 13.21.59.935 Processing script: /var/opt/universal/tmp/1626247956140075935E200X9070UHH0_udm:
		2021.07.14 13.22.00.652 UNV3992I control: Host=10.217.0.160, Port=49048, NFT=no, Protocol=TLSv1.2, Kx=
		2021.07.14 13.22.00.758 Data session established using cipher: NULL-MD5
		2021.07.14 13.22.00.799 Data session established using compression: None
		2021.07.14 13.22.00.800 Data session established using TT mode: Direct
		2021.07.14 13.22.00.841 Third party session established with dest (component 1626268393) and src (comp
		2021.07.14 13.22.00.849 Transfer mode settings:
		2021.07.14 13.22.00.889 type=binary
		2021.07.14 13.22.00.889 trim=no
		2021.07.14 13.22.00.889 Session options:
		2021.07.14 13.22.00.889 Keep Alive Interval: 120
		2021.07.14 13.22.00.889 Network Fault Tolerant: yes
		2021.07.14 13.22.00.889 NFT ACK Window: 0
STDOUT	1	2021.07.14 13.22.00.989 src: '/home/nils/demo/index.html' is being transferred in binary mode
		2021.07.14 13.22.00.989 src: '/podshare/index.html' will be used as the destination filename
		2021.07.14 13.22.00.989 src: '/home/nils/demo/index.html' transfered successfully in 0:00:00.017.
		2021.07.14 13.22.00.989 src: 476 bytes read 476 bytes written
		2021.07.14 13.22.00.989 src: Transfer operation complete. 1 file(s) copied in 0:00:00.017.
		2021.07.14 13.22.01.009 src: 476 bytes transferred (28000.00 bytes per second)
		2021.07.14 13.22.01.053 src: '/home/nils/demo/redhat.png' is being transferred in binary mode
		2021.07.14 13.22.01.053 src: '/podshare/redhat.png' will be used as the destination filename
		2021.07.14 13.22.01.053 src: '/home/nils/demo/redhat.png' transfered successfully in 0:00:00.025.
		2021.07.14 13.22.01.053 src: 62823 bytes read 62823 bytes written
		2021.07.14 13.22.01.053 src: Transfer operation complete. 1 file(s) copied in 0:00:00.025.
		2021.07.14 13.22.01.080 src: 62823 bytes transferred (2512920.00 bytes per second)
		2021.07.14 13.22.01.084 Finished processing script: /var/opt/universal/tmp/1626247956140075935E200X9070
		2021.07.14 13.22.01.138 Session closed
STDERR	1	2021.07.14 13.21.59.934 UNV2800I Universal Data Mover 7.0.0 Level 0 Release Build 44 started.
		2021.07.14 13.22.01.139 UNV2801I Universal Data Mover 7.0.0 Level 0 Release Build 44 ended successfull

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**

Red Hat OpenShift Container Platform

Project: newsflash

### Create DeploymentConfig

Create by manually entering YAML or JSON definitions, or by dragging and dropping a file into the editor.

```
1  apiVersion: apps.openshift.io/v1
2  kind: DeploymentConfig
3  metadata:
4    name: newsflash-with-ua
5    namespace: newsflash
6  spec:
7    selector:
8      app: newsflash
9    replicas: 2
10   template:
11     metadata:
12       labels:
13         app: newsflash
14     spec:
15       volumes:
16         - name: shared-data
17           persistentVolumeClaim:
18             claimName: my-custom-pvc
19       containers:
20         - name: nginx-container
21           image: bitnami/nginx
22           ports:
23             - containerPort: 8080
24               protocol: TCP
25           volumeMounts:
26             - name: shared-data
27               mountPath: /opt/bitnami/nginx/html
```

[View shortcuts](#) | [View sidebar](#)

[Create](#) [Cancel](#) [Download](#)

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

Project: newsflash

DeploymentConfigs > DeploymentConfig details

**DC newsflash-with-ua** Actions

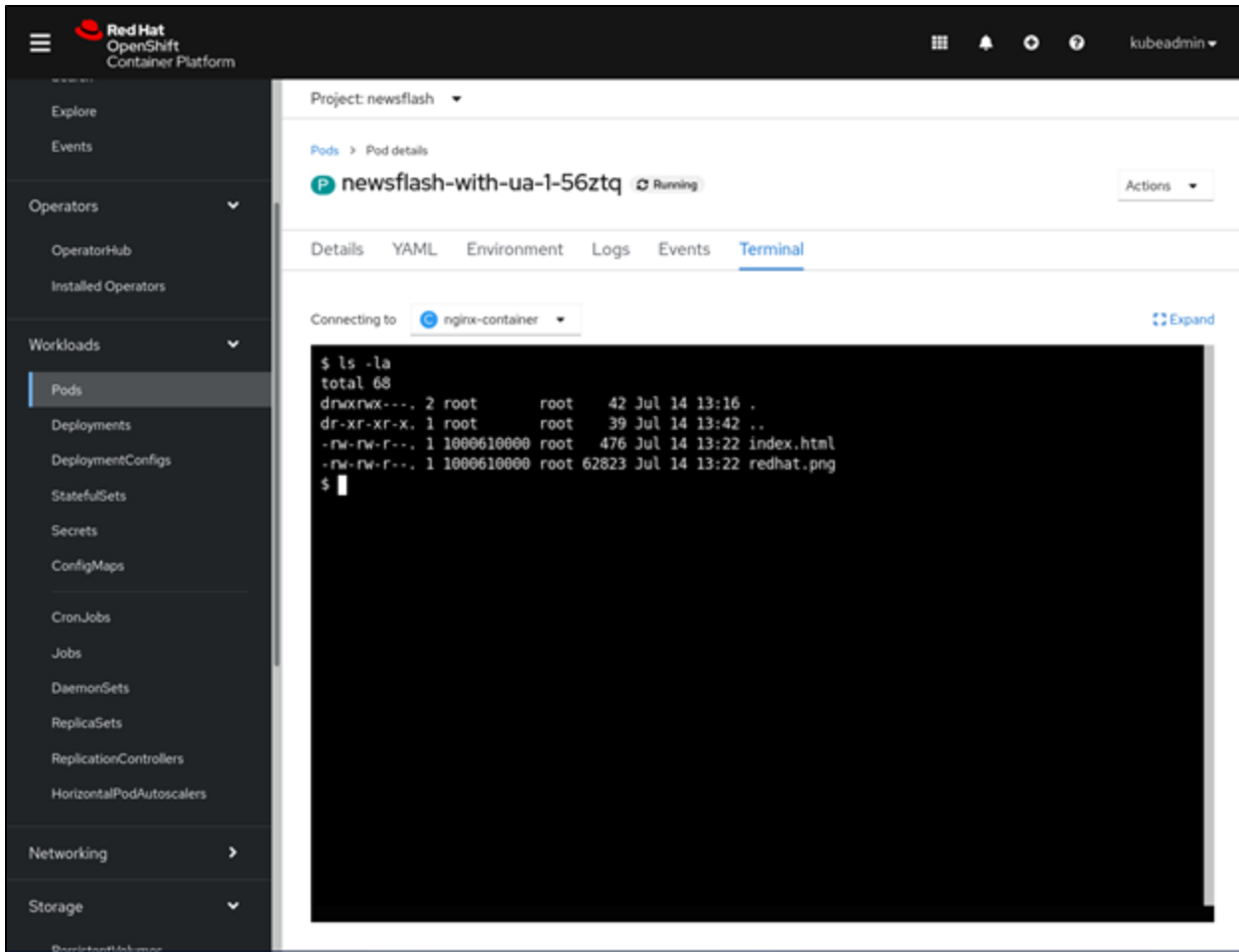
Details | YAML | ReplicationControllers | Pods | Environment | Events

### DeploymentConfig details

2 Pods

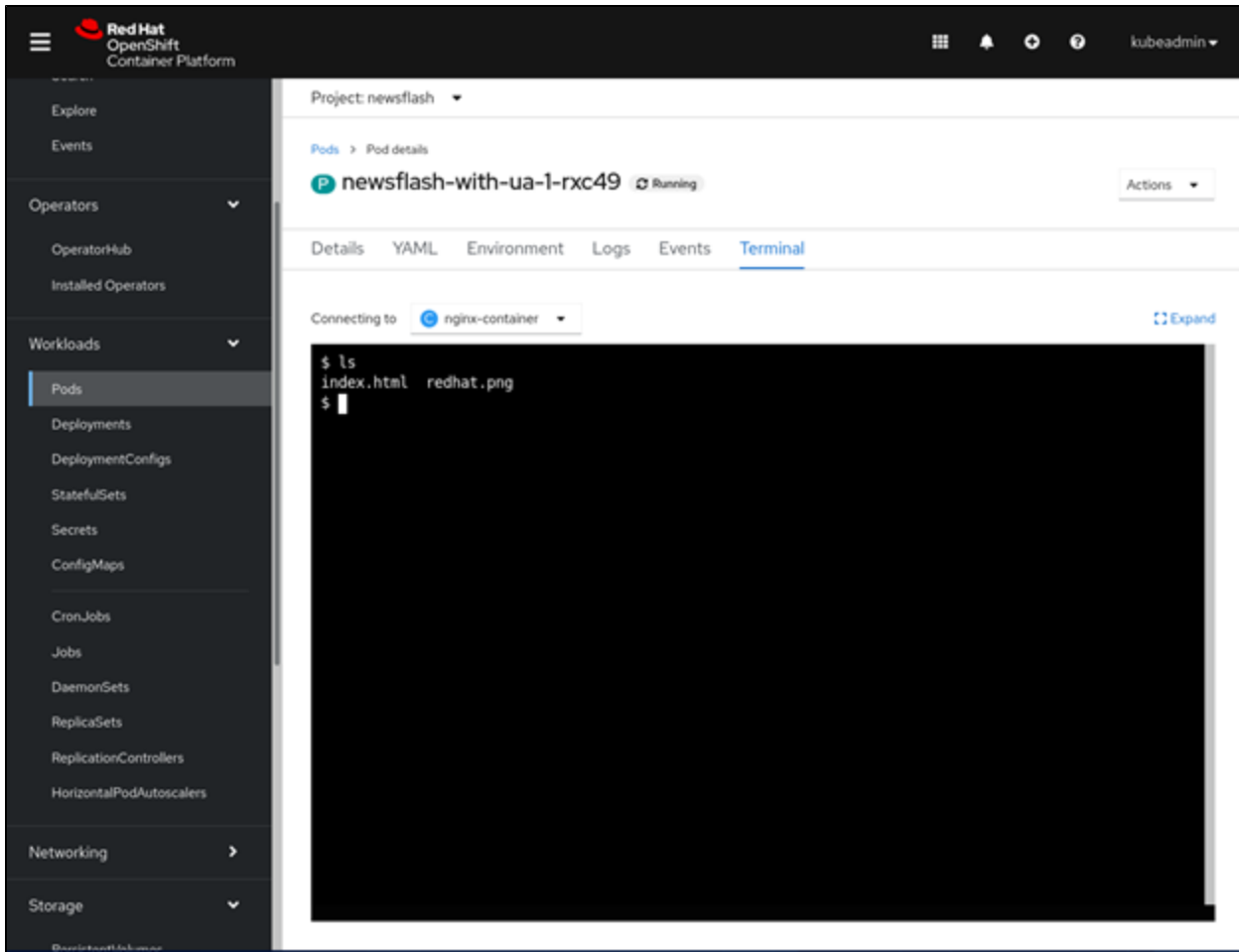
<b>Name</b> newsflash-with-ua	<b>Latest version</b> 1
<b>Namespace</b> NS newsflash	<b>Message</b> config change
<b>Labels</b> No labels <a href="#">Edit</a>	<b>Update strategy</b> Rolling
<b>Pod selector</b> Q app=newsflash	<b>Max unavailable</b> 25% of 2 pods
<b>Node selector</b> No selector	<b>Max surge</b> 25% greater than 2 pods
<b>Tolerations</b> 0 tolerations <a href="#">Edit</a>	<b>Min ready seconds</b> Not configured
<b>Annotations</b> 0 annotations <a href="#">Edit</a>	<b>Triggers</b> ConfigChange
<b>Status</b>	

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 OPENSIFT 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 OPENSIFT cluster (and vice versa).