Tutorial: Update an application in Azure Kubernetes Service (AKS)

Article • 12/21/2021 • 4 minutes to read • 12 contributors

In this article

Before you begin
Update an application
Update the container image
Test the application locally
Tag and push the image
Deploy the updated application
Test the updated application
Next steps

After an application has been deployed in Kubernetes, it can be updated by specifying a new container image or image version. An update is staged so that only a portion of the deployment is updated at the same time. This staged update enables the application to keep running during the update. It also provides a rollback mechanism if a deployment failure occurs.

In this tutorial, part six of seven, the sample Azure Vote app is updated. You learn how to:

- ✓ Update the front-end application code
- ✓ Create an updated container image
- ✓ Push the container image to Azure Container Registry
- ✓ Deploy the updated container image

Before you begin

In previous tutorials, an application was packaged into a container image. This image was uploaded to Azure Container Registry, and you created an AKS cluster. The application was then deployed to the AKS cluster.

An application repository was also cloned that includes the application source code, and a pre-created Docker Compose file used in this tutorial. Verify that you've created a clone of the repo, and have changed directories into the cloned directory. If you haven't completed these steps, and want to follow along, start with Tutorial 1 – Create container images.

Azure CLI | Azure PowerShell

This tutorial requires that you're running the Azure CLI version 2.0.53 or later. Run `az --version` to find the version. If you need to install or upgrade, see Install Azure CLI.

Update an application

Let's make a change to the sample application, then update the version already deployed to your AKS cluster. Make sure that you're in the cloned *azure-voting-app-redis* directory. The sample application source code can then be found inside the *azure-vote* directory. Open the *config_file.cfg* file with an editor, such as `vi`:

Console | Copy

```
vi azure-vote/azure-vote/config_file.cfg
```

Change the values for *VOTE1VALUE* and *VOTE2VALUE* to different values, such as colors. The following example shows the updated values:

Copy

```
# UI Configurations
TITLE = 'Azure Voting App'
VOTE1VALUE = 'Blue'
VOTE2VALUE = 'Purple'
SHOWHOST = 'false'
```

Save and close the file. In `vi`, use `:wq`.

Update the container image

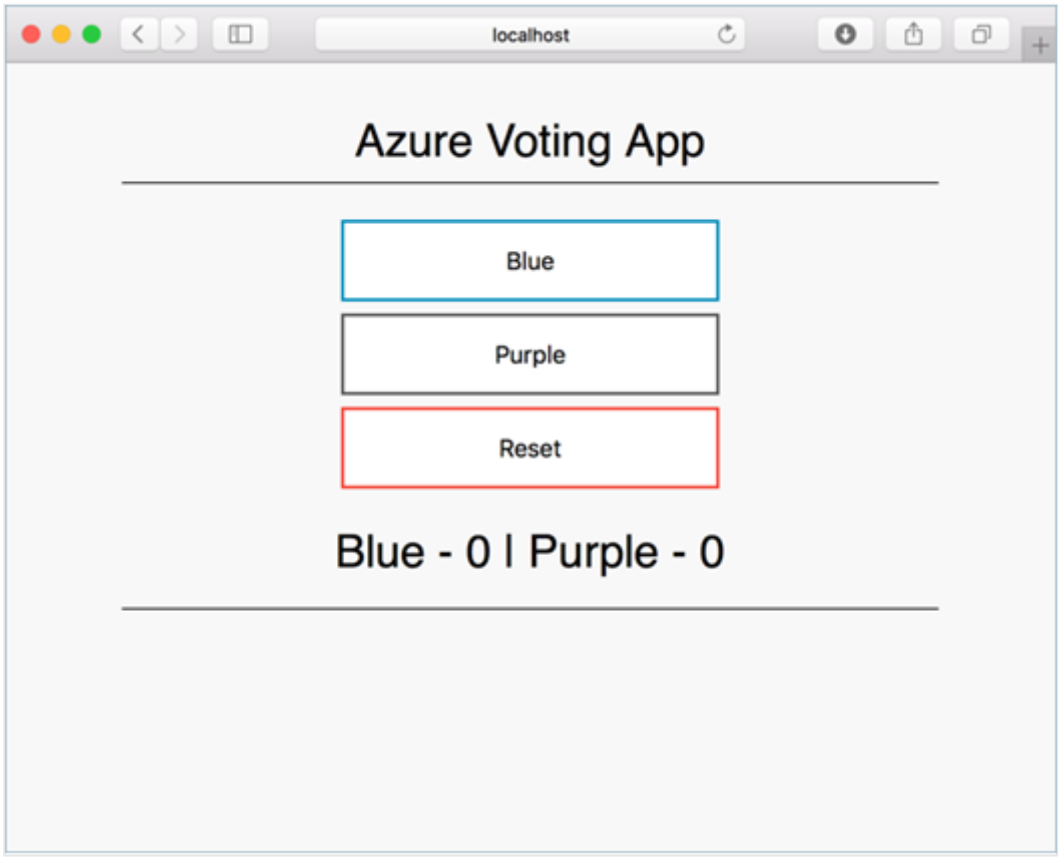
To re-create the front-end image and test the updated application, use docker-compose . The `--build` argument is used to instruct Docker Compose to re-create the application image:

Console Copy

```
docker-compose up --build -d
```

Test the application locally

To verify that the updated container image shows your changes, open a local web browser to `http://localhost:8080`.



The updated values provided in the *config_file.cfg* file are displayed in your running application.

Tag and push the image

Azure CLI | Azure PowerShell

To correctly use the updated image, tag the *azure-vote-front* image with the login server name of your ACR registry. Get the login server name with the az acr list command:

Azure CLI Copy

```
az acr list --resource-group myResourceGroup --query "[].
{acrLoginServer:loginServer}" --output table
```

Use docker tag to tag the image. Replace `<acrLoginServer>` with your ACR login server name or public registry hostname, and update the image version to *:v2* as follows:

Console Copy

```
docker tag mcr.microsoft.com/azuredocs/azure-vote-front:v1
<acrLoginServer>/azure-vote-front:v2
```

Now use docker push to upload the image to your registry. Replace `<acrLoginServer>` with your ACR login server name.

Azure CLI | Azure PowerShell

Note

If you experience issues pushing to your ACR registry, make sure that you are still logged in. Run the **az acr login** command using the name of your Azure Container Registry that you created in the **Create an Azure Container Registry** step. For example, `az acr login --name <azure container registry name>`.

Console Copy

```
docker push <acrLoginServer>/azure-vote-front:v2
```

Deploy the updated application

To provide maximum uptime, multiple instances of the application pod must be running. Verify the number of running front-end instances with the kubectl get pods command:

Copy

```
$ kubectl get pods

NAME                                READY     STATUS    RESTARTS   AGE
azure-vote-back-217588096-5w632     1/1       Running   0          10m
```

```
azure-vote-front-233282510-b5pkz   1/1       Running   0          10m
azure-vote-front-233282510-dhrtr   1/1       Running   0          10m
azure-vote-front-233282510-pqbfk   1/1       Running   0          10m
```

If you don't have multiple front-end pods, scale the *azure-vote-front* deployment as follows:

Console Copy

```
kubectl scale --replicas=3 deployment/azure-vote-front
```

To update the application, use the kubectl set command. Update `<acrLoginServer>` with the login server or host name of your container registry, and specify the *v2* application version:

Console Copy

```
kubectl set image deployment azure-vote-front azure-vote-front=
<acrLoginServer>/azure-vote-front:v2
```

To monitor the deployment, use the kubectl get pod command. As the updated application is deployed, your pods are terminated and re-created with the new container image.

Console Copy

```
kubectl get pods
```

The following example output shows pods terminating and new instances running as the deployment progresses:

Copy

```
$ kubectl get pods

NAME                               READY     STATUS        RESTARTS   AGE
azure-vote-back-2978095810-gq9g0   1/1       Running       0          5m
azure-vote-front-1297194256-tpjlg  1/1       Running       0          1m
azure-vote-front-1297194256-tptnx  1/1       Running       0          5m
azure-vote-front-1297194256-zktw9  1/1       Terminating   0          1m
```

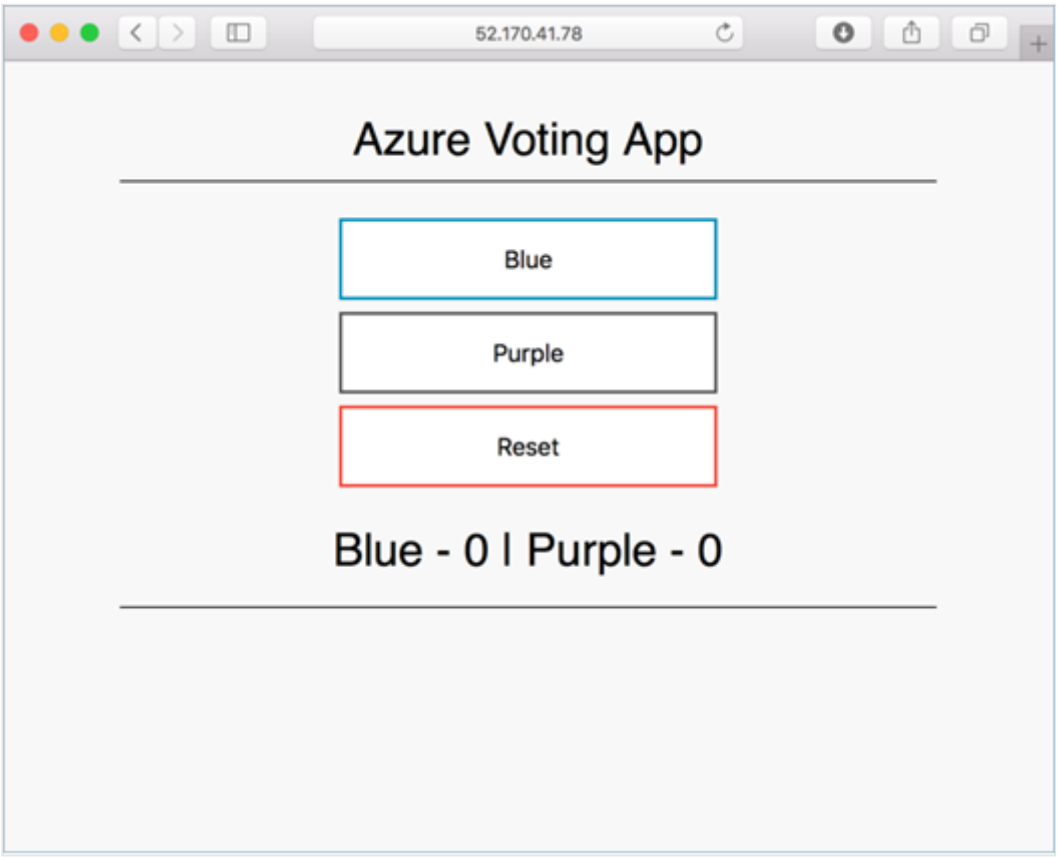
Test the updated application

To view the update application, first get the external IP address of the `azure-vote-front` service:

Console | Copy

```
kubectl get service azure-vote-front
```

Now open a web browser to the IP address of your service:



Next steps

In this tutorial, you updated an application and rolled out this update to your AKS cluster. You learned how to:

- ✓ Update the front-end application code
- ✓ Create an updated container image
- ✓ Push the container image to Azure Container Registry
- ✓ Deploy the updated container image

Advance to the next tutorial to learn how to upgrade an AKS cluster to a new version of Kubernetes.

Upgrade Kubernetes

Recommended content

Kubernetes on Azure tutorial - Upgrade a cluster - Azure Kubernetes Service

In this Azure Kubernetes Service (AKS) tutorial, you learn how to upgrade an existing AKS cluster to the latest available Kubernetes version.

Kubernetes on Azure tutorial - Deploy an application - Azure Kubernetes Service

In this Azure Kubernetes Service (AKS) tutorial, you deploy a multi-container application to your cluster using a custom image stored in Azure Container Registry.

Kubernetes on Azure tutorial - Create a container registry - Azure Kubernetes Service

In this Azure Kubernetes Service (AKS) tutorial, you create an Azure Container Registry instance and upload a sample application container image.

Kubernetes on Azure tutorial - Prepare an application - Azure Kubernetes Service

In this Azure Kubernetes Service (AKS) tutorial, you learn how to prepare and build a multi-container app with Docker Compose that you can then deploy to AKS.

Kubernetes on Azure tutorial - Deploy a cluster - Azure Kubernetes Service

In this Azure Kubernetes Service (AKS) tutorial, you create an AKS cluster and use kubectl to connect to the Kubernetes master node.

Build, test, and deploy containers to Azure Kubernetes Service using GitHub Actions - Azure Kubernetes Service

Learn how to use GitHub Actions to deploy your container to Kubernetes

Kubernetes on Azure tutorial - Scale Application - Azure Kubernetes Service

In this Azure Kubernetes Service (AKS) tutorial, you learn how to scale nodes and pods in Kubernetes, and implement horizontal pod autoscaling.

Develop on Azure Kubernetes Service (AKS) with Helm - Azure Kubernetes Service

Use Helm with AKS and Azure Container Registry to package and run application containers in a cluster.

Show more