LU12a - Jenkins Secrets

Anstatt Secrets direkt in die Jenkinsfile-Datei zu schreiben, wo es für alle mit Lesezugriff auf das Code-Repository lesbar wäre, können auf Jenkins „secrets“ hinterlegt werden. Die verschiedenen Typen von „secrets“ sind in folgender Tabelle aufgelistet.

Typ	**Zweck**	**Beispiele**	**Verwendung in Pipeline**
Secret text	Einzelner String-Wert	API Keys, Tokens	`withCredentials([string(...)])`
Username/Password	Login-Paare	DB, Docker Registry	`withCredentials([usernamePassword(...)])`
SSH Key	SSH Zugriff	Server Deploy, Git SSH	`sshUserPrivateKey`
Secret file	Datei als Secret	kubeconfig, JSON Keys, Zertifikate	`file(credentialsId, ...)`
Certificate	X.509 Zertifikate	mTLS, interne APIs	Zertifikatsbasierte Auth (zum Beispiel für Webhooks)

Nachfolgend ist die Verwendung in einem Jenkinsfile zu sehen. Die separate Helper-Function sorgt dafür, dass bei mehreren Stages nicht die Secret-ID mehrfach angegeben werden muss.

Jenkinsfile mit Passwort-Leak ❌

```
environment {
        DB_USER             = "appuser"
        DB_PASSWORD         = "apppassword"
    }
    ...
            steps {
                sh """
                    docker run -d \
                        --name $DB_CONTAINER \
                        --restart unless-
stopped \
                        --network infra-net \
                        -e
POSTGRES_USER=$DB_USER \
                        -e
POSTGRES_PASSWORD=$DB_PASSWORD \
                        -e POSTGRES_DB=$DB_NAME
\
                        -v
$DB_VOLUME:/var/lib/postgresql/data \
                        -v
$WORKSPACE/database:/docker-entrypoint-initdb.d
\
                        postgres:17
                """
            }
```

Jenkinsfile mit Secret-Helper-Function ✔

```
def withDbCredentials(body) {
    withCredentials([
        usernamePassword(
            credentialsId: 'postgres-creds',
            usernameVariable: 'DB_USER',
            passwordVariable: 'DB_PASSWORD'
        )
    ]) {
        body()
    }
}
...
            steps {
                withDbCredentials {
                    sh """
                        docker run -d \
                            --name
$DB_CONTAINER \
                            --restart unless-
stopped \
                            --network infra-net
\
                            -e
POSTGRES_USER=$DB_USER \
                            -e
POSTGRES_PASSWORD=$DB_PASSWORD \
                            -e
POSTGRES_DB=$DB_NAME \
                            -v
$DB_VOLUME:/var/lib/postgresql/data \
                            -v
$WORKSPACE/database:/docker-entrypoint-initdb.d
\
                            postgres:17
                    """
                }
            }
```

Ein wichtiger Grundsatz bezüglich der Sicherheit von Jenkins-Secrets lautet: „Wenn ein Benutzer eine Pipeline editieren kann, kann er in der Regel auch Secrets exfiltrieren.“