Azure DevOps: Managing Settings on a Per-Branch Basis

When using Azure DevOps Pipelines for CI/CD, it's convenient to leverage the built-in pipeline variables for secrets management, but using pipeline variables for secrets management has its disadvantages:

- *Pipeline variables are managed outside the code that references them.* This makes it easy to introduce drift between the source code and the secrets, e.g. adding a reference to a new secret in code but forgetting to add it to the pipeline variables (leads to confusing build breaks), or deleting a reference to a secret in code and forgetting to remote it from the pipeline variables (leads to confusing pipeline variables).

- *Pipeline variables are global shared state.* This can lead to confusing situations and hard to debug problems when developers make concurrent changes to the pipeline variables which may override each other. Having a single global set of pipeline variables also makes it impossible for secrets to vary per environment (e.g. when using a branch-based deployment model where 'master' deploys using the production secrets, 'development' deploys using the staging secrets, and so forth).

A solution to these limitations is to manage secrets in the Git repository jointly with the project's source code. As described in secrets management, don't check secrets into the repository in plain text. Instead we can add an encrypted version of our secrets to the repository and enable our CI/CD agents and developers to decrypt the secrets for local usage with some pre-shared key. This gives us the best of both worlds: a secure storage for secrets as well as side-by-side management of secrets and code.

```
# first, make sure that we never commit our plain text secrets and generate a
strong encryption key
echo ".env" >> .gitignore
ENCRYPTION_KEY="$(LC_ALL=C < /dev/urandom tr -dc '_A-Z-a-z-0-9' | head -c128)"

# now let's add some secret to our .env file
echo "MY_SECRET=..." >> .env

# also update our secrets documentation file
cat >> .env.template <<< "
# enter description of your secret here
MY_SECRET=
"
```

```
# next, encrypt the plain text secrets; the resulting .env.enc file can safely be
committed to the repository
echo "${ENCRYPTION_KEY}" | openssl enc -aes-256-cbc -md sha512 -pass stdin -in
.env -out .env.enc
git add .env.enc .env.template
git commit -m "Update secrets"
```

When running the CI/CD, the build server can now access the secrets by decrypting them. E.g. for Azure DevOps, configure `ENCRYPTION_KEY` as a secret pipeline variable and then add the following step to `azure-pipelines.yml`:

```
steps:
  - script: echo "$(ENCRYPTION_KEY)" | openssl enc -aes-256-cbc -md sha512 -pass
stdin -in .env.enc -out .env -d
    displayName: Decrypt secrets
```

You can also use variable groups linked directly to Azure key vault for your pipelines to manage all secrets in one location.

Last update: August 26, 2024