**Lab: allow a max of 1 hour to complete**

Deploying PHP Guestbook application with Redis

For this lab, we will be using Kubernetes (running locally as part of a Docker Desktop installation) to deploy a web application using Redis as a backing data store. Follow along with the tutorial at <https://kubernetes.io/docs/tutorials/stateless-application/guestbook/>.

Objectives

* Start up a Redis leader
* Start up two Redis followers
* Start up the guestbook frontend
* Expose and view the frontend service
* Clean up

Your Kubernetes installation must be at or later than version v1.14. To check the version, enter `kubectl version` in the terminal.

1. Apply the manifest describing the Redis leader deployment using the following YAML. You can find the manifest at <https://k8s.io/examples/application/guestbook/redis-leader-deployment.yaml>. You can copy/paste or deploy directly from the link but be sure you explore the defined configuration to confirm understanding of each instruction:

Graphical user interface, text, application, email

Description automatically generated

1. Check for pod status using `kubectl get pods`
2. You can view logs using `kubectl logs -f deployment/redis-leader`
3. Apply the manifest describing the Redis leader service using the following YAML. You can find the manifest at <https://k8s.io/examples/application/guestbook/redis-leader-service.yaml>. You can copy/paste or deploy directly from the link but be sure you explore the defined configuration to confirm understanding of each instruction:

Graphical user interface, text, application, email

Description automatically generated

1. Check for service status using `kubectl get service`
2. Apply the manifest describing the Redis follower deployment using the following YAML. You can find the manifest at <https://k8s.io/examples/application/guestbook/redis-follower-deployment.yaml>. You can copy/paste or deploy directly from the link but be sure you explore the defined configuration to confirm understanding of each instruction:

Graphical user interface, text, application, email

Description automatically generated

1. Check pod status as you did with the Redis leader deployment
2. Apply the manifest describing the Redis follower service using the following YAML. You can find the manifest at <https://k8s.io/examples/application/guestbook/redis-follower-service.yaml>. You can copy/paste or deploy directly from the link but be sure you explore the defined configuration to confirm understanding of each instruction:

Graphical user interface, text, application, chat or text message, email

Description automatically generated

1. Check service status as you did with the Redis leader service
2. Apply the manifest describing the guestbook frontend deployment using the following YAML. You can find the manifest at <https://k8s.io/examples/application/guestbook/frontend-deployment.yaml>. You can copy/paste or deploy directly from the link but be sure you explore the defined configuration to confirm understanding of each instruction:

Graphical user interface, text, application, email

Description automatically generated

1. Execute `kubectl get pods -l app=guestbook -l tier=frontend`. What are the “-l” flags used for? How many replicas are running?
2. Apply the manifest describing the guestbook frontend service as a LoadBalancer service type using the following YAML. You can find the manifest at <https://k8s.io/examples/application/guestbook/frontend-service.yaml>. You can copy/paste or deploy directly from the link but be sure you explore the defined configuration to confirm understanding of each instruction:

Graphical user interface, text, application, email

Description automatically generated

1. Execute `kubectl get service frontend` to query the external IP address. Try hitting <http://external-ip> to access the frontend and try adding some guestbook entries.
2. Execute `kubectl scale deployment frontend --replicas=5` to scale out the number of frontend instances; use `kubectl get pods` to verify
3. Execute `kubectl scale deployment frontend --replicas=2` to scale in the number of frontend instances; use `kubectl get pods` to verify
4. Execute `kubectl delete deployment -l app=redis` to remove all Redis-related deployments
5. Execute `kubectl delete service -l app=redis` to remove all Redis-related services
6. Execute `kubectl delete deployment frontend` to remove all frontend-related deployments/instances
7. Execute `kubectl delete service frontend` to remove all frontend-related services

**Pluralsight Courses – Getting Started with Docker & Kubernetes**

View and complete the following Pluralsight courses:

* <https://app.pluralsight.com/library/courses/getting-started-docker/table-of-contents> (Duration is ~1h 26m)
* <https://app.pluralsight.com/library/courses/kubernetes-getting-started/table-of-contents> (Duration is ~3h 3m)

**Articles for Follow-up Reading (x4)**

1. VMware Tanzu Community Edition 🡪 Overview at <https://tanzucommunityedition.io/docs/latest/>
2. VMware Tanzu Community Edition 🡪 Architecture at <https://tanzucommunityedition.io/docs/latest/architecture/>
3. An Introduction to Kubernetes 🡪 <https://www.digitalocean.com/community/tutorials/an-introduction-to-kubernetes>
4. 4 Reasons You Should Use Kubernetes 🡪 <https://www.infoworld.com/article/3173266/4-reasons-you-should-use-kubernetes.html>