

# LAB EXERCISE: RedHat OpenShift with Contrail 5.0

## Author

Savithru Lokanath <slokanath@juniper.net>, v1.4

## Prerequisites

- An AWS account with permissions to CRUD Compute, Storage & Network resources
- CentOS 7 subscription

## Installation

Follow [this guide](#) to bring up the cluster on AWS

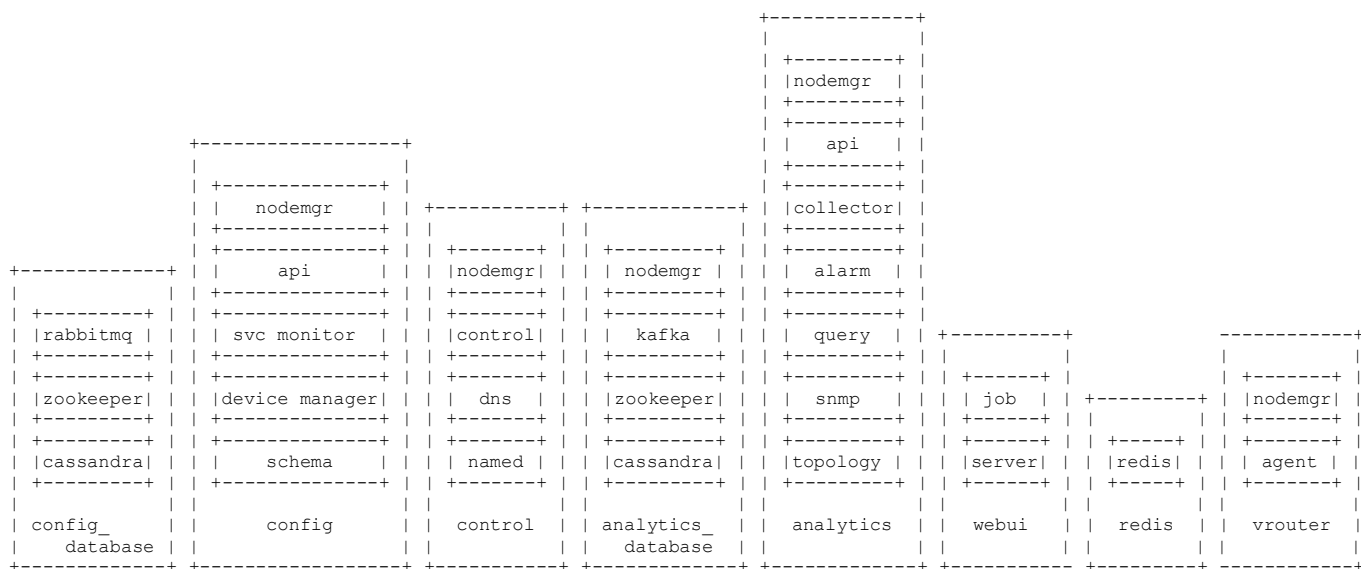
**NOTE:** This installer provisions 1 Master (Control) & 1 Minion (Compute)

## Architecture

Starting with release 5.0, all Contrail services are containerized & are orchestrated by OpenShift Container Platform.

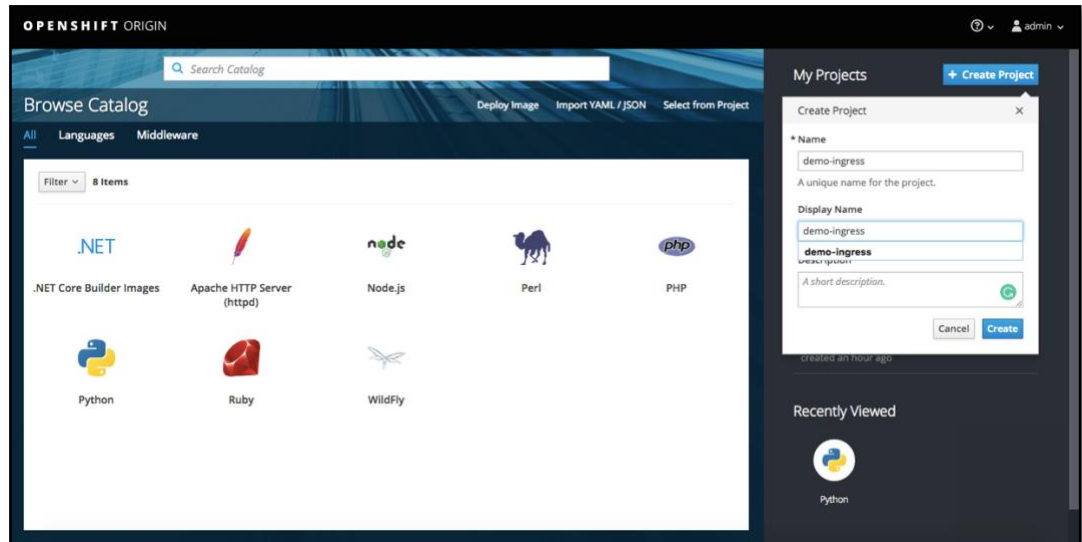
OpenShift Container Platform leverages the Kubernetes concept of a *pod*, which is one or more [containers](#) deployed together on one host, and the smallest compute unit that can be defined, deployed, and managed.

The inner structure of Contrail *pods* is as below



## Use Case - 2

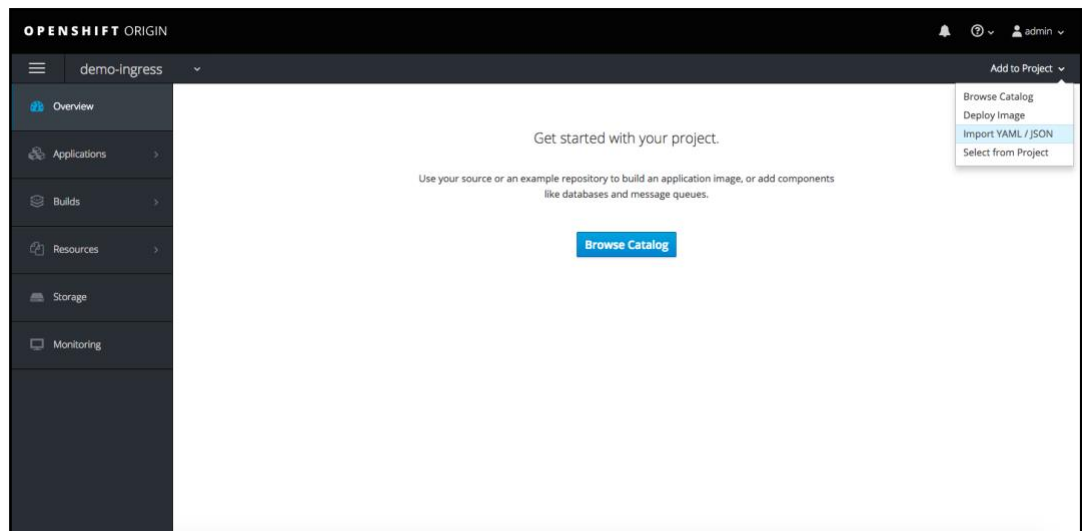
- Expose highly scalable applications using OpenShift
- Login** to the OpenShift dashboard & **create** a new project



- Launch** the pods in the new project you just created

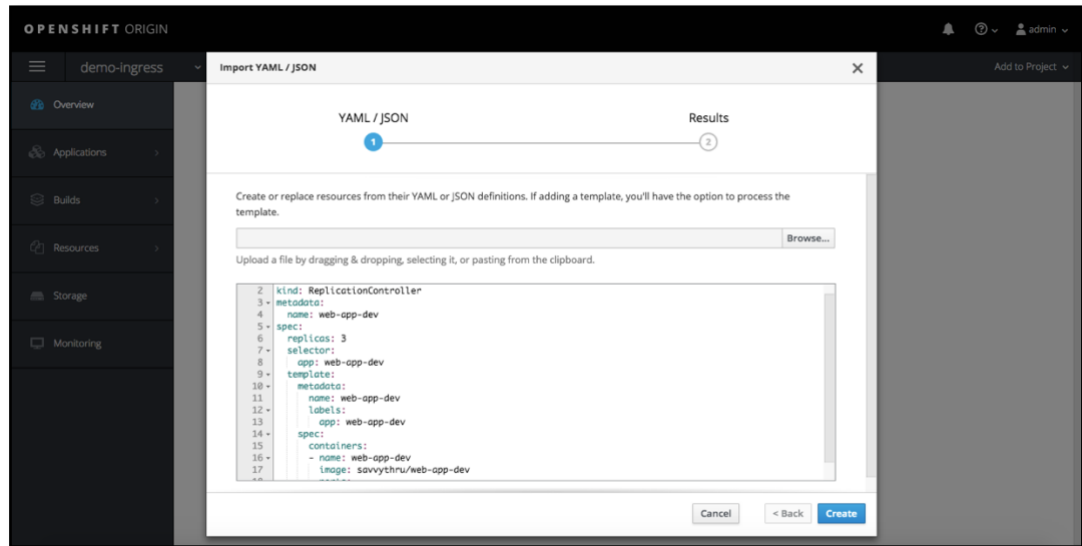
*OpenShift Dashboard > Add to Project > Import YAML/JSON*

**NOTE:** All pod specs are located [here](#). Copy paste the pod YAMLs into the OpenShift window & hit **Create**

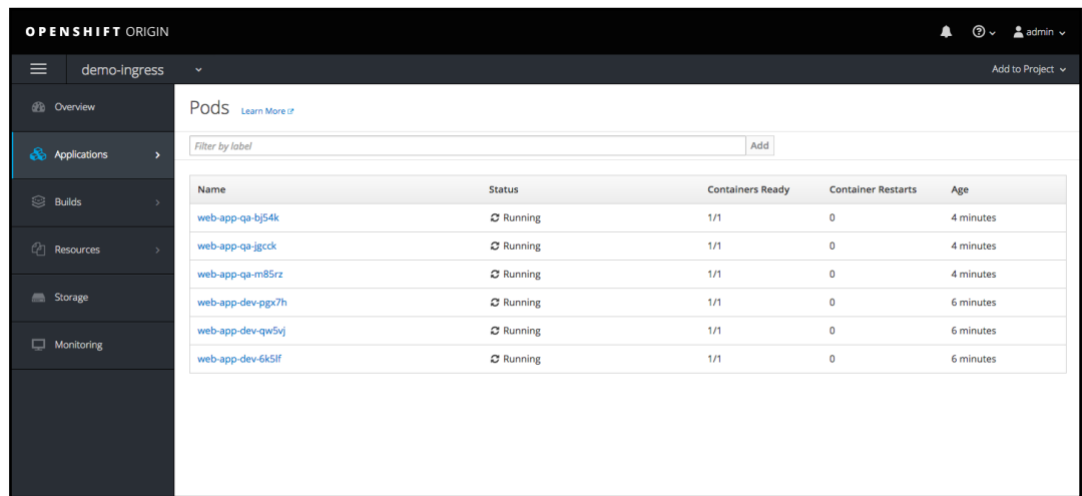


Create two different pods: “web-app-dev-pod” & “web-app-qa-pod”

This will create 3 replicas of each pod, so a total of 6.



Navigate to *OpenShift Dashboard > Applications > Pods* & verify pods came up fine



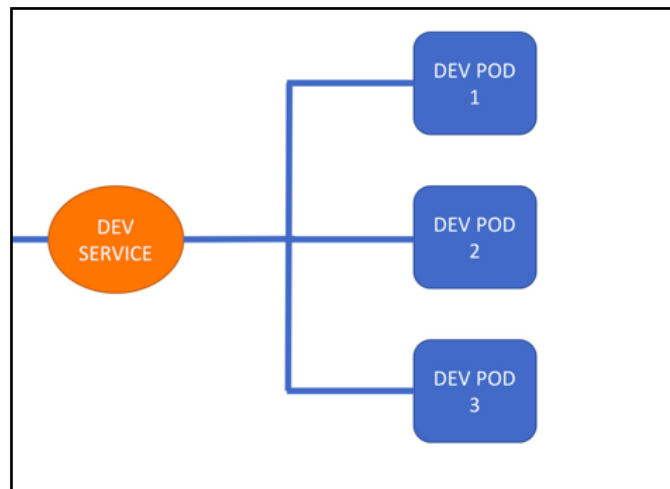
3. Similarly, **launch** the services in the new project you just created

*OpenShift Dashboard > Add to Project > Import YAML/JSON*

**NOTE:** Services, by definition are logical entities that load balance (L4) the input requests to the backend pods

**NOTE:** All service specs are located [here](#). Copy paste the service YAMLs into the OpenShift window & hit **Create**

Create two different services: “web-app-dev-svc” & “web-app-qa-svc”



Navigate to *OpenShift Dashboard > Applications > Services* & verify services came up fine

OPENSIFT ORIGIN

demo-ingress

Services [Learn More](#)

Filter by label  Add

| Name                        | Cluster IP    | External IP | Ports  | Selector        | Age       |
|-----------------------------|---------------|-------------|--------|-----------------|-----------|
| <a href="#">web-app-qa</a>  | 10.99.237.244 | none        | 80/TCP | app=web-app-qa  | 4 minutes |
| <a href="#">web-app-dev</a> | 10.110.233.85 | none        | 80/TCP | app=web-app-dev | 4 minutes |

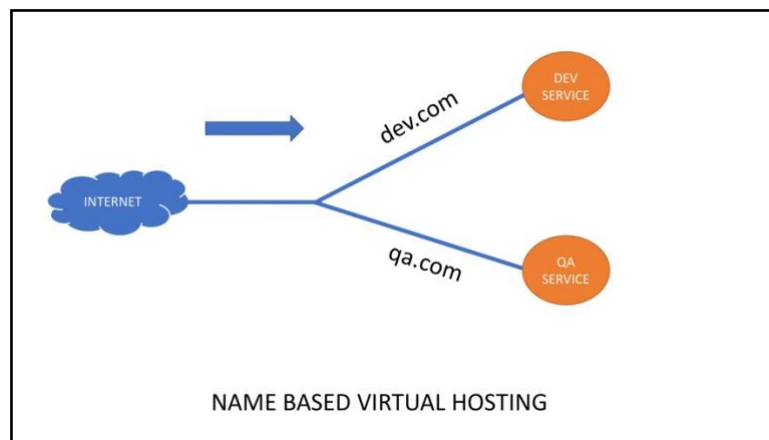
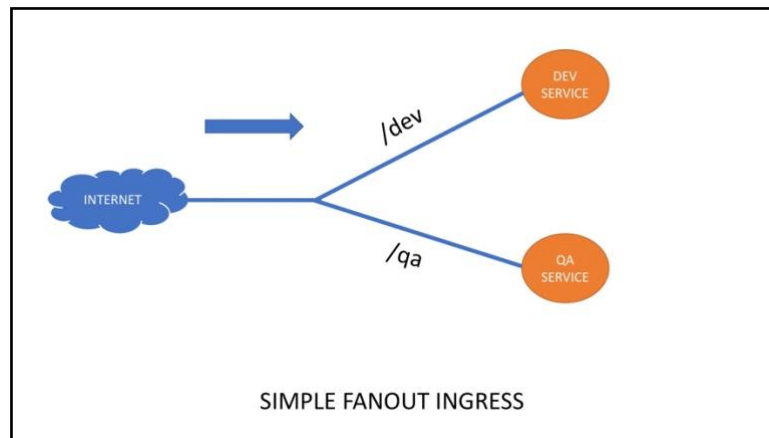
4. Similarly, **launch** the ingress controllers in the new project you just created

*OpenShift Dashboard > Add to Project > Import YAML/JSON*

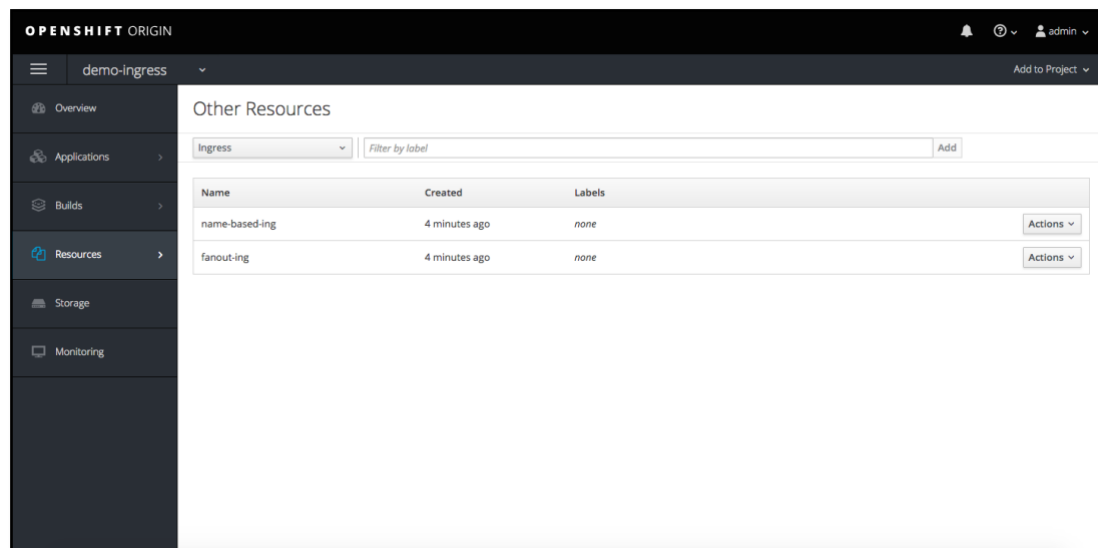
**NOTE:** Ingress, by definition are logical entities that load balance (L7) the input requests to the backend services

**NOTE:** All ingress specs are located [here](#). Copy paste the ingress YAMLs into the OpenShift window & hit **Create**

Create two different ingress controllers: **“fanout-ingress”** & **“name-based-ingress”**



Navigate to *OpenShift Dashboard* > *Resources* > *Other Resources* & verify ingress controllers came up fine



5. Once the ingress controllers are setup, **verify** that you can access the application using the ingress IP address

Get the ingress's IP address from the Contrail dashboard (**10.47.255.X**)

*Contrail Dashboard > Configure > Networking > Load Balancing (demo-ingress namespace)*

| Name                        | Description | Subnet              | Fixed IPs     | Floating IPs | Listener | Operating Status | Admin State |
|-----------------------------|-------------|---------------------|---------------|--------------|----------|------------------|-------------|
| demo-ingress_web-a-pp-dev   | -           | undefined/undefined | 10.110.233.85 | -            | 1        | Online           | Yes         |
| demo-ingress_web-a-pp-qa    | -           | undefined/undefined | 10.99.237.244 | -            | 1        | Online           | Yes         |
| demo-ingress_fanout-ing     | -           | undefined/undefined | 10.47.255.242 | -            | 2        | Online           | Yes         |
| demo-ingress_name-based-ing | -           | undefined/undefined | 10.47.255.241 | -            | 2        | Online           | Yes         |

Get the minion's public DNS from AWS dashboard. With this, we will create an SSH tunnel to localhost

**(localhost)#** `ssh -L localhost:10101:<fanout-ingress-IP-address>:80 root@<minion-public-DNS>`

Once done, the application should be available at **localhost:10101**

| Name              | Instance ID          | Instance Type | Availability Zone | Instance State | Status Checks  | Alarm Status | Public DNS (IPv4)   |
|-------------------|----------------------|---------------|-------------------|----------------|----------------|--------------|---------------------|
| master-demo-stack | i-0ad373675a1e230... | t2.2xlarge    | us-west-1a        | running        | 2/2 checks ... | None         | ec2-13-56-80-84...  |
| minion-demo-stack | i-0a9022f35ce0eb8de  | t2.2xlarge    | us-west-1a        | running        | 2/2 checks ... | None         | ec2-13-56-241-97... |

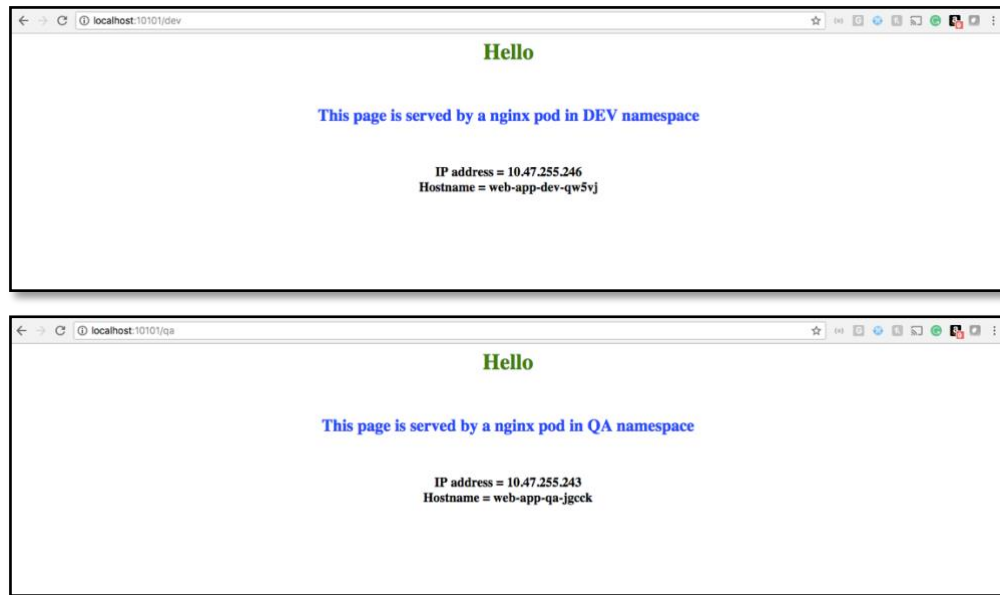
  

| Instance: i-0a9022f35ce0eb8de (minion-demo-stack) |                     | Public DNS: ec2-13-56-241-97.us-west-1.compute.amazonaws.com |                                                  |
|---------------------------------------------------|---------------------|--------------------------------------------------------------|--------------------------------------------------|
| Instance ID                                       | i-0a9022f35ce0eb8de | Public DNS (IPv4)                                            | ec2-13-56-241-97.us-west-1.compute.amazonaws.com |
| Instance state                                    | running             | IPv4 Public IP                                               | 13.56.241.97                                     |
| Instance type                                     | t2.2xlarge          | IPv6 IPs                                                     | -                                                |
| Elastic IPs                                       | -                   | Private DNS                                                  | ip-10-10-10-11.us-west-1.compute.internal        |

**NOTE:** The fanout/name-based ingress controllers are URL based HAProxy loadbalancers

Navigate to **localhost:10101/dev** to be served by “**web-app-dev**” services/pods

Navigate to **localhost:10101/qa** to be served by “**web-app-qa**” services/pods



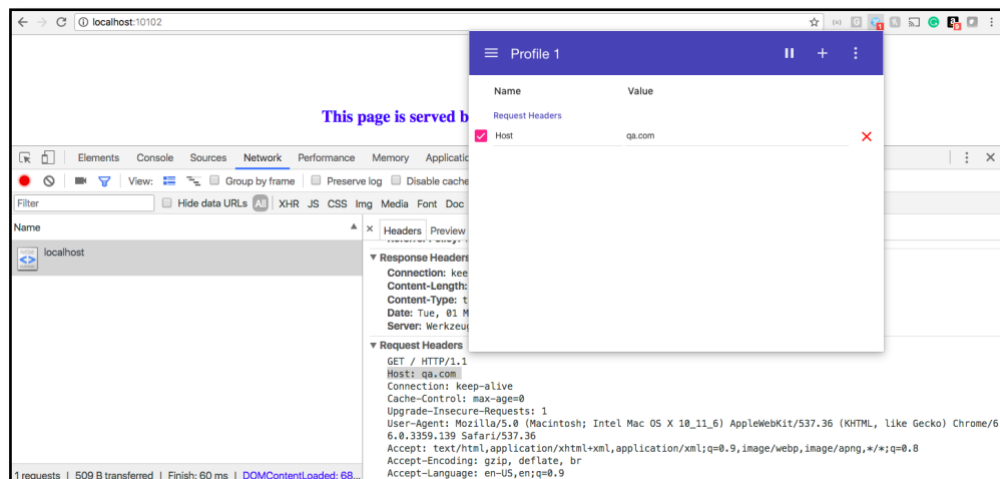
For **name-based** ingress type, pass the configured hostname in the HTTP request

**NOTE:** You will need to install a plugin for Chrome which allows you to modify the HTTP header  
[Add-to-Chrome](#)

**(localhost)#** ssh -L localhost:10102:<name-based-ingress-IP-address>:80 root@<minion-public-DNS>

Once done, the application should be available at **localhost:10102**

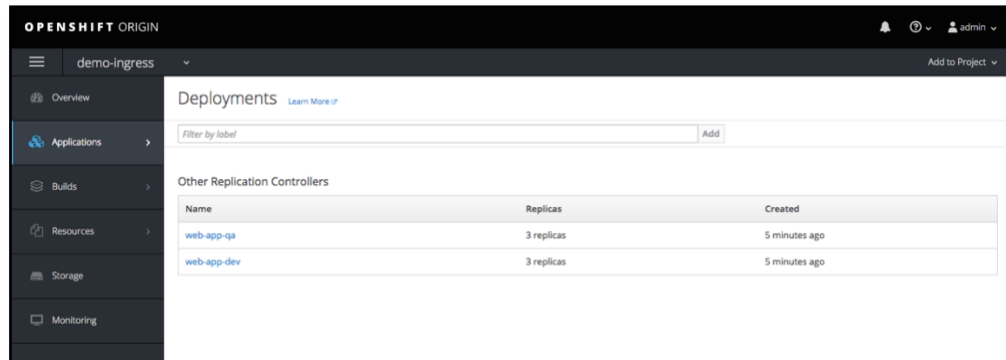
Pass the host info into the HTTP header



Similarly, pass **host=qa.com** in the HTTP header & verify that the application is served by QA service

## 6. Scale up/down the backend pods & verify ECMP load balancing works

Navigate to *OpenShift Dashboard > Applications > Replication Controllers (demo-ingress)*



Choose a replication-controller & click on the arrows to **scale up/down**. Verify the ECMP load balancing is now across the scaled-up pods (Eg. 7 as opposed to 3 before)

