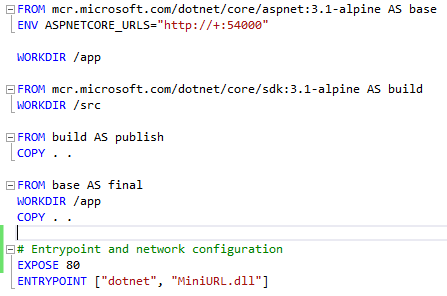
**CI/CD pipeline configuration**

Pre-requisites

* Azure DevOps
* Azure Container Registry
* Azure K8S Service
* Docker Desktop

Steps

1. Add ***Dockerfile*** and ***dockerignore*** files into the Web project.
2. Update ***Dockerfile*** with steps to create a docker image as below.



1. Test if build image is created properly using: “***docker build -t {{image-name}} .***” in local.
2. Add a folder *azure-pipelines* and create a yaml file eg. ***miniurl-pipeline.yaml***
3. Add some yaml code to define trigger, resources, stages (Build, Development, Stage, Production)
   1. For actual code refer the project folder shared as git repository.
4. Now the pipeline code is ready to setup in Azure DevOps.
5. In Azure DevOps portal navigate to ***Pipelines*** -> ***New Pipeline*** -> ***Azure Repos Git*** -> ***{{Code-Repository}}*** -> ***Existing Azure Pipelines YAML file*** -> Select path to ***miniurl-pipeline.yaml*** *->* ***Review*** *and* ***Create****.*
6. In Azure DevOps portal navigate to ***Library*** and create a ***variable group*** for each environment.
7. ***Add permission*** for all pipeline to access these new variable groups.
8. ***Assign security roles***, who should be able to add/update variables.
9. Now we should ***add approvers*** for the pipeline to deploy on Stage and production K8S service. This will prevent unnecessary deployments to sensitive environments.
10. In Azure DevOps portal navigate to ***Environments*** -> ***{{Environment}}*** and under the 3 dots on top right corner of the page select ***Approvals and checks****.*
11. Add the list of approvers who will be responsible to approve any deployment to these environments.
12. Once ***domain URL*** pointing is completed to the service running in Azure K8S Service.
13. Browse the URL and see the API working.

**Deployment Plan**

Steps

1. QA testing, security testing and bug fixing.
2. QA approval.
3. Rollout plan and assigning ownership.
4. Decide minimal impacting time for release.
5. Release details/intention (rollout plan, rollback plan, ownership, impact, QA approval, release date and time) approvals by IT verticals.
6. Ahead of time schedule a bridge with all concerning people/teams.
7. Rollout changes and monitoring.
8. Timely progress updates over the bridge chat/email.
9. Any issue, execute rollback plan (mostly running old stable production pipeline run).
10. Finding the gap and apply the changes if feasible, else re-plan the release.
11. Final status of the release over the bridge chat/email.
12. Sending release notes and appreciation to participants.

**Requirements Review**

* Sensitive values should come directly from KeyVault and stored into a class with singleton scope. Avoid storing sensitive details in appsettings file. Values such as Encryption Key, Connection String and any sensitive credentials.
* Decision of using RijndaelManaged has a gap, as this type is now obsolete, use AES instead.
* Encryption is predictable, first 6 characters will be same for same protocol. For all URLs starting with http will have same 6 characters, same applies for https and ftp.
* Decryption is not possible unless we take complete encrypted string. Hence, we cannot get original URL back from first 6 encrypted characters.