Admin Guide

A system administrator needs to be able to manage the Continuous Integration/Continuous Deployment (CI/CD) pipeline, and upgrade the application when necessary. Some of the tasks that a system administrator might want to do are listed below.

Prerequisites.

1. Login details for the development and production IBM cloud environments.
2. Installing the IBM cloud command line interface (CLI).
   Several of the steps below require using the IBM cloud CLI. Details for installing this can be found here: https://cloud.ibm.com/docs/cli?topic=cloud-cli-getting-started.
3. Installing kubectl.
   Kubectl is required to run commands against Kubernetes Clusters. It is required for several of the steps below, and details for how to install it can be found here: https://kubernetes.io/docs/tasks/tools/install-kubectl/.

Trigger a deployment to production.

A brief overview of the Continuous Integration portion of our pipeline is as follows: There are two main branches, master and develop. We are following a gitflow workflow, so feature branches are cut from develop, worked on locally and then merged back into develop. Changes to any branch will trigger a deployment of that branch to develop, however only changes to the master branch will trigger a deployment to the production environment. As such, whenever one wants to start a deployment to production the steps are as follows:

- Cut a release branch from develop, following the format “Version<Major change>.<Minor change>”.
- Create a Pull Request (PR) to master, which will have to be approved by another member of the team.
- Creating the release branch will trigger a deployment to develop of that branch, so before it is merged you can check that it properly deploys to develop.
- Once the PR is approved, merge it to master, triggering a deployment to the production environment.

Check the status of the Develop/Production Cloud Registries.

The IBM Cloud Registry (CR) service is where we are storing containers of our application. When using the free tier of this service, there are restrictions on storage (how many containers we can store) and pull traffic (how many containers we can pull from the

CR). Every time a new deployment is made, the relevant container is pulled from the CR, so the pull traffic restriction is essentially a limit on deployments per month.

- Log into develop/production CR, depending on which one you're checking.
  ```ibmcloud login```
  This will require the email and password of the account.
- Check the CR is logged into the correct region (Frankfurt for develop, Dallas for production)
  ```ibmcloud cr region```
  If the region is not correct, you can set it using:
  ```ibmcloud cr region-set``` and selecting the desired region.
- List the images in the current CR
  ```ibmcloud cr images```
- Check the quotas for the current CR
  ```ibmcloud cr quota```

Modify/Update the CR image retention policy.

There is a storage limit for the free tier of the IBM CR service. As of writing this is 500MB. Depending on the size of the containers being stored here, this will limit the amount of containers we can store. Currently our docker images are ~10MB, although this is most likely to increase slightly in the future. Since each new build results in a new image being stored, eventually we will reach our storage limit.

IBM cloud provides a solution for this with their image retention policies. Details of this can be found here: https://cloud.ibm.com/docs/Registry?topic=registry-registry_retention.

To list the current retention policies:

- Log into develop/production CR, depending on which one you're checking.
  ```ibmcloud login```
  This will require the email and password of the account.
- List the retention policies.
  ```ic cr retention-policy-list```

To set and run a retention policy:

- Log into develop/production CR, depending on which one you're checking.
  ```ibmcloud login```
  This will require the email and password of the account.
- Set a retention policy. This will retain only the given amount of images per repository in the given namespace. It will run this policy immediately and then daily after this.
  ```ibmcloud cr retention-policy-set --images <image_count> sweng-devops```, where <image_count> is the amount of images to be kept.
  ```

Check the status of the Develop/Production Kubernetes Clusters.

This can be done either through the IBM cloud user interface, or by using the IBM cloud CLI.

To check using the IBM cloud UI:

- Navigate to the IBM cloud login page: https://cloud.ibm.com/login
- Login in using the email and password of either the develop or production account.
- Once logged in, the easiest way to find the kubernetes clusters is to navigate to https://cloud.ibm.com/kubernetes/clusters using your browser. This brings you to a page listing all current kubernetes clusters.
- Depending on which deployment environment has been logged into, there should be a “prod_cluster” or a “develop_cluster”.
- Clicking on the cluster will bring you to an overview of that cluster, with information regarding whether the cluster is working normally.

To check using the IBM CLI:

- Log into develop/production cloud acocunt, depending on which one you’re checking.
  ```ibmcloud login```
  This will require the email and password of the account.
- To get an overview of the cluster, replace <cluster_name> with either “prod_cluster” or “develop_cluster” in the follow command:
  ``ibmcloud ks cluster get --cluster <cluster_name>```

Check the color of the current deployment.

We are currently using a Blue/Green deployment strategy, so it may be required at some point to see whether the current deployment is a blue or a green one. This can be done by checking the Travis CI logs of the last successful build, or through the IBM cloud CLI.

To check using the IBM CLI:

- Log into develop/production cloud account, depending on which one you’re checking.
  ```ibmcloud login```
  This will require the email and password of the account.
- In order to execute commands against our cluster using `kubectl` we need to download the Kubernetes configuration files and certificates.
  ```ibmcloud ks cluster config --cluster <cluster_name>```, where <cluster_name> is either “prod_cluster” or “develop_cluster”.
- We now need to extract the deployment from the kubernetes service which exposes our application.
  ``` kubectl get service sweng-devops -o json | jq -r '.spec.selector.deployment' ```

To check the Travis CI logs:

- Navigate to https://travis-ci.com/github/christophernixon/DevOps-Pipeline-sweng/branches in

your browser and select the branch which you want to check (develop for develop deployments and master for production deployments).
- Select the last successful build on that branch.
- Once on the build, navigate to the “Deploy to develop” or “Deploy to production” stages, depending on which deployment environment you want to check.
- Scroll to the bottom of the logs, and expand the “Deploying application” line.
- Below this line will be the logs of the last deployment, the log output colors alone should indicate which color the last one was.

Roll-back to previous deployment.

Since we are using a blue-green deployment strategy, this allows for quick and easy switching between two production deployment environments. This might be required if, for example, something goes wrong with a production deployment so it’s required to roll back to the previous production deployment.

- Log into the production cloud account.
  ```ibmcloud login```
  This will require the email and password of the account.
- In order to execute commands against our cluster using `kubectl` we need to download the Kubernetes configuration files and certificates.
  ```ibmcloud ks cluster config --cluster prod_cluster```
- Check whether it is a blue or green deployment currently
  ``` kubectl get service sweng-devops -o json | jq -r '.spec.selector.deployment' ```
- If current deployment is blue:
  ```DEPLOYMENT=green \
  envsubst < service.yml | kubectl apply -f -```
- Else if current deployment is green:
  ```DEPLOYMENT=blue \
  envsubst < service.yml | kubectl apply -f -```

This will switch the deployment that the service which exposes our application is pointing at from blue to green, or visa versa. These commands can also be run for the develop environment, by logging into it and replacing “prod_cluster” with “develop_cluster”.

## Check the endpoints for current Develop/Production deployments.

The endpoints of our deployments are in the format “<cluster IP>:<node port>”, where cluster IP is the public IP address of the cluster, and node port is the port on which the application is exposed by our kubernetes service. Since the cluster IP is static for that cluster and the node port is set by the `service.yml` file, these endpoints should remain static. It may however be required to discover the endpoint of the deployments at some point, for example if the cluster needs to be re-built at some point.

- Log into develop/production cloud account, depending on which one you’re checking.
  ```ibmcloud login```
  This will require the email and password of the account.

- In order to execute commands against our cluster using `kubectl` we need to download the Kubernetes configuration files and certificates.
  ```ibmcloud ks cluster config --cluster <cluster_name>```, where <cluster_name> is either “prod_cluster” or “develop_cluster”.
- Retrieve the public IP of the cluster.
  ``` public_ip=$(ibmcloud ks workers --cluster <cluster_name> --json -s | jq -r '.[0].publicIP') ```, where <cluster_name> is either “prod_cluster” or “develop_cluster”.
- Retrieve the NodePort of the service exposing the application.
  ``` nodeport=$(kubectl get service sweng-devops -o json | jq -r '.spec.ports[0].nodePort') ```
- Print out the endpoint.
  ```echo "The application can now be viewed at http://$public_ip:$nodeport/\n" ```

## Add another stage to the pipeline.

It may be necessary, at some point, to add another stage to the pipeline. An example would be if one wanted to add end-to-end testing on production deployments, automatically rolling back to previous production if testing failed. The pipeline stages are defined in the `.travis.yml` file, located at the root of the project. Helpful Travis CI documentation on Build Stages can be found here: https://docs.travis-ci.com/user/build-stages/ .

To add another stage:

- Add the stage name under the “stages” field. This field dictates the order stages are run in, and the running of stages can be conditioned on the branch if necessary.
- Under the “jobs.include” field, one can add the scripts to be run during the stage. The format can be modelled on the existing stages.