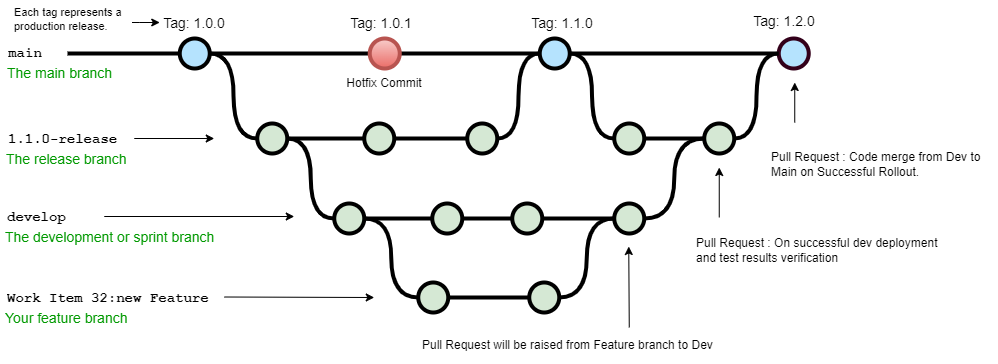
## Branching strategy:

Trunk based Development.



.

We will have dev and release branches.

Dev Branch

The source code in the develop branch reaches a stable point and is ready to be released, all of the changes should be merged back into master and then tagged with a release number.

#### Release Branch

Must merge back into: dev and master.

Any Hot fix, code changes related to the release

## 

## Azure Pipeline

#### Build Pipeline

Azure DevOps build pipeline will have the following task

Enable continuous integration trigger on the Master Branch.

The Hosted Agent will spin up and analyze the pipeline and checkout the source code.

Install docker in Agent and run the docker file

Docker Engine:

* Install maven
* copy source code
* Maven Clean, compile, Run Unit Test, Package
* Install tomcat
* deploy package to tomcat.

Docker Build Task:

Build a container image and tag it.

Release Pipeline

Azure DevOps release pipeline will have the following task

Create service connection for ACR.

Create Docker Registry Connection.

Provide access to the pipeline using service connection.

Push the container image inside the azure container registry.

ACR managed service will scan for vulnerabilities in image.

Publish kubernetes manifests file to pipeline workspace.

Enable trigger on release pipeline for continuous deployment.

Configure AKS Dev Environment (Cluster & Namespace).

Create Service Connection for AKS.

Provide access to the pipeline using service connection.

Create Image pull secret from AKS to ACR.

Download manifest file.

Deploy downloaded manifest files

Deployment.yaml :

Pods with container images from ACR will be created with a mentioned number of replicas.

Service.yaml:

Expose the Pods using load balance service to the external network.

Deploy container image to AKS

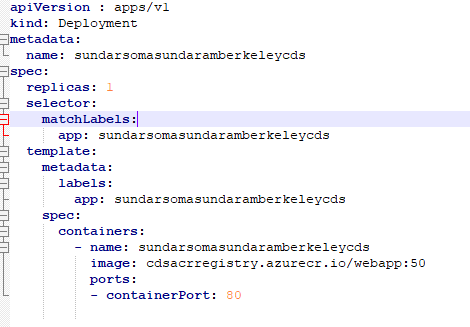
Pull the image from ACR using the secret created.

Production Deployment strategy: Canary deployment model

Since application need to be tested in the production environment before complete rollout, this model will deploy x % pods in production.

Kubernetes manifests

deployment yaml



Deployment.yaml : Declarative way of defining desired state of *pod* and *replica set*.

Pod : it contains one or more application containers which are relatively tightly coupled.

Replicas: Desired number of pods

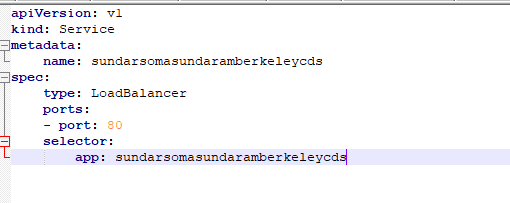
Selector: Deployment finds which Pods to manage using matchLabels.

Spec : Container and other specification related to the deployment

Image: container image to be pulled from

Port: Container running port

service yaml



Service.yaml : expose an application running on a set of [Pods](https://kubernetes.io/docs/concepts/workloads/pods/) as a network service.

Selector: Service uses the selector to identify the Pods.

Type: Service will be exposed as LoadBalancer/ClusterIP/NodePort.

Deployment Strategy

#### Blue Green Deployment

We can test the application in an environment replica of production before rolling out. So we have Blue Green deployment, where we will have a staging environment.

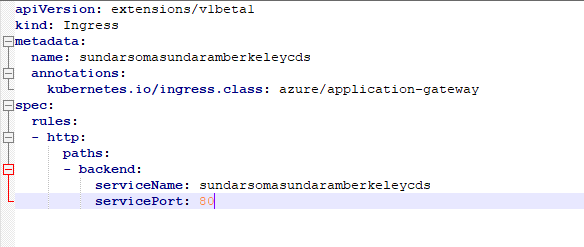
* Application will be deployed to staging environment (new Pods)
* Test the features/changes
* Switch the application service to Prod

Canary Deployment

* Test new features of the application using this strategy.
* Managed by a service through selectors and labels.
* If testing fails, point back service to the old version of deployment.

Exposing to public access

We are using a Public load balancer and it will be exposed to the public using Azure Application Gateway.



Output

