# ModernApps.ninja starter guide template.

Please reference the content below for formatting examples, and replace with your desired content. # Lab Excercise Page Syle Template - 1st level - Main Header

**Contents:**

* Step 1:
* Step 2:
* Step 3:
* Step 4:
* Step 5:
* Next Steps

## Step 1: 2nd level header, steps often have multiple substeps and subsections

1.1 Uses dotted decimal numbering. This sentence 1.1 is a substep of step 1. Use a single decimal format for each substep that itself does not have other substeps. For substeps that have their own substeps, use a subsection format shown in steps 1.2 and 1.3

This format is intended to find an optimal balance of usability for the user and flexibility and simplicity for content developers. As this paragraph demonstrates, its perfectly fine to add prose inline within each step as needed to sufficiently explain the step, keeping in mind that it is crucial for user experience to keep the document streamlined, and so recommend liberal use of hidden and expandable section blocks as shown below

Click to expand

If you have any long text sections such as detailed explanations, code examples, configuration files, etc, please wrap them in expanding sections as shown here.

Keep in mind this template is optimized for Lab Exercise guides which generally include lots of tasks that the reader needs to do.

Also please place all images inside expanding blocks, further details about images will be shown in step 1.4 below

1.2 Minor subsection headers

1.2.1 if you have a substep that includes its own substeps, you need a subsection. This style guide offers two options for subsection handling, the minor subsection format shown here in step 1.2, and the major subsection format shown in step 1.3

### 1.3 Major Subsection Headers

Use major subsection headers whenever they are a better fit for the flow of your document. It is fine to use both minor and major subsection styles within the same document, so long as the overall flow and organization of the document make sense to the reader

The rest of the text below is sample text copied from a lab exercise guide that uses this style

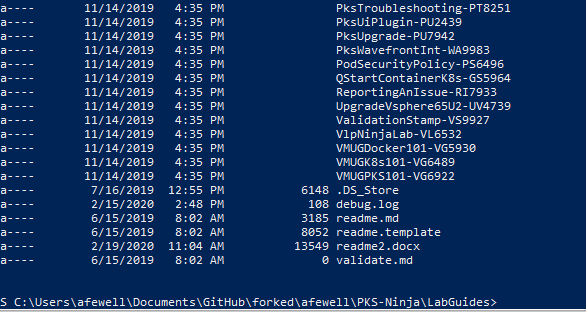
1.3.1 Make a copy of the frontend-deployment\_all\_k8s.yaml file, save it as frontend-deployment\_ingress.yaml

Example: cp frontend-deployment\_all\_k8s.yaml frontend-deployment\_ingress.yaml

1.3.2 Get the URL of your smarcluster with the following command, be sure to replace ‘afewell-cluster’ with the name of your cluster:

vke cluster show afewell-cluster | grep Address

Screenshot 1.3.2



1.3.3 Edit the frontend-deployment\_ingress.yaml file, near the bottom of the file in the ingress spec section, change the value for spec.rules.host to URL for your smartcluster as shown in the following snippet:

NOTE: Be sure to replace the URL shown here with the URL for your own smartcluster

spec:  
 rules:  
 - host: afewell-cluster-69fc65f8-d37d-11e8-918b-0a1dada1e740.fa2c1d78-9f00-4e30-8268-4ab81862080d.vke-user.com  
 http:  
 paths:  
 - backend:  
 serviceName: planespotter-frontend  
 servicePort: 80

Click to expand to see the full contents of frontend-deployment\_ingress.yaml

When reviewing the file contents below, observe that it includes a ClusterIP service spec which only provides an IP address that is usable for pod-to-pod communications in the cluster. The file also includes an ingress spec which implements the default VKE ingress controller.

In the following steps after you deploy the planespotter-frontend with ingress controller, you will be able to browse from your workstation to the running planespotter app in your VKE environment even though you have not assigned a nat or public IP for the service.

Ingress controllers act as a proxies, recieving http/s requests from external clients and then based on the URL hostname or path, the ingress controller will proxy the request to the corresponding back-end service. For example mysite.com/path1 and mysite.com/path2 can be routed to different backing services running in the kubernetes cluster.

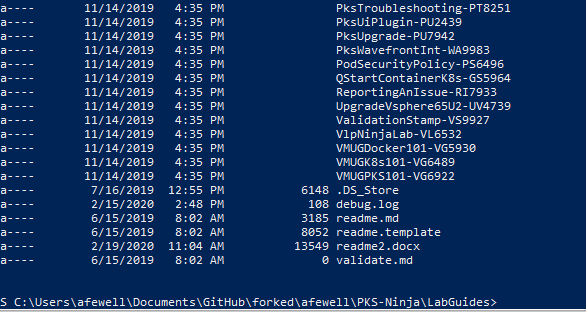
In the file below, no rules are specified to different paths and so accordingly, all requests sent to the host defined in the spec, your VKE SmartCluster URL, will be proxied by the ingress controller to the planespotter-frontend ClusterIP service also defined in the frontend-deployment\_ingress.yaml file

---  
apiVersion: apps/v1beta1  
kind: Deployment  
metadata:  
 name: planespotter-frontend  
 namespace: planespotter  
 labels:  
 app: planespotter-frontend  
 tier: frontend  
spec:  
 replicas: 2  
 selector:  
 matchLabels:  
 app: planespotter-frontend  
 template:  
 metadata:  
 labels:  
 app: planespotter-frontend  
 tier: frontend  
 spec:  
 containers:  
 - name: planespotter-fe  
 image: yfauser/planespotter-frontend:d0b30abec8bfdbde01a36d07b30b2a3802d9ccbb  
 imagePullPolicy: IfNotPresent  
 env:  
 - name: PLANESPOTTER\_API\_ENDPOINT  
 value: planespotter-svc  
 - name: TIMEOUT\_REG  
 value: "5"  
 - name: TIMEOUT\_OTHER  
 value: "5"  
---  
apiVersion: v1  
kind: Service  
metadata:  
 name: planespotter-frontend  
 namespace: planespotter  
 labels:  
 app: planespotter-frontend  
spec:  
 ports:  
 # the port that this service should serve on  
 - port: 80  
 selector:  
 app: planespotter-frontend  
---  
apiVersion: extensions/v1beta1  
kind: Ingress  
metadata:  
 name: planespotter-frontend  
 namespace: planespotter  
spec:  
 rules:  
 - host: afewell-cluster-69fc65f8-d37d-11e8-918b-0a1dada1e740.fa2c1d78-9f00-4e30-8268-4ab81862080d.vke-user.com  
 http:  
 paths:  
 - backend:  
 serviceName: planespotter-frontend  
 servicePort: 80

1.3.4 Run the updated planespotter-frontend app and verify deployment with the following commands. Make note of the external IP address/hostname shown in the output of kubectl get services

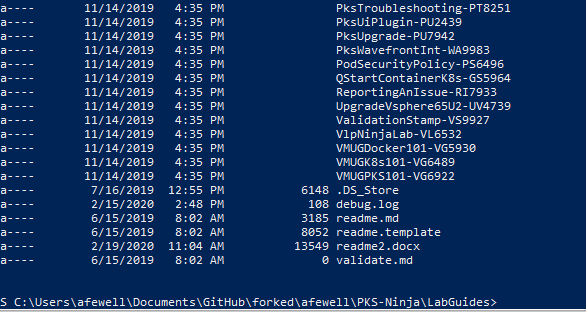
kubectl create -f frontend-deployment\_ingress.yaml  
kubectl get pods  
kubectl get deployments  
kubectl get services  
kubectl get ingress  
kubectl describe ingress

Screenshot 1.3.4



1.3.5 Open a browser and go to the url of your VKE SmartCluster to verify that planespotter-frontend is externally accessible with the LoadBalancer service

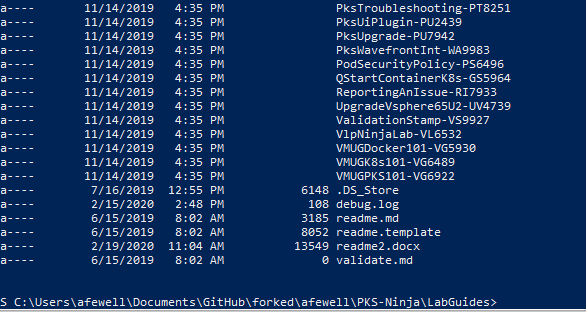
Screenshot 5.5.5



1.3.6 Clean up the planespotter-frontend components and verify with the following commands:

kubectl delete -f frontend-deployment\_ingress.yaml  
kubectl get pods  
kubectl get deployments  
kubectl get services  
kubectl get ingress

Screenshot 5.5.6



## Next Steps

This lab provided an introductory overview of Kubernetes operations. Additional topics such as persistent volumes, network policy, config maps, stateful sets and more will be covered in more detail in the ongoing labs.

If you are following the PKS Ninja cirriculum, [click here to proceed to the next lab](../Lab2-PksInstallationPhaseOne). As you proceed through the remaining labs you will learn more advanced details about Kubernetes using additional planespotter app components as examples and then deploy the complete planespotter application on a PKS environment.

If you are not following the PKS Ninja cirriculum and would like to deploy the complete planespotter app on VKE, you can find [complete deployment instructions here](https://github.com/Boskey/run_kubernetes_with_vmware)

### Thank you for completing the Introduction to Kubernetes Lab!

### [Please click here to proceed to Lab2: PKS Installation Phase 1](../Lab2-PksInstallationPhaseOne)