SECRETS IN KUBERNETES

JERRY JALAVA

Senior System Architect, Google
Developer Expert, Authorised Trainer

JERRY@QVIK.FI | @W_I

Google Developers Experts

QVIK

QUESTION

WHAT SECRETS DO APPLICATIONS HAVE?

COMMON SECRETS

- Database credentials
- API credentials & endpoints (Twitter, FB, etc.)
- Infrastructure API credentials (Google, AWS, Azure)
- Private (TLS) Keys (careful here)
- Etc.

Photos: CC0 License, https://www.pexels.com

WELL, EASY ENOUGH…

I'LL JUST INCLUDE THEM WITH MY CODE…

DON’T!

JUST AS AN EXAMPLE

Dev put AWS keys on Github.
Bots are crawling all over GitHub seeking secret keys...

"When I woke up the next morning, I had emails and a missed phone call from Amazon AWS - ~140 servers running on my AWS account"
- a developer served with a $2,375 Bitcoin mining bill

http://www.theregister.co.uk/

WELL, EASY ENOUGH…

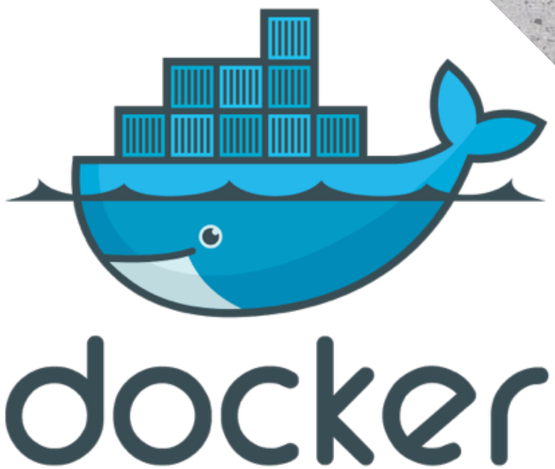
I'LL JUST INCLUDE THEM IN MY CONTAINER…

AGAIN, DON'T!

DOCKER

is about transportable and executable code.
Containers can be inspected, exported and published publicly.

Putting sensitive material inside Docker Container is **NOT** a good idea.



THE TWELVE-FACTOR APP

- http://12factor.net/
- Methodology for building software-as-a-service apps, collection of best practises
- Suggests following:
  - Store *config* in the environment
  - *config* is everything that is likely to vary between deploys

KUBERNETES (K8s)

- Ancient Greek for “pilot” or “helmsman”; root of the English word “governor”
- Orchestrator for containers
- Supports multi-cloud environments
- Started by Google
- Open source
- Manage applications, not machines

MAIN COMPONENTS

PODS

Ephemeral *units used to manage 1-n tightly coupled containers*

LABELS

Metadata *attached to objects such as Pods. Enable organization and selection of objects*

REPLICATION CONTROLLERS

Manages requested number of Pod "replicas" from defined template

SERVICES

Low overhead load-balancing of requests to set of Pods based on Labels

INTRODUCING

KUBERNETES SECRETS

http://kubernetes.io/docs/user-guide/secrets/

WHAT ARE THEY?

- K8s Secret-objects are first-class citizens in the ecosystem
- Designed to hold all kinds of sensitive information in safe and flexible way.
- They can be used by Pods (FS & Env) and the underlying kubelet when pulling images

```yaml
apiVersion: v1
kind: Secret
metadata:
  name: mysecret
type: Opaque
data:
  password: MWYyZDFlMmU2N2RmCg==
  username: YWRtaW4K
```

```yaml
kind: Pod
metadata:
  name: secret-env-pod
spec:
  containers:
    - name: mycontainer
      image: redis
      env:
        - name: SECRET_USERNAME
          valueFrom:
            secretKeyRef:
              name: mysecret
              key: username
```

```yaml
spec:
  containers:
    - name: mycontainer
      image: redis
      volumeMounts:
        - name: "secrets"
          mountPath: "/etc/my-secrets"
          readOnly: true
  volumes:
    - name: "secrets"
      secret:
        secretName: "mysecret"
```

PROS

- Secrets can be mounted as data volumes or be exposed as environment variables to be used by a container in a pod
- A secret is only sent to a node if a pod on that node requires it
- Secret data on nodes is stored in tmpfs volumes and thus does not come to rest on the node
- Communication between user to the api-server, and from api-server to the kubelets, is protected by SSL/TLS

CONS

- In the API server secret data is stored as plaintext in *etcd*, therefore:
  - Administrators should limit access to *etcd* to admin users
  - Secret data in the API server is at rest on the disk that *etcd* uses (wipe/shred disks when not used)
- It is not possible currently to control which users of a K8s cluster can access a secret (Support planned)
- It is still possible to **accidentally** push the Secrets definition to version control

GOTCHAS

- A Secret needs to be created before any pods that depend on it
- Individual secrets are limited to 1MB in size
- They can only be referenced by pods in same namespace
- Once a pod is started, its secret volumes will not change, even if the secret resource is modified
- It is not possible currently to check what resource version of a secret object was used when a pod was created (planned feature)
- The key must be formatted as DNS subdomain (leading dots allowed), with max length of 253 characters

EXAMPLES

- *echo "admin" > ./username.txt && echo "1f2d1e2e67df" > ./password.txt*
  - kubectl create secret generic db-user-pass --from-file=./username.txt --from-file=./password.txt
- *echo "admin" | base64*
  - kubectl create -f ./mysecret.yaml

```
apiVersion: v1
kind: Secret
metadata:
  name: mysecret
type: Opaque
data:
  password: MWYyZDFlMmU2N2RmCg==
  username: YWRtaW4K
```

DEMO TIME

https://github.com/jerryjj/devsec_050416/blob/master/demo/secrets.md

INTRODUCING

KUBERNETES CONFIGMAP

http://kubernetes.io/docs/user-guide/configmap/

WHAT ARE THEY?

- key-value pairs of configuration data
- Similar to Secrets, but designed to more conveniently support working with strings that do not contain sensitive information
- Can be used to store fine-grained information like individual properties or coarse-grained information like entire config files or JSON blobs

DEMO TIME

https://github.com/jerryjj/devsec_050416/blob/master/demo/configmaps.md

RESOURCES

- https://github.com/jerryjj/devsec_050416
- http://kubernetes.io/docs/
- http://12factor.net/

QVIK

THANK YOU

www.qvik.fi

QVIK CREATES MEANINGFUL SERVICES! WOULD YOU LIKE TO JOIN US?