**NGINX+ KIC Lab Guide**

The detail lab documentation for the workshop can be found here:  
<http://udf.nginx-experience.com/>

Note that you may need to change your YAML files accordingly when required as shown in this lab guide.

Contents

[Demo Environment: 3](#_Toc112862477)

[Getting Around the Lab: 4](#_Toc112862478)

[Create kubectl autocomplete: 5](#_Toc112862479)

[Lab 3.1: Deploy Arcadia Application in AWS EKS (K8s): 6](#_Toc112862480)

[3.1.1 Deploy the application (Arcadia) in K8s 6](#_Toc112862481)

[3.1.2 Verify the deployed pods 7](#_Toc112862482)

[3.1.3 Verify the deployed services 7](#_Toc112862483)

[Lab 3.2: NGINX K8s Ingress Controller: 8](#_Toc112862484)

[3.2.1 NGINX+ K8s Ingress Installation 9](#_Toc112862485)

[3.2.2 Review each installation manifest file 9](#_Toc112862486)

[3.2.3 Expose the NGINX+ Ingress Dashboard Service 10](#_Toc112862487)

[3.2.4 Publish the application (Arcadia) to the Internet 12](#_Toc112862488)

[3.2.5 Enable HTTPS & Health Checks Monitoring 15](#_Toc112862489)

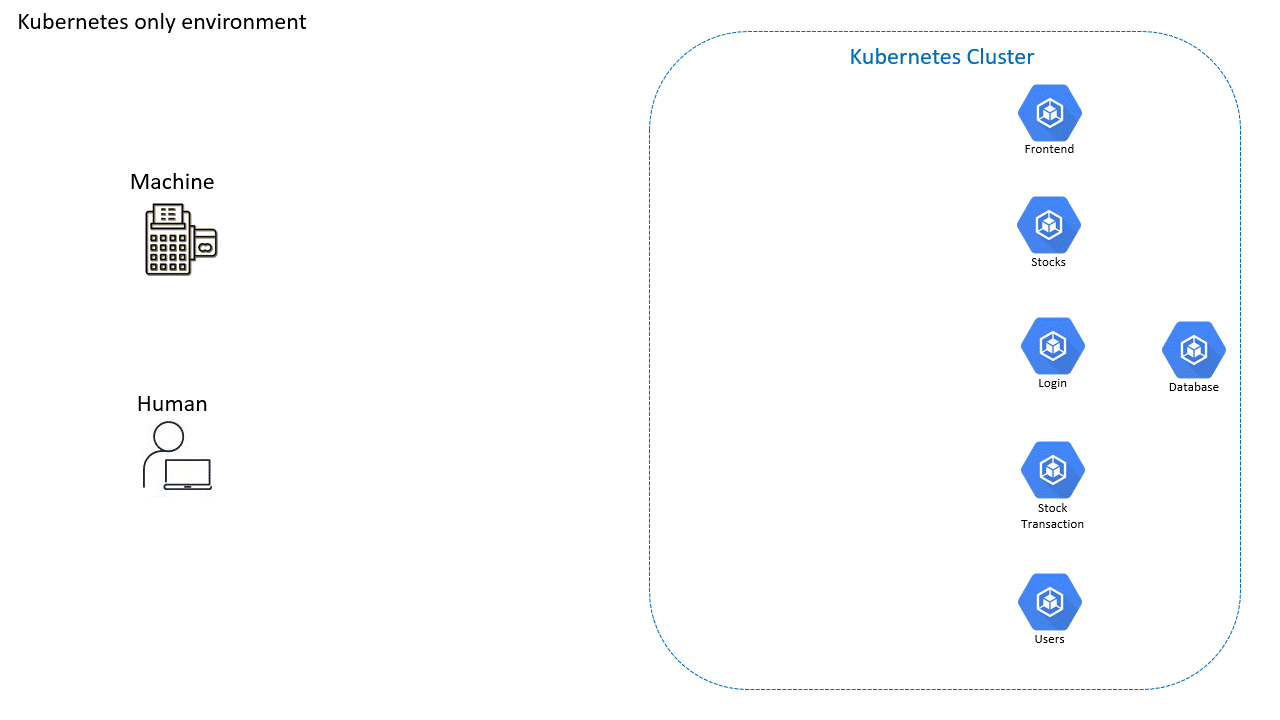
[3.2.6 Using VirtualServer and VirtualServerRoute CRDs 20](#_Toc112862490)

Demo Environment:

This workshop will provide guidelines on how to deploy an application from scratch in Amazon  
Elastic Kubernetes Service environment while protecting and enhancing the application availability  
and usability with NGINX+ solutions.

For this workshop we are going to use the “Arcadia Crypto” application. The application is built with  
6 different microservices that are deployed in the Kubernetes environment.

By the end of the workshop the “Arcadia Crypto” will be fully deployed and protected as described in  
the below diagram.



Getting Around the Lab:  
Once you login to the UDF environment, click on deployment

Graphical user interface, application

Description automatically generated

Navigate to **Jumpbox**, click on **ACCESS** via **Web Shell**

Text

Description automatically generated

In the Web Shell, navigate to **/home/ubuntu/startup** directory

**cd /home/ubuntu/startup**

Text

Description automatically generated

Verify the terraform deployment status. The progress may take 15 – 20 mins to complete.

**tail -f startup.log**

Text

Description automatically generatedPress **Ctrl-c** to return to shell prompt.

Ensure the worker node is up and running, switch from **root** to **ubuntu**

|  |
| --- |
| **# su ubuntu $ kubectl get nodes** |

A picture containing text

Description automatically generated

Ensure you can view the k8s worker node

Navigate to /**home/ubuntu/lab** for the lab files:

**$ cd /home/ubuntu/lab**

A picture containing graphical user interface

Description automatically generated

# Create kubectl autocomplete:

**source <(kubectl completion bash)**

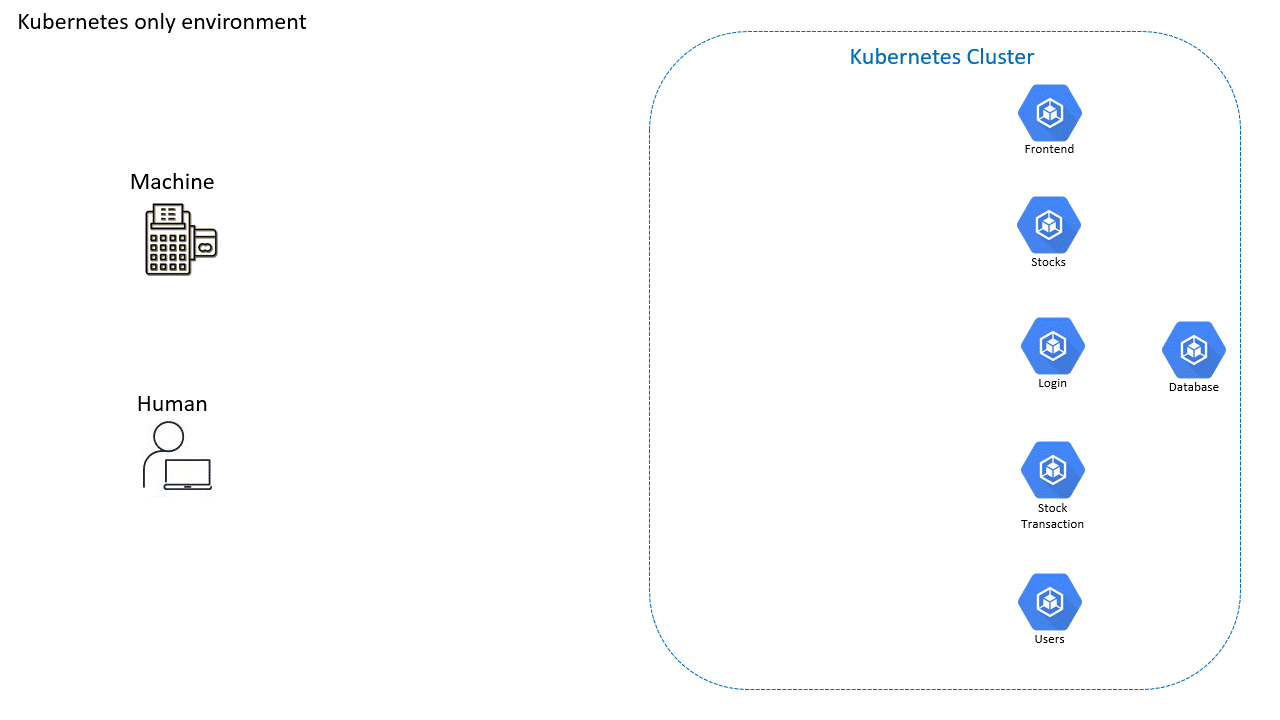
**echo "source <(kubectl completion bash)" >> ~/.bashrc**

**alias k=kubectl**

**complete -F \_\_start\_kubectl k**

Lab 3.1: Deploy Arcadia Application in AWS EKS (K8s):  
As stated before, these are the 6 microservices which we will deploy.

* Frontend - serves the non-dynamic content for like html, js, css and images
* Login - in in charge of dealing with anything related to the login user functionality
* Users - all user data interaction is done through this microservice only
* Stocks - connects to external resources to get the latest crypto data and serves it to the  
  application clients
* Stocks Transaction - Deal with all related to buying or selling crypto currencies. It  
  interacts with other microservices like Users and Stocks
* Database - Database where all information is stored



3.1.1 Deploy the application (Arcadia) in K8s

Ensure that you’re in the /**home/ubuntu/lab** directory, deploy the Arcadia Crypto application  
**$ kubectl apply -f files/4ingress/1arcadia\_delpoy.yaml**

Text

Description automatically generated

3.1.2 Verify the deployed pods

|  |
| --- |
| **$ kubectl get pods** |

Text

Description automatically generated

3.1.3 Verify the deployed services

|  |
| --- |
| **$ kubectl get svc -o wide** |

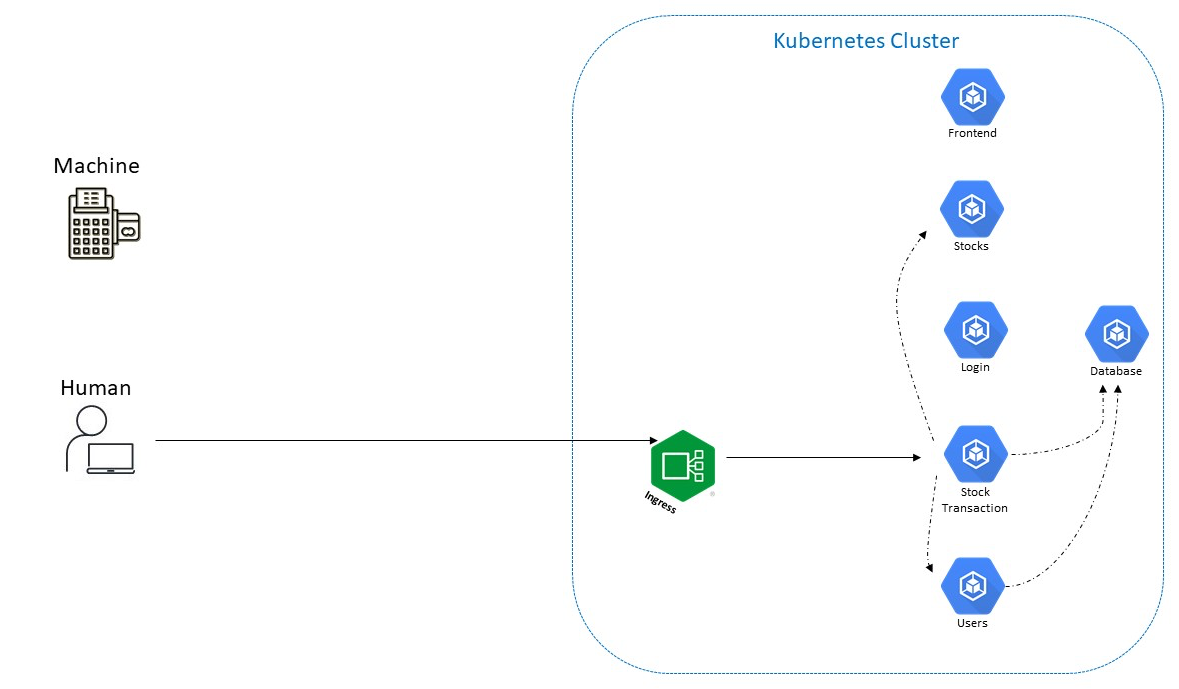
A picture containing text

Description automatically generated

As of now the services are not accessible, we shall deploy the NGINX+ Ingress in the following lab.

Lab 3.2: NGINX K8s Ingress Controller:

Previously we have deployed the application but did not expose the services.  
We need to be able to route the requests to the relevant service.  
The NGINX Ingress Controller for Kubernetes provides enterprise-grade delivery services for  
Kubernetes applications, with benefits for users of both NGINX Open Source and NGINX Plus. With  
the NGINX Ingress Controller for Kubernetes, you get **basic load balancing**, **SSL/TLS termination,**support for **URI rewrites**, and **upstream SSL/TLS encryption**. NGINX Plus (NGINX+) users additionally get **session persistence** for stateful applications and **JSON Web Token (JWT) authentication** for APIs.



3.2.1 NGINX+ K8s Ingress InstallationWe are going to use the NGINX installation manifest based the NGINX Ingress Controller installation  
guide. To simplify and the pace of the lab we have already prepared an installation script.

|  |
| --- |
| 💡 You’re encourage to view the installation **ingress\_install.sh** shell script located at /**home/ubuntu/lab/files/4ingress** |

Run the command below, to run the ingress installation shell script. <Ensure you’re still at the  
**/home/ubuntu/lab** directory>  
**$ ./files/4ingress/ingress\_install.sh**

Text

Description automatically generated

3.2.2 Review each installation manifest file

We are going to review each installation manifest file as shown in the above screenshot. Also refer to this installation link: <https://docs.nginx.com/nginx-ingress-controller/installation/installation-with-manifests/>

3.2.3 Expose the NGINX+ Ingress Dashboard ServiceCopy all the below to the Web-shell to create a yaml file to expose NGINX+ Ingress Dashboard Service

cat << EOF > ingress-dashboard-svc.yaml

apiVersion: v1

kind: Service

metadata:

name: dashboard-nginx-ingress

namespace: nginx-ingress

annotations:

service.beta.kubernetes.io/aws-load-balancer-backend-protocol: "tcp"

service.beta.kubernetes.io/aws-load-balancer-type: nlb

spec:

type: LoadBalancer

ports:

- port: 80

targetPort: 8080

protocol: TCP

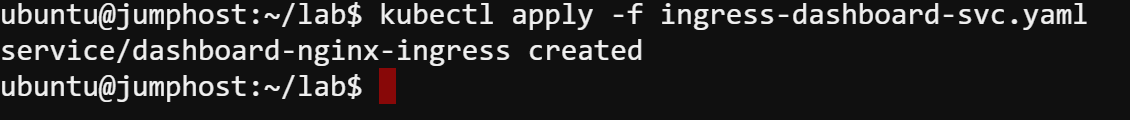
name: http

selector:

app: nginx-ingress

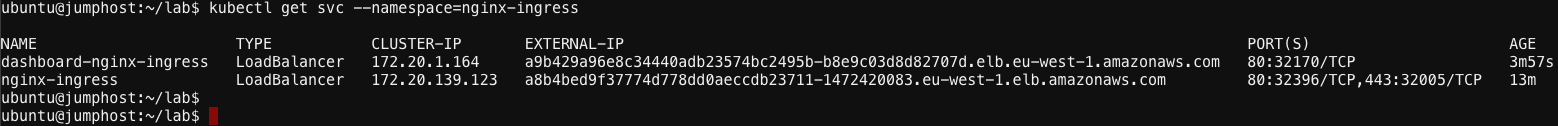
EOF

**$ kubectl apply -f ingress-dashboard-svc.yaml**



Get the NGINX+ ingress service in **nginx-ingress** namespace

**$ kubectl get svc --namespace=nginx-ingress**

  
Note the EXTERNAL-IP of the “dashboard-nginx-ingress”: This is the hostname that we will be using to view the NGINX+ Dashboard.

Copy all the below to the Web-shell to save the EXTERNAL-IPs as env variables for later use:

**export dashboard\_nginx\_ingress=$(kubectl get svc dashboard-nginx-ingress --namespace=nginx-ingress | tr -s " " | cut -d' ' -f4 | grep -v "EXTERNAL-IP")  
export nginx\_ingress=$(kubectl get svc nginx-ingress --namespace=nginx-ingress | tr -s " " | cut -d' ' -f4 | grep -v "EXTERNAL-IP")**



Copy all the below to the Web-shell to check on these saved environment variables:

**echo $dashboard\_nginx\_ingress**

**echo $nginx\_ingress**

Browse to the following location and verify you can see the NGINX+ dashboard:  
http://<DASHBOARD-EXTERNAL-IP>/dashboard.html

**Copy your own EXTERNAL-IP**For example  
http://a9b429a96e8c34440adb23574bc2495b-b8e9c03d8d82707d.elb.eu-west-  
1.amazonaws.com/dashboard.html

Note the EXTERNAL-IP of the “nginx-ingress”: This is the hostname that we will be using for publishing the Arcadia web application.

Browse to the following location and verify that you receive a “404 status” code:  
http://<INGRESS-EXTERNAL-IP>/

**Copy your own EXTERNAL-IP**For example  
http://a8b4bed9f37774d778dd0aeccdb23711-1472420083.eu-west-1.elb.amazonaws.com

3.2.4 Publish the application (Arcadia) to the Internet  
**Expose the application services and route the traffic based on the HTTP path, expose the Arcadia  
app to the world  
https://docs.nginx.com/nginx-ingress-controller/configuration/virtualserver-andvirtualserverroute-resources/**

Copy all the below to the Web-shell to create a yaml file to create a NGINX+ Ingress VirtualServer. Review this ingress which only uses VirtualServer CRD and observe those highlighted in yellow:

cat << EOF > ingress-vs-only.yaml

apiVersion: k8s.nginx.org/v1

kind: VirtualServer

metadata:

name: arcadia

spec:

host: $nginx\_ingress

upstreams:

- name: arcadia-users

service: arcadia-users

port: 80

- name: arcadia-login

service: arcadia-login

port: 80

- name: arcadia-stocks

service: arcadia-stocks

port: 80

- name: arcadia-stock-transaction

service: arcadia-stock-transaction

port: 80

- name: arcadia-frontend

service: arcadia-frontend

port: 80

routes:

- path: /v1/user

action:

pass: arcadia-users

- path: /v1/login

action:

pass: arcadia-login

- path: /v1/stock

action:

pass: arcadia-stocks

- path: /v1/stockt

action:

pass: arcadia-stock-transaction

- path: /

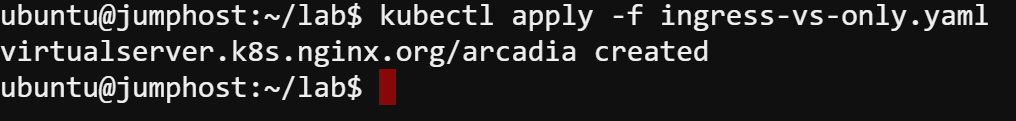
action:

pass: arcadia-frontend

EOF

Apply the yaml file:

**$ kubectl apply -f ingress-vs-only.yaml**



Observe how the various HTTP paths (/v1/user, /v1/login, /v1/stock, /v1/stockt) are routed by Ingress to the relevant K8s services.

At this stage the basic install is finished and all that’s left is to check the connectivity to the Arcadia  
web application. Get the public hostname of the exposed **nginx-ingress** service.

Browse to the following location and verify that you can access the site: http://<INGRESSEXTERNAL-IP>/

To get your External IP, you can run either of these 2 commands (one of them is a saved environment variable):

* **echo $nginx\_ingress**
* **kubectl get svc --namespace=nginx-ingress**

Login to the application using the following credentials:  
Username: satoshi@bitcoin.com  
Password: bitcoin

Graphical user interface, application, Teams

Description automatically generated

Graphical user interface, application, Teams

Description automatically generated

At the moment, we still have **2 missing key features**:

1. We are serving **only HTTP, not HTTPS**. We want our site to be fully secured, therefore all  
   communications need to be encrypted.
2. We are **not actively monitoring the health** of the pods through the data path.

We will increase the number of pods (replicas) to 2 for each deployment: **$ kubectl apply -f files/4ingress/1arcadia\_increase.yaml**

|  |
| --- |
| 💡 You’re encourage to view the yaml **files/4ingress/1arcadia\_increase.yaml** file located at /**home/ubuntu/lab/files/4ingress cat /home/ubuntu/lab/files/4ingress/1arcadia\_increase.yaml** |

Text

Description automatically generated

Look at the NGINX+ dashboard and click on “HTTP Upstreams”, you can see that right now all services  
have two members but no health check

A picture containing table

Description automatically generated

3.2.5 Enable HTTPS & Health Checks Monitoring  
We will enable health check & https for the applications

Copy all the below to the Web-shell to create a yaml file to store TLS certificate and key as K8s secret.

cat << EOF > tls-secret.yaml

apiVersion: v1

data:

tls.crt: 

tls.key: 

kind: Secret

metadata:

name: arcadia-wildcard

type: kubernetes.io/tls

EOF

Apply the yaml file:

**$ kubectl apply -f** **tls-secret.yaml**

Text

Description automatically generated

Make a copy of the existing ingress-vs.yaml file to add TLS and health checks.

**$ cp ingress-vs-only.yaml ingress-vs-https-healthchecks.yaml**

**$ vi ingress-vs-https-healthchecks.yaml**

Add those lines which are highlighted in yellow into your yaml file.

apiVersion: k8s.nginx.org/v1

kind: VirtualServer

metadata:

name: arcadia

spec:

host: $nginx\_ingress

***tls:***

***secret: arcadia-wildcard*** # Represents the server certificate

***redirect:***

***enable: true*** # Always redirect to https if incoming request is http

upstreams:

- name: arcadia-users

service: arcadia-users

port: 80

***healthCheck:*** # This is the most basic healthcheck config. For more info, follow this link https://docs.nginx.com/nginx-ingress-controller/configuration/virtualserver-and-virtualserverroute-resources/#upstream-healthcheck

***enable: true***

***path: /healthz***

- name: arcadia-login

service: arcadia-login

port: 80

***healthCheck:***

***enable: true***

***path: /healthz***

- name: arcadia-stocks

service: arcadia-stocks

port: 80

***healthCheck:***

***enable: true***

***path: /healthz***

- name: arcadia-stock-transaction

service: arcadia-stock-transaction

port: 80

***healthCheck:***

***enable: true***

***path: /healthz***

- name: arcadia-frontend

service: arcadia-frontend

port: 80

***healthCheck:***

***enable: true***

***path: /healthz***

routes:

- path: /v1/user

action:

pass: arcadia-users

- path: /v1/login

action:

pass: arcadia-login

- path: /v1/stock

action:

pass: arcadia-stocks

- path: /v1/stockt

action:

pass: arcadia-stock-transaction

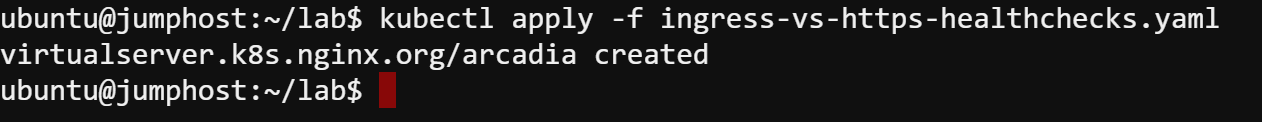
- path: /

action:

pass: arcadia-frontend

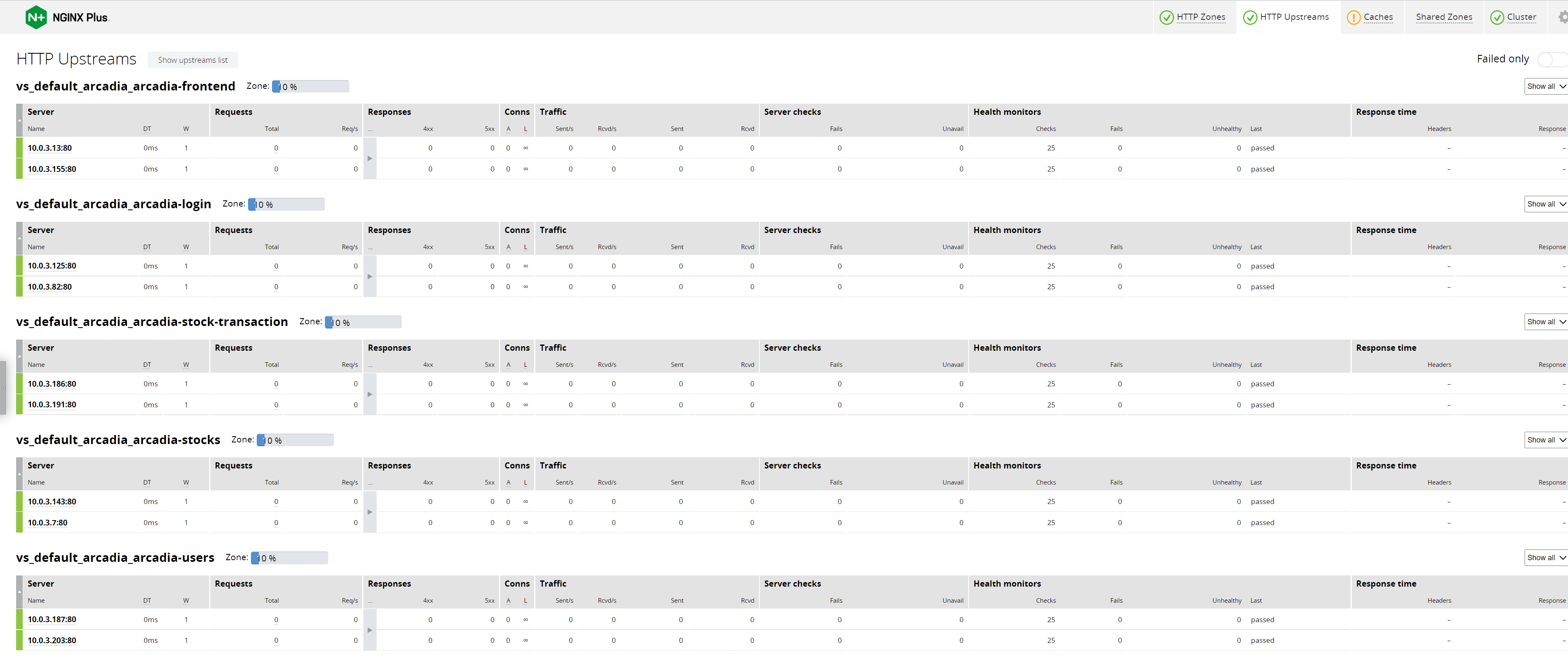
Apply the yaml file:

**$ kubectl apply -f ingress-vs-https-healthchecks.yaml**



|  |
| --- |
| Alternatively, you can copy all the below to the Web-shell to create a new yaml file:  cat << EOF > ingress-vs-https-healthchecks.yaml  apiVersion: k8s.nginx.org/v1  kind: VirtualServer  metadata:  name: arcadia  spec:  host: $nginx\_ingress  tls:  secret: arcadia-wildcard  redirect:  enable: true  upstreams:  - name: arcadia-users  service: arcadia-users  port: 80  healthCheck:  enable: true  path: /healthz  - name: arcadia-login  service: arcadia-login  port: 80  healthCheck:  enable: true  path: /healthz  - name: arcadia-stocks  service: arcadia-stocks  port: 80  healthCheck:  enable: true  path: /healthz  - name: arcadia-stock-transaction  service: arcadia-stock-transaction  port: 80  healthCheck:  enable: true  path: /healthz  - name: arcadia-frontend  service: arcadia-frontend  port: 80  healthCheck:  enable: true  path: /healthz  routes:  - path: /v1/user  action:  pass: arcadia-users  - path: /v1/login  action:  pass: arcadia-login  - path: /v1/stock  action:  pass: arcadia-stocks  - path: /v1/stockt  action:  pass: arcadia-stock-transaction  - path: /  action:  pass: arcadia-frontend  EOF  Apply the yaml file:  **$ kubectl apply -f ingress-vs-https-healthchecks.yaml** |

Browse to the Arcadia website with HTTP and you will be automatically redirected to HTTPS.  
Verify the HTTP Upstreams in NGINX+ Dashboard and observe that NGINX+ has started health monitoring for the pods.



3.2.6 Using VirtualServer and VirtualServerRoute CRDs  
Finally, we will explore the use of VirtualServer and VirtualServerRoute CRDs which is the key feature in NGINX+ Ingress Controller.

Review the existing ingress resource which only uses VirtualServer CRD and observe the differences (highlighted in yellow):

apiVersion: k8s.nginx.org/v1

kind: VirtualServer

metadata:

name: arcadia

spec:

host: $nginx\_ingress

tls:

secret: arcadia-wildcard

redirect:

enable: true

upstreams:

- name: arcadia-users

service: arcadia-users

port: 80

healthCheck:

enable: true

path: /healthz

- name: arcadia-login

service: arcadia-login

port: 80

healthCheck:

enable: true

path: /healthz

- name: arcadia-stocks

service: arcadia-stocks

port: 80

healthCheck:

enable: true

path: /healthz

- name: arcadia-stock-transaction

service: arcadia-stock-transaction

port: 80

healthCheck:

enable: true

path: /healthz

- name: arcadia-frontend

service: arcadia-frontend

port: 80

healthCheck:

enable: true

path: /healthz

routes:

- path: /v1/user

action:

pass: arcadia-users

- path: /v1/login

action:

pass: arcadia-login

- path: /v1/stock

action:

pass: arcadia-stocks

- path: /v1/stockt

action:

pass: arcadia-stock-transaction

- path: /

action:

pass: arcadia-frontend

Copy all the below to the Web-shell to create a yaml file with VirtualServer and VirtualServerRoute CRDs. Review the below ingress resource yaml file which uses both VirtualServer and VirtualServerRoute CRDs and observe the differences (highlighted in yellow).

cat << EOF > ingress-vs-vsr.yaml

apiVersion: k8s.nginx.org/v1

kind: VirtualServer

metadata:

name: arcadia

spec:

host: $nginx\_ingress

tls:

secret: arcadia-wildcard

redirect:

enable: true

upstreams:

- name: arcadia-frontend

service: arcadia-frontend

port: 80

healthCheck:

enable: true

path: /healthz

routes:

- path: /

action:

pass: arcadia-frontend

- path: /v1

route: default/arcadia-v1

---

apiVersion: k8s.nginx.org/v1

kind: VirtualServerRoute

metadata:

name: arcadia-v1

spec:

host: $nginx\_ingress

upstreams:

- name: arcadia-users

service: arcadia-users

port: 80

healthCheck:

enable: true

path: /healthz

- name: arcadia-login

service: arcadia-login

port: 80

healthCheck:

enable: true

path: /healthz

- name: arcadia-stocks

service: arcadia-stocks

port: 80

healthCheck:

enable: true

path: /healthz

- name: arcadia-stock-transaction

service: arcadia-stock-transaction

port: 80

healthCheck:

enable: true

path: /healthz

subroutes:

- path: /v1/user

action:

pass: arcadia-users

- path: /v1/login

action:

pass: arcadia-login

- path: /v1/stock

action:

pass: arcadia-stocks

- path: /v1/stockt

action:

pass: arcadia-stock-transaction

EOF

Apply the yaml file:

**$ kubectl apply -f ingress-vs-vsr.yaml**

Text

Description automatically generated

Verify the deployed Custom Resource Definitions (CRDs)

**$ kubectl get crd**

Verify the deployed VirtualServer CRD

**$ kubectl get virtualservers.k8s.nginx.org --all-namespaces**

Verify the deployed VirtualServerRoute CRD

**$ kubectl get virtualserverroutes.k8s.nginx.org --all-namespaces**

Text

Description automatically generated

Verify the HTTP Upstreams in NGINX+ Dashboard. Observe the difference between the previous Dashboard view which only uses VirtualServer CRD.

A screenshot of a computer

Description automatically generated with medium confidence

**End**