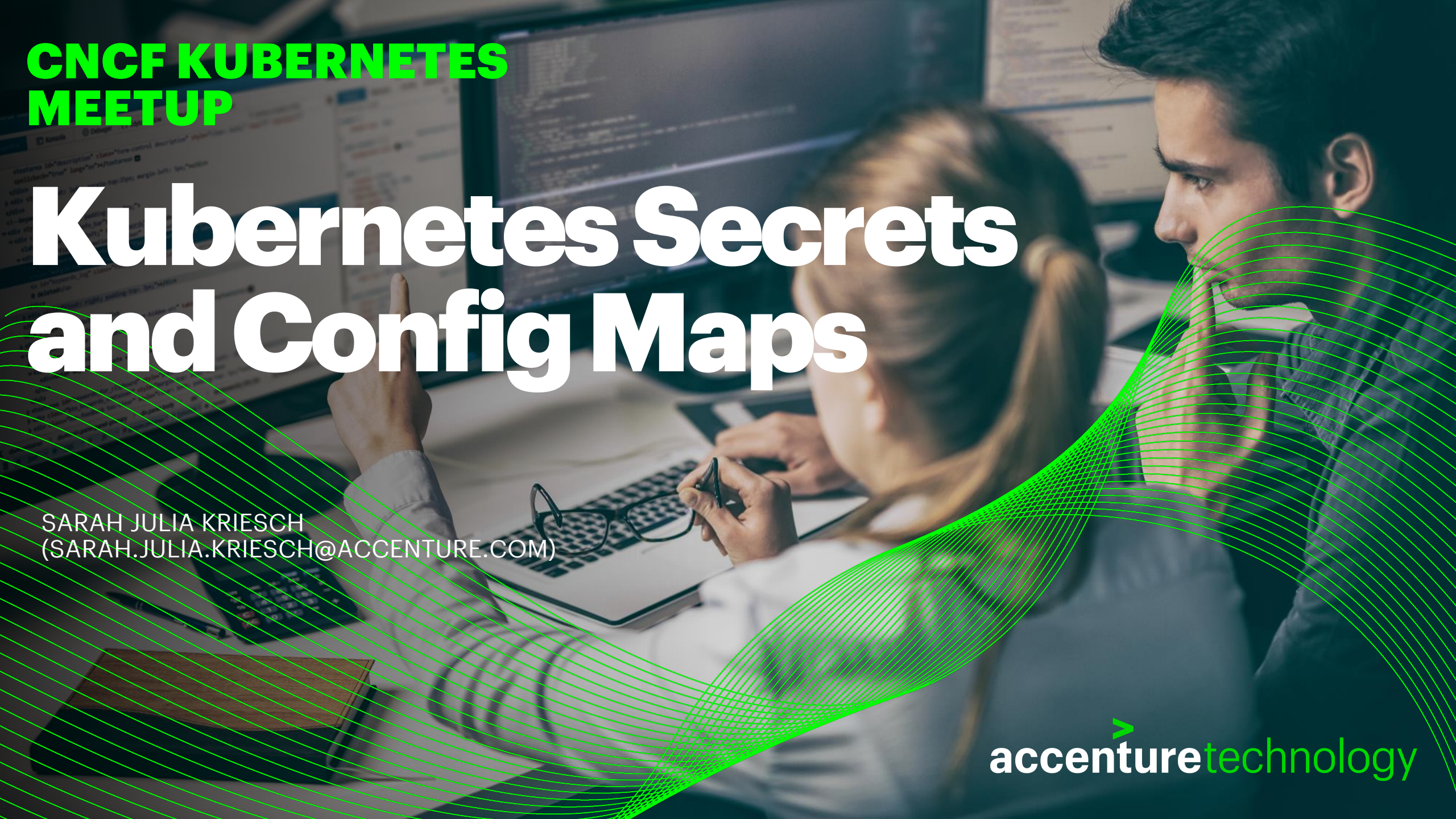
CNCF KUBERNETES MEETUP

Kubernetes Secrets and Config Maps



SARAH JULIA KRIESCH
(SARAH.JULIA.KRIESCH@ACCENTURE.COM)

accenture technology

AGENDA

01 Introducing Kubernetes Secrets

02 Methods to encrypt private data

03 How to integrate these data

04 How to verify and decrypt these data

KUBERNETES SECRETS IN GENERAL

INTRODUCTION
WHY KUBERNETES SECRETS

- Credentials and private data should be kept encrypted
- Customer data should not be stored anywhere outside of the used system
- Keep your yaml files in your Kubernetes cluster clean
- Easy integration with key-value pairs

INTRODUCTION
KUBERNETES SECRETS VERSUS CONFIGMAPS

Kubernetes Secrets
- For confidential data
- Using Base64 encoding

ConfigMaps
- For non-confidential data

Consistent:
- key-value pairs
- Stored as env variables

TYPES OF SECRETS

Opaque	arbitrary user-defined data
kubernetes.io/service-account-token	service account token
kubernetes.io/dockercfg	serialized ~/.dockercfg file
kubernetes.io/dockerconfigjson	serialized ~/.docker/config.json file
kubernetes.io/basic-auth	credentials for basic authentication
kubernetes.io/ssh-auth	credentials for SSH authentication
kubernetes.io/tls	data for a TLS client or server
bootstrap.kubernetes.io/token	bootstrap token data

DIFFERENCE BETWEEN OPAQUE AND BASIC-AUTH

Opaque:

- Default secret type
- Own definition of variables possible

```yaml
apiVersion: v1
kind: Secret
metadata:
  name: mysecret
type: Opaque
data:
  username: YWRtaW4=
  password: MWYyZDFlMmU2N2Rm
```

Basic authentication:

- For storing credentials
- Username and password required
- Base64 and text possible

```yaml
apiVersion: v1
kind: Secret
metadata:
  name: secret-basic-auth
type: kubernetes.io/basic-auth
stringData:
  username: admin
  password: t0p-Secret
```

ENCRYPTION

BASE64 VIA THE COMMAND LINE

```
$ echo –n `admin´ | base64
YWRtaW4=

$ echo –n `1f2d1e2e67df` | base64
MWYyZDF1MmU2N2Rm
```

HTPASSWD FOR USER AUTHENTIFICATION (NGINX/INGRESS)

$ htpasswd –c auth user1 #auth name of the file and user1 the user

Create basic-auth secrets based on htpasswd file
$ kubectl create secret generic basic-auth --from-file=auth

```
$ kubectl get secret basic-auth -o yaml
apiVersion: v1
data:
  auth: Zm9vOiRhcHIxJE9GRzNYeWJwJGNrTDBGSERBa29YWUlsSDkuY3lzVDAK
kind: Secret
metadata:
  name: basic-auth
  namespace: default
type: Opaque
```

SECRET GENERATOR KUSTOMIZE

- **Kubernetes secrets generator (secretGenerator) and ConfigMap Generator (configMapGenerator)**
- **kustomization.yaml as a base file**
- **Reading from files and literals possible**

- **Usage of Kustomize:**
- **$ kubectl kustomize <kustomization_directory>**
- **$ kubectl apply –k <kustomization_directory>**

KUSTOMIZE BASED ON FILES

```
# Create a password.txt file
cat <<EOF >./password.txt
username=admin
password=secret
EOF

cat <<EOF >./kustomization.yaml
secretGenerator:
- name: example-secret-1
  files:
  - password.txt
EOF
```

$ kubectl get secret example-secret-1

```
apiVersion: v1
data:
  password.txt: dXNlcm5hbWU9YWRtaW4KcGFzc3dvcmQ9c2VjcmV0Cg==
kind: Secret
metadata:
  name: example-secret-1-t2kt65hgtb
type: Opaque
```

KUSTOMIZE BASED ON LITERALS

- **Text in kustomization.yaml will be encypted**

```
cat <<EOF >./kustomization.yaml
secretGenerator:
- name: example-secret-2
  literals:
  - username=admin
  - password=secret
EOF
```

$ kubectl get secret example-secret-2

```
apiVersion: v1
data:
  password: c2VjcmV0
  username: YWRtaW4=
kind: Secret
metadata:
  name: example-secret-2-t52t6g96d8
type: Opaque
```

KUBESEAL

- **cli tool for Kubernetes based Sealed Secrets**
- **The Sealed Secrets Controller will decrypt any Sealed Secret to a Kubernetes Secret**
- **Kubeseal is a client for Linux/MacOS by Bitnami**

- **Installation:**
- **brew (MacOS), snap, OBS (openSUSE Tumbleweed), other available Linux packages**
- **Helm:**
  - helm repo add sealed-secrets https://bitnami-labs.github.io/sealed-secrets
  - helm install sealed-secrets -n kube-system --set-string fullnameOverride=sealed-secrets-controller sealed-secrets/sealed-secrets

KUBESEAL USAGE

secret.yaml

```
apiVersion: v1
kind: Secret
metadata:
  creationTimestamp: null
  name: my-secret
data:
  password: dmRGcmU4R21Mcw==
  username: YWRtaW4=
```

cat secret.yaml | kubeseal --controller-namespace kube-system \ --controller-name sealed-secrets-controller \ --format yaml \ > sealed-secret.yaml

```
kind: SealedSecret
metadata:
  creationTimestamp: null
  name: my-secret
  namespace: default
spec:
  encryptedData:
    password: AgCCzOHDzdnhExDiE5Zph4BLgy7P6EUQVeL/LJk4Jzi/nBAeP0HOV3cWCdOdYr/THPvg4d1UQcGRqgEEaSyGiS5Z
yGlz2kWu+MkfLnMBNHS4N43I0TmRzoOFXEbwtVfUB34BHKSoyca1aZEbTnF5ySqjGq8NztZIZlfEdw1SZ2r96i/EGMRBMOtY/V1wX6
xe1ITD8Op3gQ2VUVO5QVR4mtQoGhqHAn0vQSaTkBfelaP0Ar0oh3aB4kf1rx0+h9q8btDMZuJGyC05T7P5xYtJ0iBM2gE83IIJjOvy
NFdQF3sRHux9T4vKDdXTUFHAinYit8S/6m8J6qvrgUCMpid0qhw7NQsMfxesefFhcAceKPuXxjCQPVYtIKqV0ZeNoORi9Csz22jvGb
f7x61dE/sV9eyZNdTLJJUKmdfeI6H07L+JZJkUeHwORDCMPCv0i9dWrW2VvdqFCOO1bE1Yqj1gJFx4hYN89D7O4OHomTK6SSkJcOYT
1LriS8oeHmSwxQoRp4DBfPPvvQYKY/hc3o/yuBeOEIcd9wbaeQbBn3vkks9wGYjQgg6nB4StGjvJPD+v3dP1Dr9G/PBjHkiWuyOM82
BW3N+0jE8m3lKEJblVBtQ8B2JflUUUFObMb3ZRdOzm/cH0fuWLkPHROkUG9HvSZh6Dmyc5nC9YigpbPLFG8xSZZomR1f5/sLpoFvLf
L6A4zFsIjzTbBcgIonrE
    username: AgAEwxTqVUVx9YFTIxnqSZqHXKvMf8s1eztH369ZhUD3Jf90aBHAjYIZFcs8jFLhWQZSG/NQttAa7rZx0I/RdJpw
erJn5WZvWp7iTvsMuo/v16fMdWoMBfISDGiTLRIQjnQ3iCzpBBqLYJS/dED0tax0wYhNq24+N8t+x5UF/JeAHnDvBt5Mp7Z7TUJFSK
Kjt+J4FsaJNnosmEoZFrctHstnZ2tFJ56P6lhs1hY53kANE2DbtHrGPE1qhUhWHYoDTbsIwNvoqxB/4eXo83mztBv3twlmcZBDv1P/
UdUC/7UTrjqs4sxOwvY434fXDtovmIkcxIVuBSG9xLl6kVOrZi9FXEm2qcOM3+PlgegsA0ZlZyddV3YeqiwBvjOMBcNH4GN5TKapVJ
bftsixqVuB40i5hZVHnEsiDSgL4Uq5JkuWy7q9no73/RVv76d722+vd2otxYstjX8Q+i7kDimAoA32DzJskejLvpUyPePsXrIkknU+
YXellQEYDN6BODiJMcKiLFSPBV47AcE6NVLIlVRhosSCys1crWYDs3YZPzHTeCzwa4O4N6O940QUh3qXK15GWyXOZPjiqJc+mWa20D
82Vwbq0sJvy2Lgg5YK9ZijP3AwBcC2hxgyACwvOP9wdXOvGX9NwOh2LKaSzaEvwjxn1Tj7ySeeWP/KX9b3qnxGZe6xmXbSsJCnF/6Y
unTCN5kH0czmDw==
  template:
    data: null
    metadata:
      creationTimestamp: null
      name: my-secret
      namespace: default
```

INTEGRATION

POD.YAML WITH SECRETS STORED AT A SPECIAL PLACE

Secret is stored under /etc/foo/my-group/my-username

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: mypod
spec:
  containers:
  - name: mypod
    image: redis
    volumeMounts:
    - name: foo
      mountPath: "/etc/foo"
      readOnly: true
  volumes:
  - name: foo
    secret:
      secretName: mysecret
      items:
      - key: username
        path: my-group/my-username
```

POD.YAML WITH REFERENCES TO SECRETS

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: envvars-multiple-secrets
spec:
  containers:
  - name: envars-test-container
    image: nginx
    env:
    - name: BACKEND_USERNAME
      valueFrom:
        secretKeyRef:
          name: backend-user
          key: backend-username
    - name: DB_USERNAME
      valueFrom:
        secretKeyRef:
          name: db-user
          key: db-username
```

- Pod.yaml is using the backend-username from the secret backend-user for SECRET_USERNAME
- Pod.yaml is using the db-username from the db-user for DB_USERNAME

USING SECRETS IN VALUES.YAML IN HELM CHARTS
MYSECRETS.YAML

- **kubectl apply –f mysecrets.yaml**

```
# file: mysecrets.yaml
apiVersion: v1
kind: Secret
metadata:
  name: mysecrets
type: Opaque
data:
  oidc: dG9wLVNlY3JldA==
  jdbc: dG9wLVNlY3JldA==
```

USING SECRETS IN VALUES.YAML IN HELM CHARTS
DEFINITION OF ENV VARIABLES IN VALUES.YAML

- **env_secret.yaml**

```
# file: env_secret.yaml
env:
# OIDC authentication:
- name: vvp.auth.oidc.registration.clientSecret
  valueFrom:
    secretKeyRef:
      name: mysecrets
      key: oidc
# JDBC persistence:
- name: spring.datasource.password
  valueFrom:
    secretKeyRef:
      name: mysecrets
      key: jdbc
```

USING SECRETS IN VALUES.YAML IN HELM CHARTS
REMOVE PASSWORDS IN VALUES.YAML AND APPLY

- helm install/upgrade . –values values.yaml –values env_secret.yaml

DECRYPTION

VERIFICATION

$ kubectl get secret my-secret –o jsonpath= {.data}
{„password“: „MWYyZDF1MmU2N2Rm“, „username“:“YWRtaW4=„}

DECODING

```
$ echo `MWYyZDF1MmU2N2Rm` | base64 –decode
1f2d1e2e67df
```

EDITING

EDIT A KUBERNETES SECRET

- **Work with Base64**
- **$ kubectl edit secrets mysecret**

DELETION

DELETION OF SECRETS

$ kubectl delete secret mysecret

AGENDA

01 Usage of Config Maps

02 Config Maps based on a directory

03 Config Maps with Kustomize

CONFIG MAPS

USAGE OF CONFIG MAPS

- **Definition of volumes data**
- **Definition of properties**
- **Definition of environment variables**

CONFIG MAPS BASED ON DATA IN DIRECTORY

```
sarah.julia.kriesch@AMAC02G29GGMD6M properties % cat  database.properties
postgresql.enabled=true
postgresql.global.postgresql.postgresqlDatabase=gitea
postgresql.global.postgresql.servicePort=5432
postgresql.pesistence.size=10Gi
```

```
sarah.julia.kriesch@AMAC02G29GGMD6M properties % cat volumes.definition
volumes.name=gitea-database
volumes.configMap.name=gitea-config
volumeMounts.name=postgresql
volumeMounts.mountPath=/gitea-postgresql
```

$ kubectl create configmap gitea-config --from-file=./properties/
configmap/gitea-config created

YAML OUTPUT OF THE CONFIG MAP

$ kubectl get configmaps gitea-config -o yaml

```
apiVersion: v1
data:
  database.properties: |
    postgresql.enabled=true
    postgresql.global.postgresql.postgresqlDatabase=gitea
    postgresql.global.postgresql.servicePort=5432
    postgresql.pesistence.size=10Gi
  volumes.definition: |
    volumes.name=gitea-database
    volumes.configMap.name=gitea-config
    volumeMounts.name=postgresql
    volumeMounts.mountPath=/gitea-postgresql
kind: ConfigMap
metadata:
  creationTimestamp: "2022-09-13T10:35:47Z"
  name: gitea-config
  namespace: default
  resourceVersion: "57982"
  uid: d5d6e6ec-3504-4fec-bddc-69e1b3f75e82
```

$kubectl describe configmaps gitea-config

```
Name:         gitea-config
Namespace:    default
Labels:       <none>
Annotations:  <none>

Data
====
database.properties:
----
postgresql.enabled=true
postgresql.global.postgresql.postgresqlDatabase=gitea
postgresql.global.postgresql.servicePort=5432
postgresql.pesistence.size=10Gi

volumes.definition:
----
volumes.name=gitea-database
volumes.configMap.name=gitea-config
volumeMounts.name=postgresql
volumeMounts.mountPath=/gitea-postgresql


BinaryData
====

Events:  <none>
```

CONFIG MAPS WITH KUSTOMIZE

```
cat <<EOF  >./kustomization.yaml
configMapGenerator:
- name: gitea-config-1
  files:
  -    properties/database.properties
  -    properties/volumes.definitions
  EOF
```

```
$ kubectl apply –k  .
```

INTEGRATION OF CONFIG MAPS REFERENCES IN YAML

```
env:
  - name: database
    valueFrom:
       configMapKeyRef:
         name: gitea-config
         key: gitea-database
```

#name of the value in this file

#name of the Config Map
#name of the value (key) in the Config Map

DELETION OF CONFIG MAPS

$ kubectl delete configmap gitea-config

REFERENCES

- https://kubernetes.io/docs/concepts/configuration/secret/
- https://opensource.com/article/19/6/introduction-kubernetes-secrets-and-configmaps
- https://kubernetes.io/docs/tasks/manage-kubernetes-objects/kustomization/
- https://ververica.zendesk.com/hc/en-us/articles/360015021359-How-to-secure-passwords-secrets-in-values-yaml-with-Kubernetes-secrets
- https://kubernetes.io/docs/tasks/configure-pod-container/configure-pod-configmap/
- https://matthewpalmer.net/kubernetes-app-developer/articles/ultimate-configmap-guide-kubernetes.html
- https://docs.openshift.com/container-platform/4.11/applications/config-maps.html

UN-CONFERENCES AND MEETUPS

23.-25. September 2022 DevOps Camp

https://devops-camp.de

24. September 2022 GDG Nürnberg DevFest

https://eventbrite.de/e/devfest-2022-gdg-nuernberg-flutter-cloud-conference-tickets-403361433907

Q&A

LICENSE

Except where otherwise noted, this work is licensed under

Text: https://creativecommons.org/licenses/by/4.0/
Examples: https://opensource.org/licenses/Apache-2.0