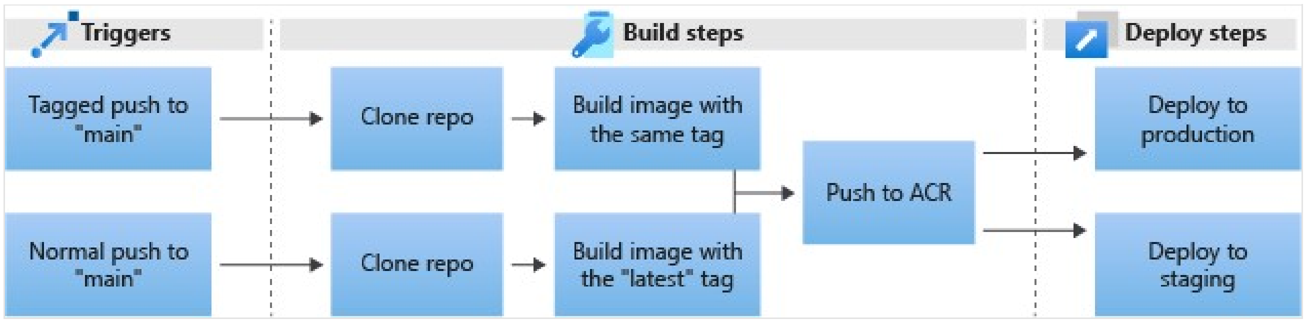
✓ 100 XP

Create the deployment pipeline

10 minutes

With the Helm charts created, you now have all the tools you need to deploy the application to AKS by using GitHub Actions. Let's use what you created to finish the deployment pipeline.

In this unit, you'll tackle the last step in the diagram—the deploy steps.



For staging, the steps include:

- Add a deploy job
- Install Helm
- Get the AKS credentials
- Create a secret
- Deploy the application
- Test the deployment

To deploy to production, we'll:

- Create the production deploy job
- Test the deployment

Create the deploy to staging job

Start by deploying the staging pipeline.

Add a deploy job

1. In GitHub, go to your fork of the repository.

2. Go to the **.github/workflows** directory in the repository, and then open the **build-latest.yml** file.

   The file should look like this example:

   YAML Copy

```yaml
on:
  push:
    branches: [ main ]

jobs:
  build_push_image:
    runs-on: ubuntu-latest

    steps:
      - uses: actions/checkout@v2

      - name: Build and push staging image
        uses: docker/build-push-action@v1.1.1
        with:
          username: ${{ secrets.ACR_LOGIN }}
          password: ${{ secrets.ACR_PASSWORD }}
          registry: ${{ secrets.ACR_NAME }}
          repository: contoso-website
          tags: latest
```

3. To add another job, below the `build_push_image` key, create a new key called `deploy`:

   YAML Copy

```yaml
on:
  push:
    branches: [ main ]

jobs:
  build_push_image:
    runs-on: ubuntu-latest

    steps:
      - uses: actions/checkout@v2

      - name: Build and push staging image
        uses: docker/build-push-action@v1.1.1
        with:
          username: ${{ secrets.ACR_LOGIN }}
          password: ${{ secrets.ACR_PASSWORD }}
          registry: ${{ secrets.ACR_NAME }}
          repository: contoso-website
          tags: latest
  deploy:
    runs-on: ubuntu-latest
    needs: build_push_image # Will wait for the execution of the previous
job
```

4. Clone and check out your working branch.

5. Add `- uses: actions/checkout@v2` as the first step:

YAML Copy

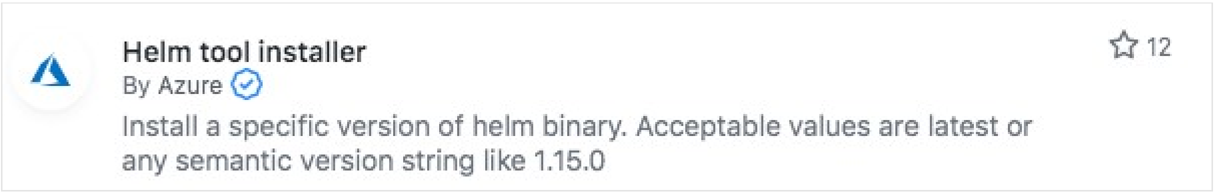
```yaml
  deploy:
    runs-on: ubuntu-latest
    needs: build_push_image

    - uses: actions/checkout@v2
```

Install Helm

In this exercise, you use Helm version `v3.3.1`. Azure has a built action that downloads and installs Helm.

1. Below the `runs-on` key, add a new `steps` key. Then, search for and then select **Helm tool installer**.



Helm tool installer
By Azure
Install a specific version of helm binary. Acceptable values are latest or any semantic version string like 1.15.0

2. Copy the YAML that appears, and then paste it below the `uses` key:

YAML Copy

```yaml
  deploy:
    runs-on: ubuntu-latest
    needs: build_push_image

    steps:
      - uses: actions/checkout@v2

      - name: Helm tool installer
        uses: Azure/setup-helm@v1
        with:
          # Version of helm
          version: # default is latest
```

3. Rename the step name `Install Helm`, and then pin the `version` key to `v3.3.1`:

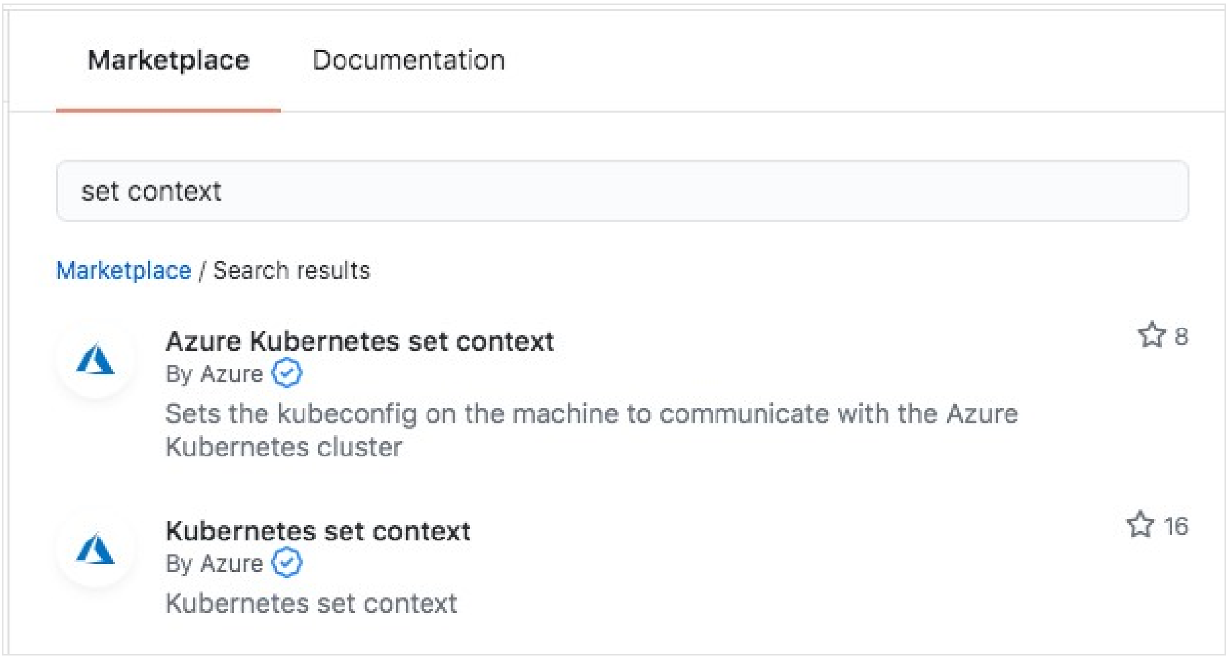
YAML Copy

```yaml
  deploy:
    runs-on: ubuntu-latest
    needs: build_push_image

    steps:
      - uses: actions/checkout@v2

      - name: Install Helm
        uses: Azure/setup-helm@v1
        with:
          version: v3.3.1
```

4. Sign in to your AKS cluster by using the Azure CLI through another action that Azure provides. In the search bar, enter **Set Context**. In the search results, select **Azure Kubernetes Set Context**.



set context

Marketplace / Search results

Azure Kubernetes set context
By Azure
Sets the kubeconfig on the machine to communicate with the Azure Kubernetes cluster
8

Kubernetes set context
By Azure
Kubernetes set context
16

5. Copy the YAML, and then paste it below the previous `Install Helm` step:

YAML Copy

```yaml
steps:
  - uses: actions/checkout@v2

  - name: Install Helm
    uses: Azure/setup-helm@v1
    with:
      version: v3.3.1
  - name: Azure Kubernetes set context
    uses: Azure/aks-set-context@v1
    with:
      # Azure credentials, i.e., output of `az ad sp create-for-rbac --sdk-
auth`
      creds: # default is
      # Resource group name
      resource-group: # optional, default is
      # AKS cluster name
      cluster-name: # optional, default is
```

Get the AKS credentials

Next, you use an action that uses the Azure CLI to get the AKS credentials. Then, you use Kubectl to deploy your workloads to the cluster.

1. Change the `name` key to `Get AKS Credentials`.

2. Change the `resource-group` key to the name of the resource group that contains your AKS resource. You can get this information by running the following command in Cloud Shell:

   Azure CLI — Copy — Try It

   ```
   az aks list -o tsv --query "[?name=='contoso-video'].resourceGroup"
   ```

3. In the `cluster-name` key, enter the cluster name. The name of the AKS cluster in this exercise is fixed as `contoso-video`.

4. In the `creds` key, define a secret called `AZURE_CREDENTIALS`. The value of this key `${{ secrets.AZURE_CREDENTIALS }}`.

   The final YAML should look like this example:

   YAML — Copy

   ```
   name: Build and push the latest build to staging

   on:
     push:
       branches: [ main ]

   jobs:
     build_push_image:
       runs-on: ubuntu-latest
   ```

```yaml
    steps:
      - uses: actions/checkout@v2

      - name: Build and push staging image
        uses: docker/build-push-action@v1.1.1
        with:
          username: ${{ secrets.ACR_LOGIN }}
          password: ${{ secrets.ACR_PASSWORD }}
          registry: ${{ secrets.ACR_NAME }}
          repository: contoso-website
          tags: latest

  deploy:
    runs-on: ubuntu-latest
    needs: build_push_image

    steps:
      - uses: actions/checkout@v2

      - name: Install Helm
        uses: Azure/setup-helm@v1
        with:
          version: v3.3.1

      - name: Get AKS Credentials
        uses: Azure/aks-set-context@v1
        with:
          creds: ${{ secrets.AZURE_CREDENTIALS }}
          resource-group: <your-resource-group>
          cluster-name: contoso-video
```

Create a secret

You've set the credential secret, but the secret isn't created yet. Let's create it.

1. In a new browser tab, go to your fork of the repository. Select **Settings** > **Secrets**.
2. Create a new secret called `AZURE_CREDENTIALS`. The value of this secret will be the output of the following command, a JSON object:

   Azure CLI   Copy   Try It

   ```azurecli
   az ad sp create-for-rbac --sdk-auth
   ```

3. Copy the output and paste it in the secret value. Then, save the secret and close the tab.

Deploy the application

Now, you have access to your cluster and you have Helm installed. The next step is to deploy the application. For this step, you use the command instructions that are native to GitHub Actions.

1. In the YAML file, below the latest step, create a new `- name:` key. Name the key `Run Helm Deploy`. Then, below this key, create another key called `run`.

   The YAML should look like this example:

   YAML Copy

```yaml
name: Build and push the latest build to staging

on:
  push:
    branches: [ main ]

jobs:
  build_push_image:
    runs-on: ubuntu-latest

    steps:
      - uses: actions/checkout@v2

      - name: Build and push staging image
        uses: docker/build-push-action@v1.1.1
        with:
          username: ${{ secrets.ACR_LOGIN }}
          password: ${{ secrets.ACR_PASSWORD }}
          registry: ${{ secrets.ACR_NAME }}
          repository: contoso-website
          tags: latest

  deploy:
    runs-on: ubuntu-latest
    needs: build_push_image

    steps:
      - uses: actions/checkout@v2

      - name: Install Helm
        uses: Azure/setup-helm@v1
        with:
          version: v3.3.1

      - name: Get AKS Credentials
        uses: Azure/aks-set-context@v1
        with:
          creds: ${{ secrets.AZURE_CREDENTIALS }}
          # Resource Group Name
          resource-group: mslearn-gh-pipelines-6667
          # AKS Cluster Name
          cluster-name: contoso-video

      - name: Run Helm Deploy
        run:
```

2. You can use the `run` key to run any shell command inside the container. Because you're using Ubuntu, your shell is Bash. In a moment, we'll run the following command inside the `run` key:

Bash

```bash
helm upgrade \
    --install \
    --create-namespace \
    --atomic \
    --wait \
    --namespace staging \
    contoso-website \
    ./kubernetes/contoso-website \
    --set image.repository=${{ secrets.ACR_NAME }} \
    --set dns.name=${{ secrets.DNS_NAME }}
```

But first, let's look at each parameter to understand what the command does:

Parameter	Action or value
`helm upgrade`	Upgrades an installed release.
`--install`	If the release doesn't exist, install it. This parameter transforms the command into an idempotent command, so you can run it exactly the same multiple times.
`--create-namespace`	If the namespace in the `--namespace` flag doesn't exist, create it.
`--atomic`	If the release fails, remove all workloads that have been installed.
`--wait`	Wait for the release to finish and return `ok`.
`--namespace staging`	Deploy this release to the `staging` namespace. The parameter overrides all `namespace` keys in the manifest files.
`contoso-website`	Release name.

Parameter	Action or value
`./kubernetes/contoso-website`	Chart directory location.
`--set image.repository`	Updates the value of the `image.repository` key in the values.yaml file *for this release only*.
`--set dns.name`	Updates the `dns.name` key in the values.yaml file *for this release only*.

Run the command, starting with the `|` character. The final YAML should look like this example:

YAML

```yaml
# ... File omitted
      - name: Run Helm Deploy
        run: |
          helm upgrade \
            --install \
            --create-namespace \
            --atomic \
            --wait \
            --namespace staging \
            contoso-website \
            ./kubernetes/contoso-website \
            --set image.repository=${{ secrets.ACR_NAME }} \
            --set dns.name=${{ secrets.DNS_NAME }}
```

3. In a new browser tab, in your fork of the repository, select **Settings** > **Secrets**.

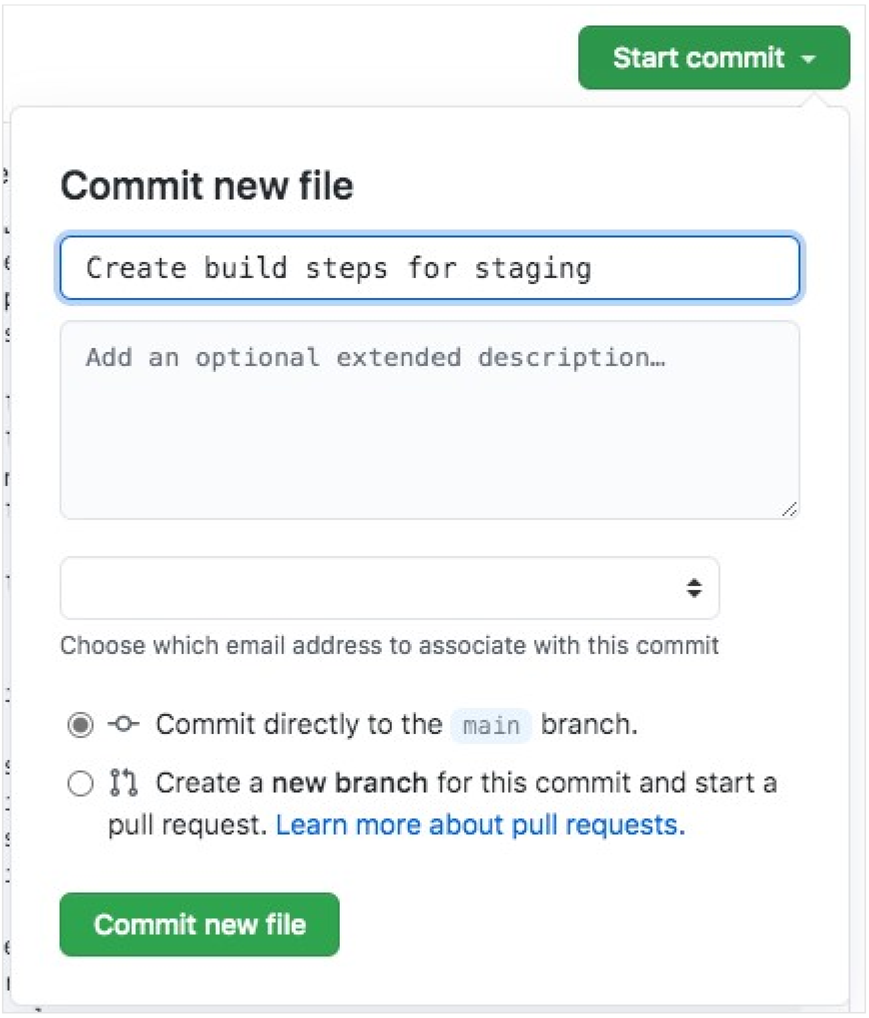
4. Create a new secret called `DNS_NAME`. You can get the value to use for this secret by running the following command in Cloud Shell:

Azure CLI

```
az aks show -g {resource-group-name} -n {aks-cluster-name} -o tsv --query
addonProfiles.httpApplicationRouting.config.HTTPApplicationRoutingZoneName
```

Save the secret and close the browser tab.

5. To commit the changes, select the green **Start commit** button. Enter a description for the commit, and then select **Commit new file**:


Start commit

Commit new file

Create build steps for staging

Add an optional extended description…

Choose which email address to associate with this commit

[✓ Commit directly to the `main` branch.]

Create a **new branch** for this commit and start a pull request. Learn more about pull requests.

Commit new file

The build starts running on the **Actions** tab.

Test the deployment

To test the staging deployment, in your browser, go to **contoso-staging.<your-dns-name>** and confirm that the website appears.

Create the production deploy

With the staging workflow created, the next step is to create the production workflow. This step is simpler because you can copy the whole `deploy` job and just change its parameters.

1. In the **Code** view on the GitHub website, go to the **.github/workflows** directory. Select the **build-production.yaml** file and edit it.

2. Copy the `deploy` step from the previous pipeline and paste it below the last line of the YAML file.

   The result should look like this example:

YAML Copy

```yaml
name: Build and push the tagged build to production

on:
  push:
    tags:
      - 'v*'

jobs:
  build_push_image:
    runs-on: ubuntu-latest

    steps:
      - uses: actions/checkout@v2

      - name: Build and Push production image
        # You can pin to the exact commit or the version.
        # uses: docker/build-push-
action@ab83648e2e224cfeeab899e23b639660765c3a89
        uses: docker/build-push-action@v1.1.1
        with:
          username: ${{secrets.ACR_LOGIN}}
          password: ${{secrets.ACR_PASSWORD}}
          registry: ${{ secrets.ACR_NAME }}
          repository: contoso-website
          tag_with_ref: true

  deploy:
    runs-on: ubuntu-latest
    needs: build_push_image

    steps:
      - uses: actions/checkout@v2

      - name: Install Helm
        uses: Azure/setup-helm@v1
        with:
          version: v3.3.1

      - name: Get AKS Credentials
        uses: Azure/aks-set-context@v1
        with:
          creds: ${{ secrets.AZURE_CREDENTIALS }}
          # Resource group name
          resource-group: mslearn-gh-pipelines-6667
          # AKS cluster name
          cluster-name: contoso-video

      - name: Run Helm Deploy
        run: |
          helm upgrade \
            --install \
            --create-namespace \
            --atomic \
            --wait \
            --namespace staging \
            contoso-website \
            ./kubernetes/contoso-website \
            --set image.repository=${{ secrets.ACR_NAME }} \
            --set dns.name=${{ secrets.DNS_NAME }}
```

3. Change the `deploy` step to deploy to the production namespace. In the `Run Helm Deploy` step, change the `--namespace` flag from `staging` to `production`.

4. At the end of the Helm command, add a new `--set image.tag=${GITHUB_REF##*/}`.

   Here, you're using a Bash feature called *parameter expansion*. This feature is defined by the syntax `${ENV##<wildcard><character>}`. It returns the last occurrence of the string after `character`.

   In this case, you want to get the tag name. This variable is defined by the GitHub Actions runtime as `GITHUB_REF`. If it's a branch, it's also defined by `refs/heads/<branch>`. If it's a tag, it's defined by `refs/tags/<tag>`.

   We want to remove `refs/tags/` to get only the tag name, so run `${GITHUB_REF##*/}` to return everything after the last `/` in the `GITHUB_REF` environment variable.

   The final YAML file should look like this example:

   YAML

```yaml
name: Build and push the tagged build to production

on:
  push:
    tags:
      - 'v*'

jobs:
  build_push_image:
    runs-on: ubuntu-latest

    steps:
      - uses: actions/checkout@v2

      - name: Build and Push production image
        # You can pin to the exact commit or the version.
        # uses: docker/build-push-
action@ab83648e2e224cfeeab899e23b639660765c3a89
        uses: docker/build-push-action@v1.1.1
        with:
          username: ${{secrets.ACR_LOGIN}}
          password: ${{secrets.ACR_PASSWORD}}
          registry: ${{ secrets.ACR_NAME }}
          repository: contoso-website
          tag_with_ref: true

  deploy:
    runs-on: ubuntu-latest
    needs: build_push_image

    steps:
      - uses: actions/checkout@v2

      - name: Install Helm
        uses: Azure/setup-helm@v1
        with:
          version: v3.3.1

      - name: Get AKS Credentials
        uses: Azure/aks-set-context@v1
        with:
          creds: ${{ secrets.AZURE_CREDENTIALS }}
          # Resource group name
          resource-group: mslearn-gh-pipelines-6667
          # AKS cluster name
          cluster-name: contoso-video

      - name: Run Helm Deploy
        run: |
          helm upgrade \
            --install \
            --create-namespace \
            --atomic \
            --wait \
            --namespace production \
            contoso-website \
            ./kubernetes/contoso-website \
            --set image.repository=${{ secrets.ACR_NAME }} \
            --set dns.name=${{ secrets.DNS_NAME }} \
            --set image.tag=${GITHUB_REF##*/}
```

5. To commit the changes, select the green **Start commit** button. Enter a description for the commit, and then select **Commit new file**.

6. In Cloud Shell, run `git pull` to fetch the latest changes. Then, run the following command to tag and push the changes:

Bash Copy

```bash
git tag -a v1.0.1 -m'Creating first production deployment' && git push
```

```
--tags
```

7. Open the **Actions** tab and see the running process.

Test the deployment

To test the production deployment, go to **contoso-production.<your-dns-name>** in your browser and confirm that the website appears.

Next unit: Summary

Continue >