**Kubernetes Migration Step by Step**

The high-level steps for migration plan include

1. Making the App Kubernetes Ready
2. Establish the CI/CD process
3. Prepare Kubernetes Cluster.

**Making App Kubernetes Ready**

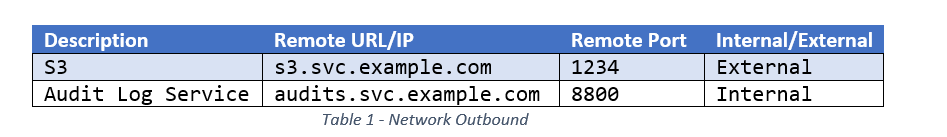
1. Review the current architecture with the various components involved and consider the components to be migrated
2. Consider the communications options between the components whether message queues, service buses or API calls.
3. Rethinking some components might be helpful but keeping to the current architecture and focusing on the migration helps not complicating the journey.
4. Containerize the application using ready to use Docker images for many frameworks and application runtimes. So many examples and documents on how to Dockerize your component for each modern programming language.
5. Map to K8’s objects, choose from the list of options provided by Kubernetes to host the components including deployments, services, endpoints, jobs and other components. The best practice is to come up with a K8s architecture choosing the objects for your components.
6. Next step is to create the Kubernetes objects for your components. Writing YAML files for Kubernetes deployment, services, pods, jobs and other components. Creating a separate repository for these configurations utilizing GitOps increasing the reliability and security of our K8’s resources.
7. Most cases leave the database as is and connect to the database from the Kubernetes application.

**Gather information about the system**

**Determine Network Interactions**

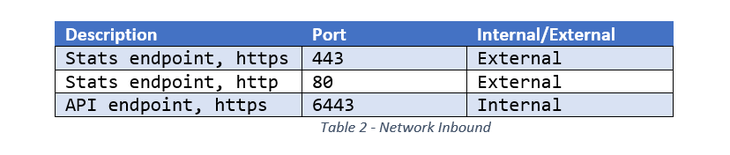
1. What must the service connect to inorder to function?

Make sure to record the URL or IP, the port and the location of the resource within or outside the cluster.



1. What services or users need access to this service?

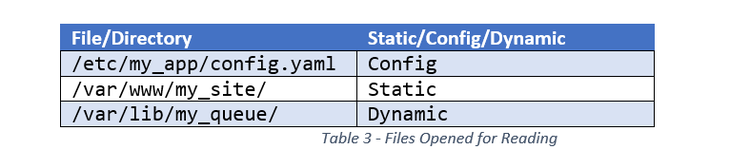
Make sure to record the port and location of the remote user/service inside the cluster and outside the cluster or both.



**Determine File Interactions**

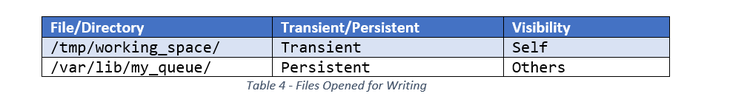
**What files and directories are read by the service ?**

For each file or directory determine whether the content is static, configuration or dynamic



**What files are modified by the service ?**

Make sure to record whether the modifications should survive service restarts i.e to be persistent or transient. Also record whether the changes must be visible to other applications or executables.



**Use the data to make a suitable decision**

After gathering the above information, it gives a deep understanding of the application which can be used to make a decision whether to migrate the application or not. Once it deemed that the migration is worth the effort the migration process involves

1. Containerization of the processes that make up the application
2. Selecting the Kubernetes Objects that will make up the components of the application in the new environment.

**How to containerize the applications?**

**Build the container image**

Docker builds images by reading instructions from a Docker file. A docker file is a simple text file that contains instructions that can be executed on the command line. Using command line, the docker build we can start a build that executes all of the command line

**Select an Azure Container Registry**

Create an Azure container registry to build and store the images.

**Deploy the containerized app on AKS**

Once the image is built it is required to deploy the application as a container on AKS (Azure Kubernetes Service).

1. Select the Azure Kubernetes Services Cluster

Specify the AKS cluster that the application should be deployed to. The selected AKS cluster must have a Linux node pool.

The cluster has to be configured to allow pulling of the images from the Azure Container Registry that was selected to store the images.

Run the following command in Azure CLI to attach the AKS cluster to the ACR.

az aks update -n <cluster-name> -g <cluster-resource-group> --attach-acr <acr-name>

1. Specify secret store and monitoring workspace: If you had opted to parameterize application configurations, then specify the secret store to be used for the application. You can choose Azure Key Vault or Kubernetes secrets for managing your application secrets.

Monitoring Workspace: If you’d selected enabled monitoring with Application Insights, then specify the application insights.