

Secrets in the Cloud

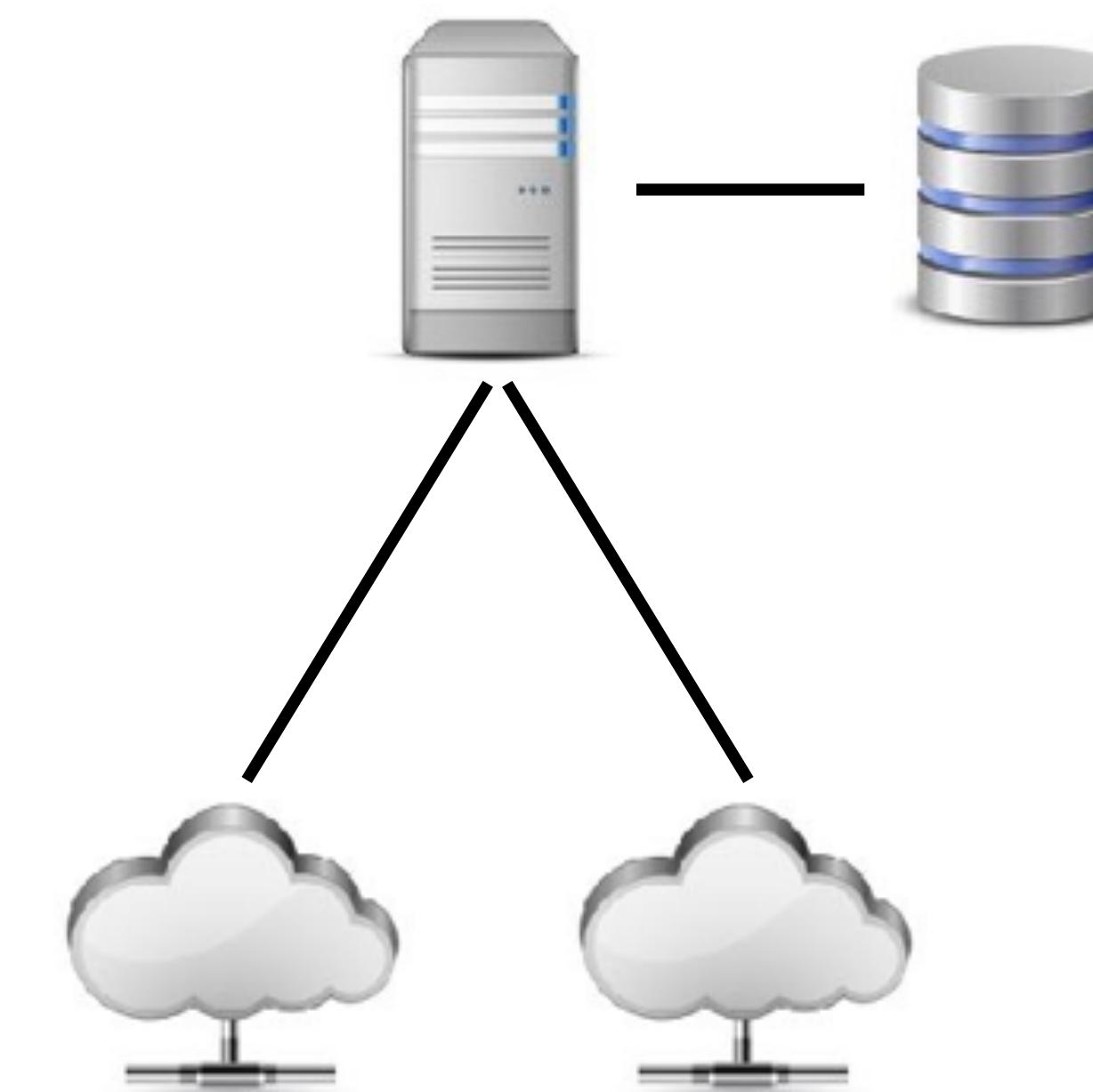
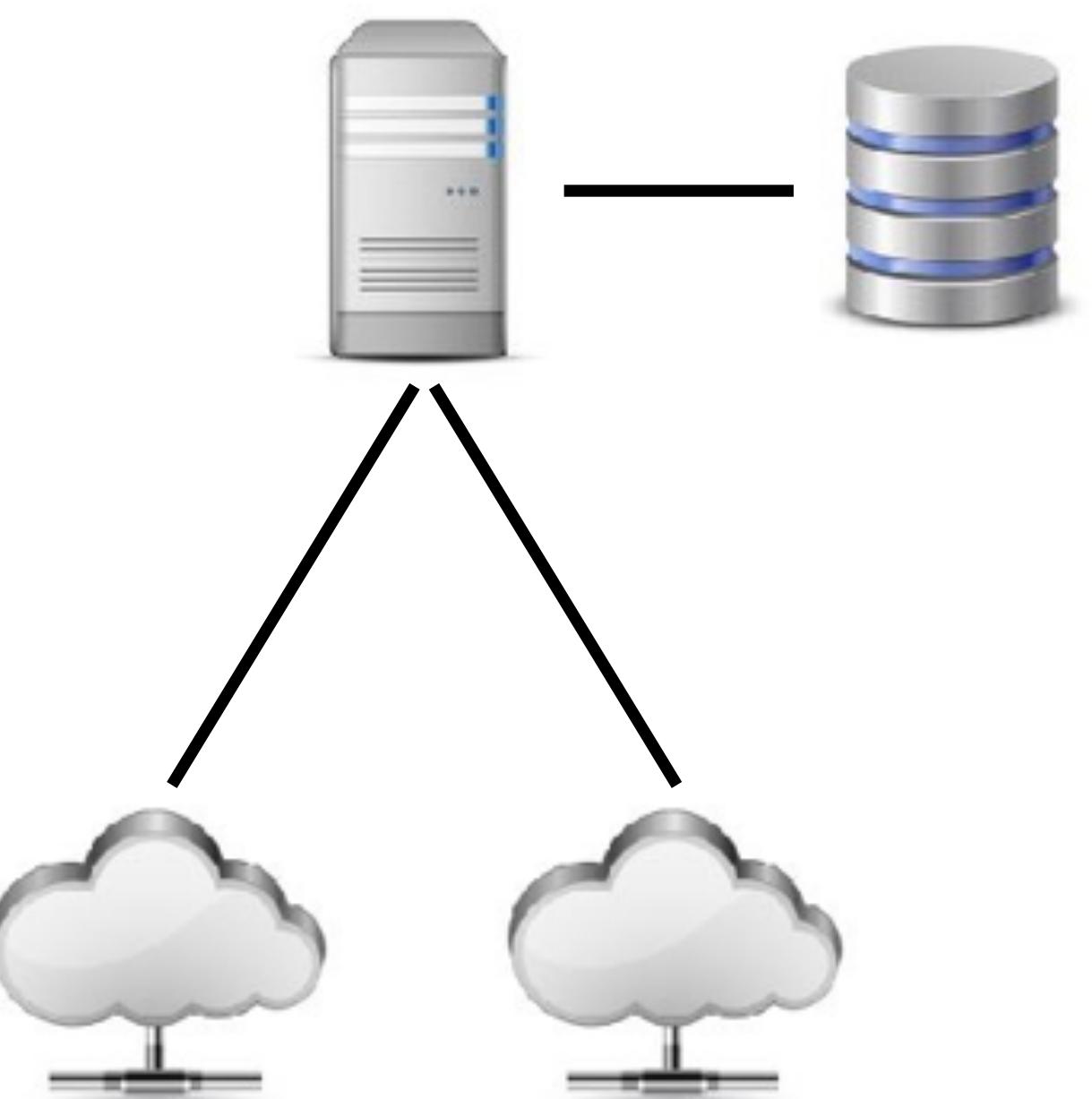
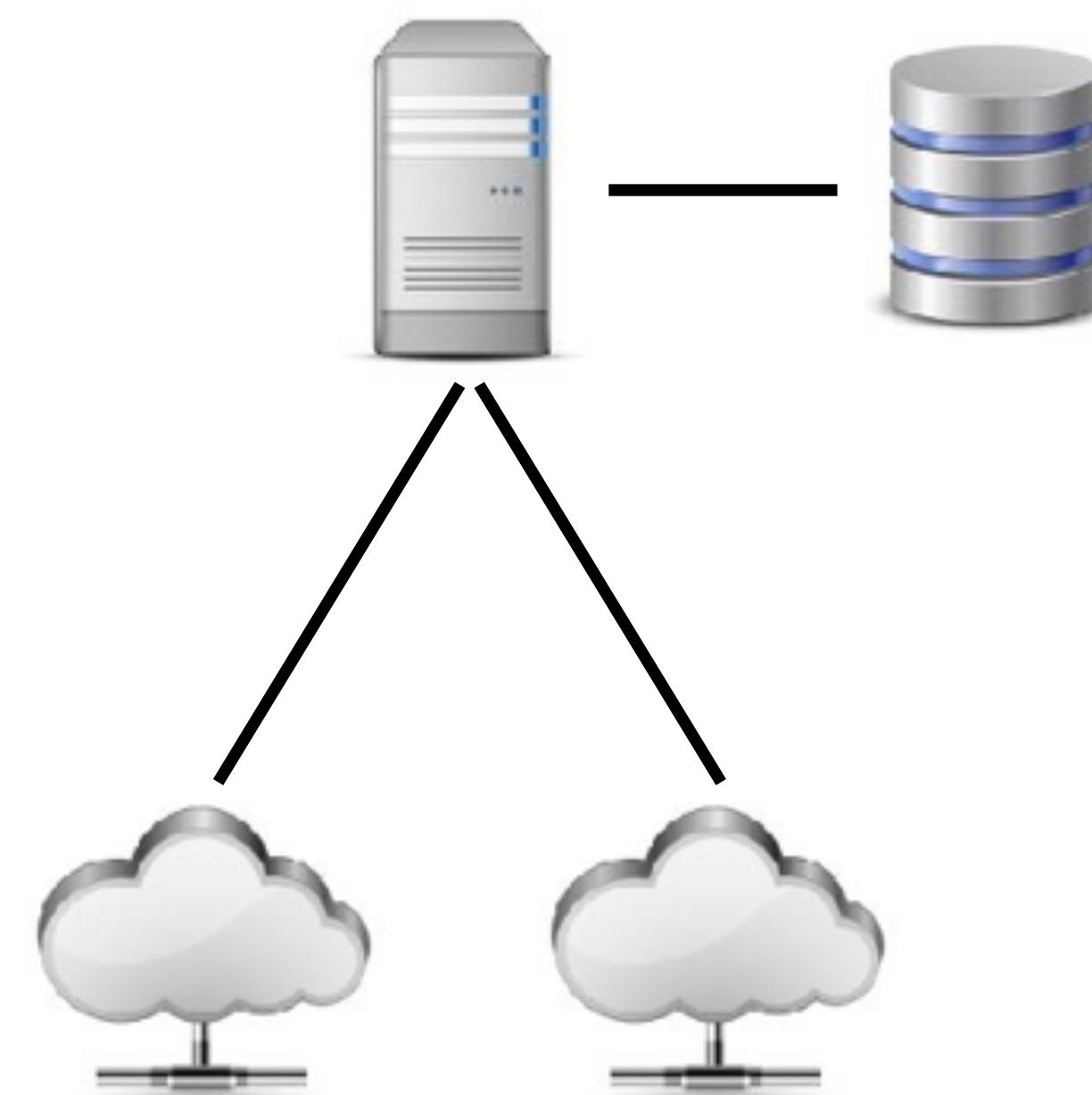
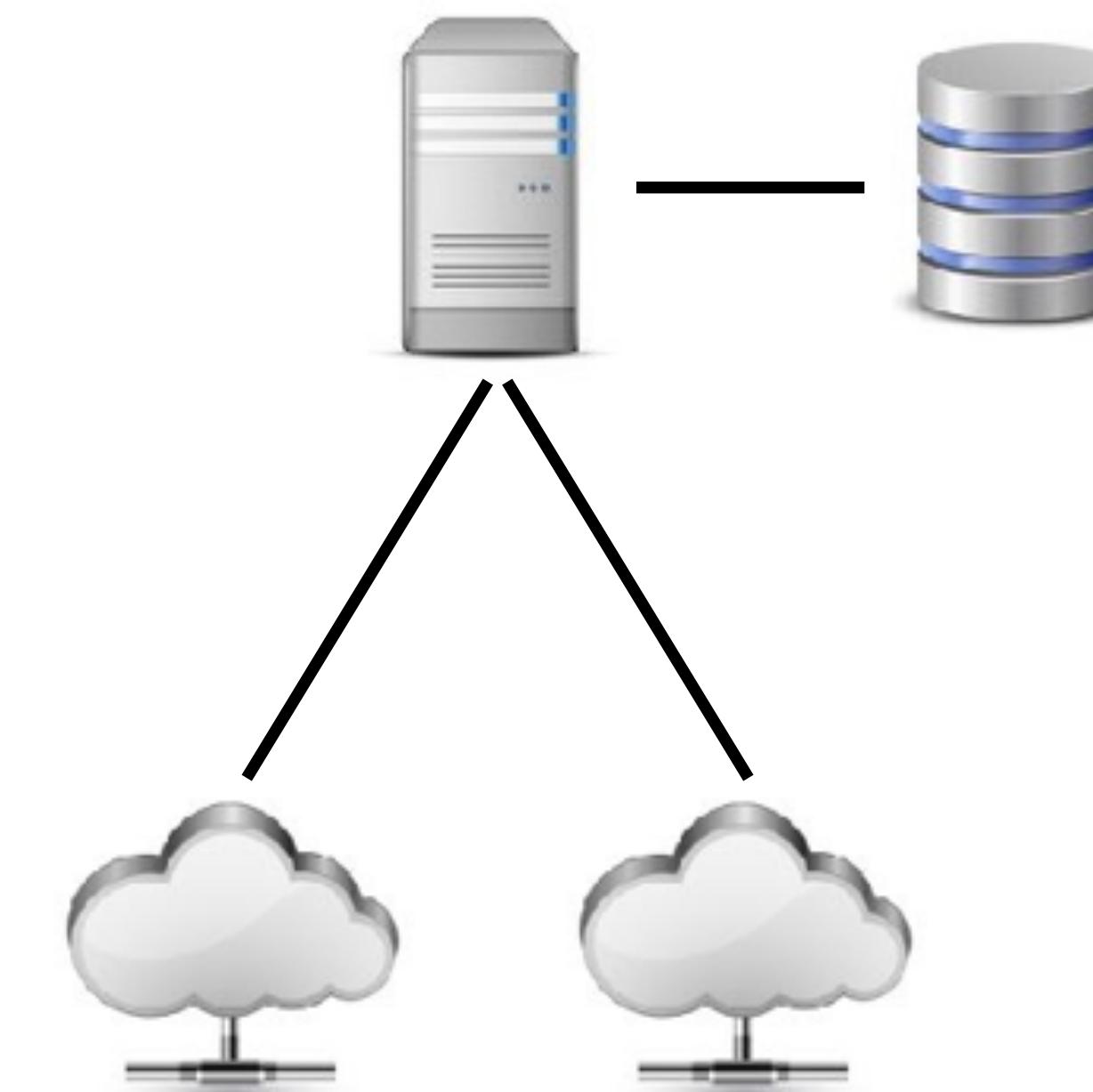
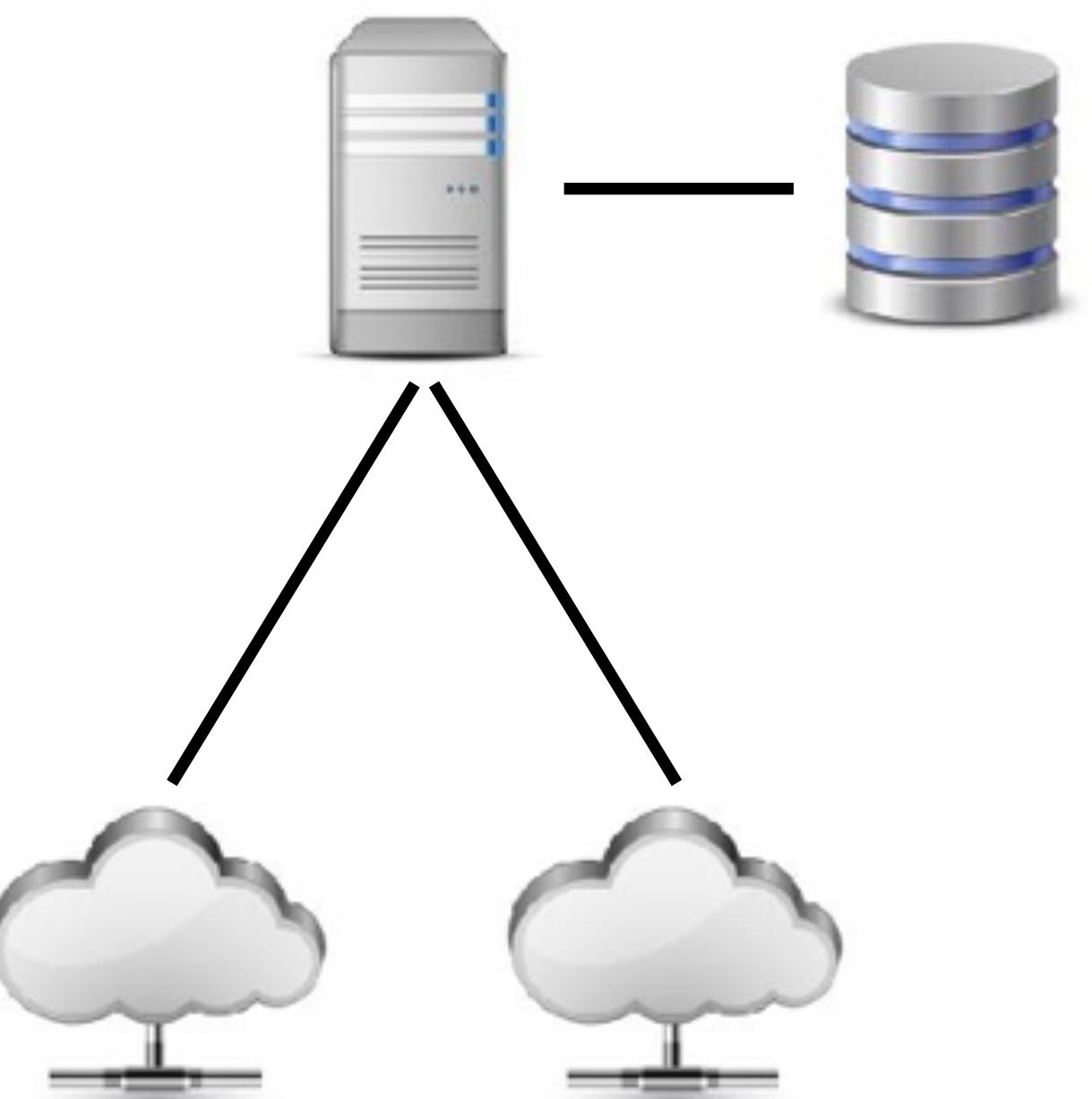
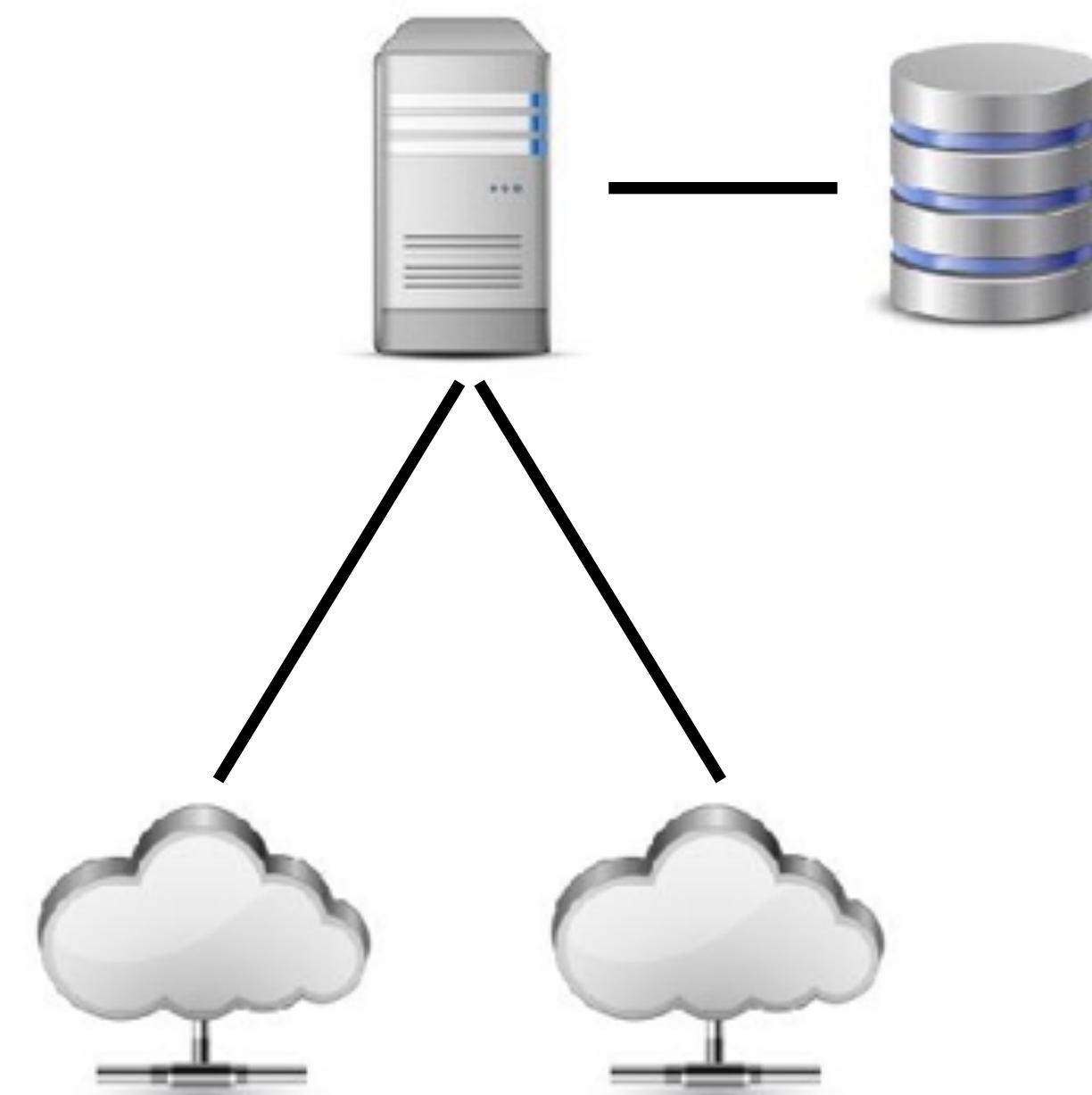
JAX 2017



Dominik Schadow
bridgingIT

```
spring:  
  datasource:  
    username: myDatabaseUser  
    password: mySuperSecretDatabasePassword
```

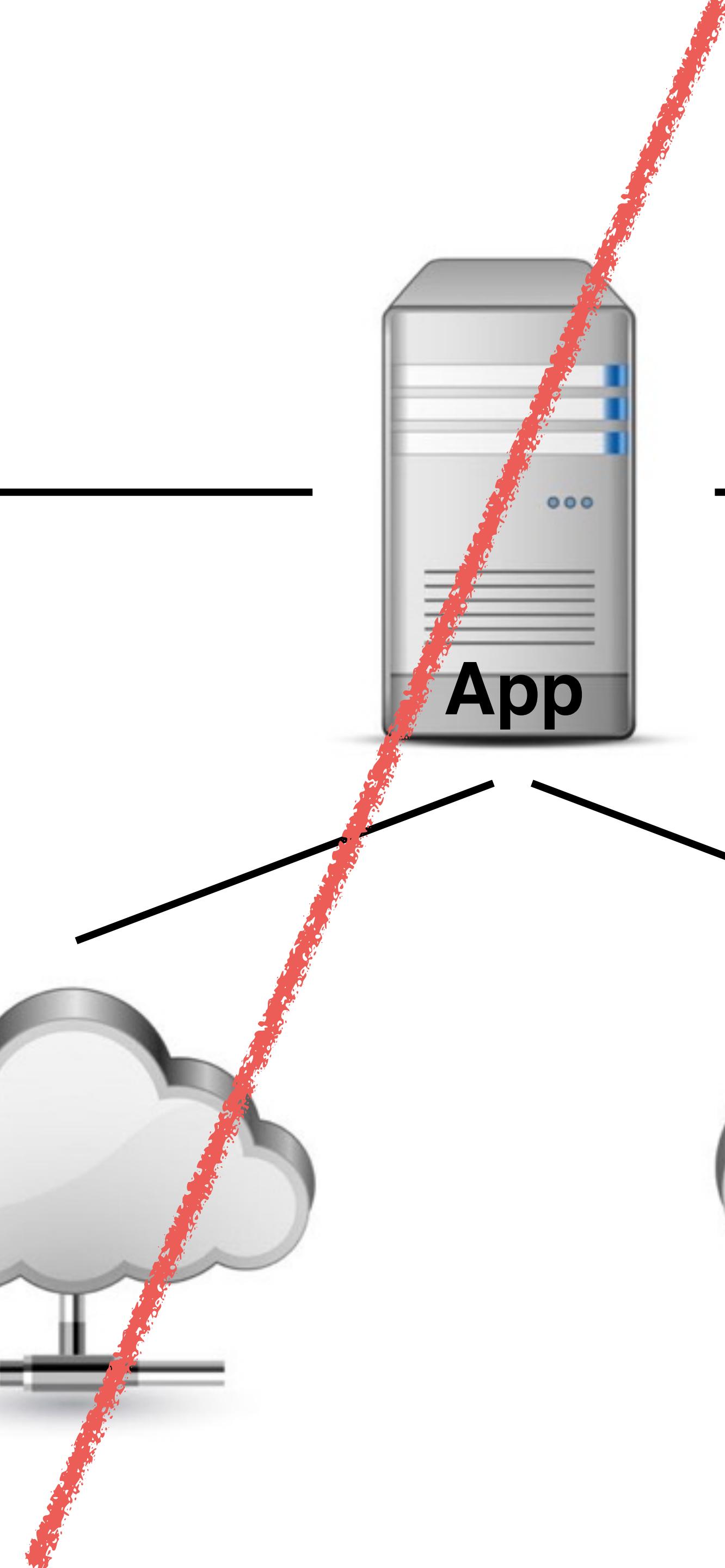
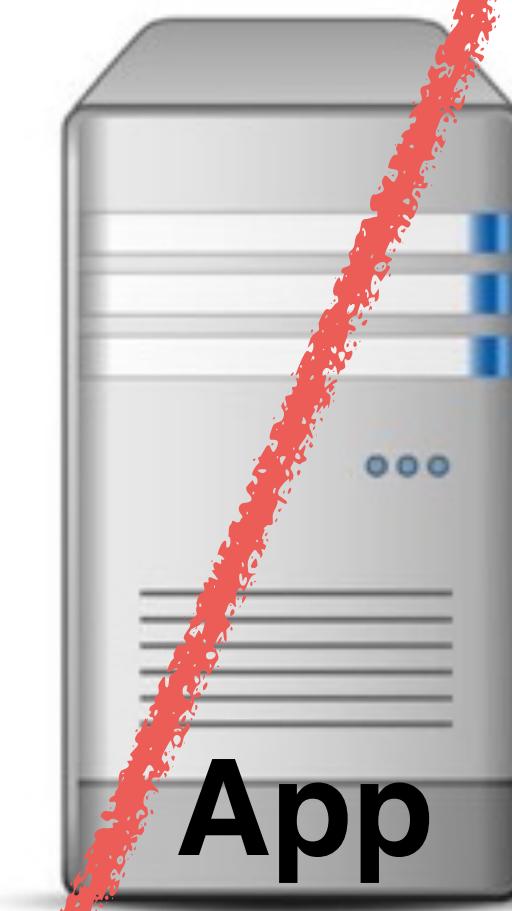
ID	USERNAME	PASSWORD	SECRET_ID	SECRET_DATA
1001	Arthur	kvgkIu7ZuPIdK9G7WUA duTvd9TinwRlvA6foux mgxM ZwUsPUdW6	42	Secret
1002	Zaphod	wC28772M7AYVwLe2BOu dF18VBo59KS5H1MbY9i rizpQhP6KCd33	42	Secret
1003	Slarti	eHkuCs817pYySnk0aKl zDeZDCSiUSedCOABqCE sRVYzS1Uc8RzK	42	Secret
1004	Ford	3kQkFnjyt008yrIjnjf tZewS6j8yKIbywJYzvs 3HOGqtfYcAVV0	42	Secret



DEV



PROD



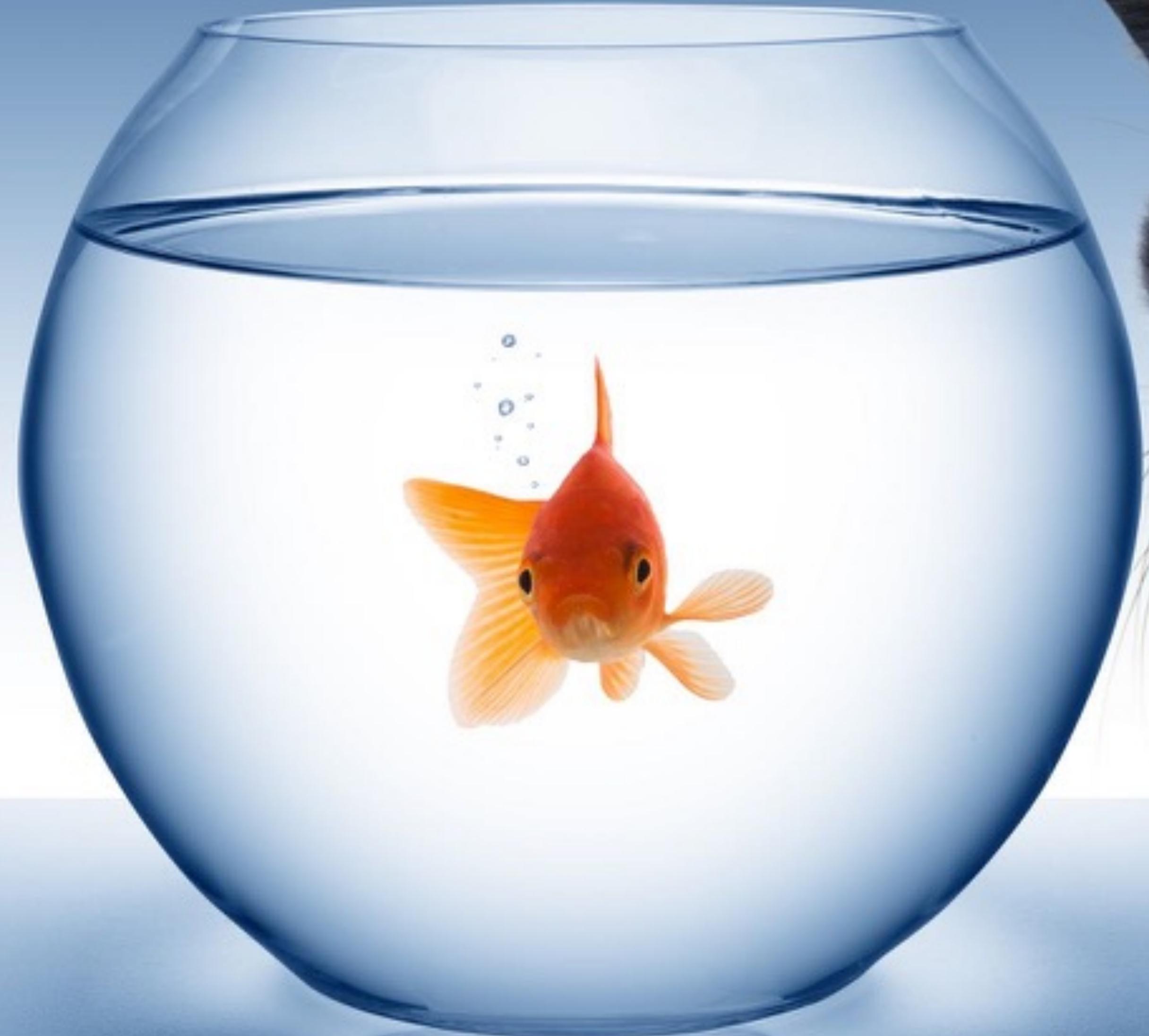


**Store
technical
and personal
secrets securely
in the cloud**



Technical Credentials

Embedded Configuration



```
<dependency>
  <groupId>com.github.ulisesbocchio</groupId>
  <artifactId>jasypt-spring-boot-starter</artifactId>
  <version>1.12</version>
</dependency>
```

```
$ encrypt.sh input="mySuperSecretDatabasePassword" password="sample-password"
```

```
-----ENVIRONMENT-----
```

```
Runtime: Oracle Corporation Java HotSpot(TM) 64-Bit Server VM 25.131-b11
```

```
-----ARGUMENTS-----
```

```
input: mySuperSecretDatabasePassword  
password: sample-password
```

```
-----OUTPUT-----
```

```
vvnkG2pj//Jd1vXe7YtuLvyFCtKBA+CVOYAT9qwfB4yuv4jngb6r/g==
```

```
spring:  
  datasource:  
    username: myDatabaseUser  
    password: ENC(vvnkG2pj//Jd1vXe7YtuLvyFCtKBA+CVOYAT9qwfB4yuv4jngb6r/g==)
```

jasypt.encryptor.password

System property Command line argument Environment variable

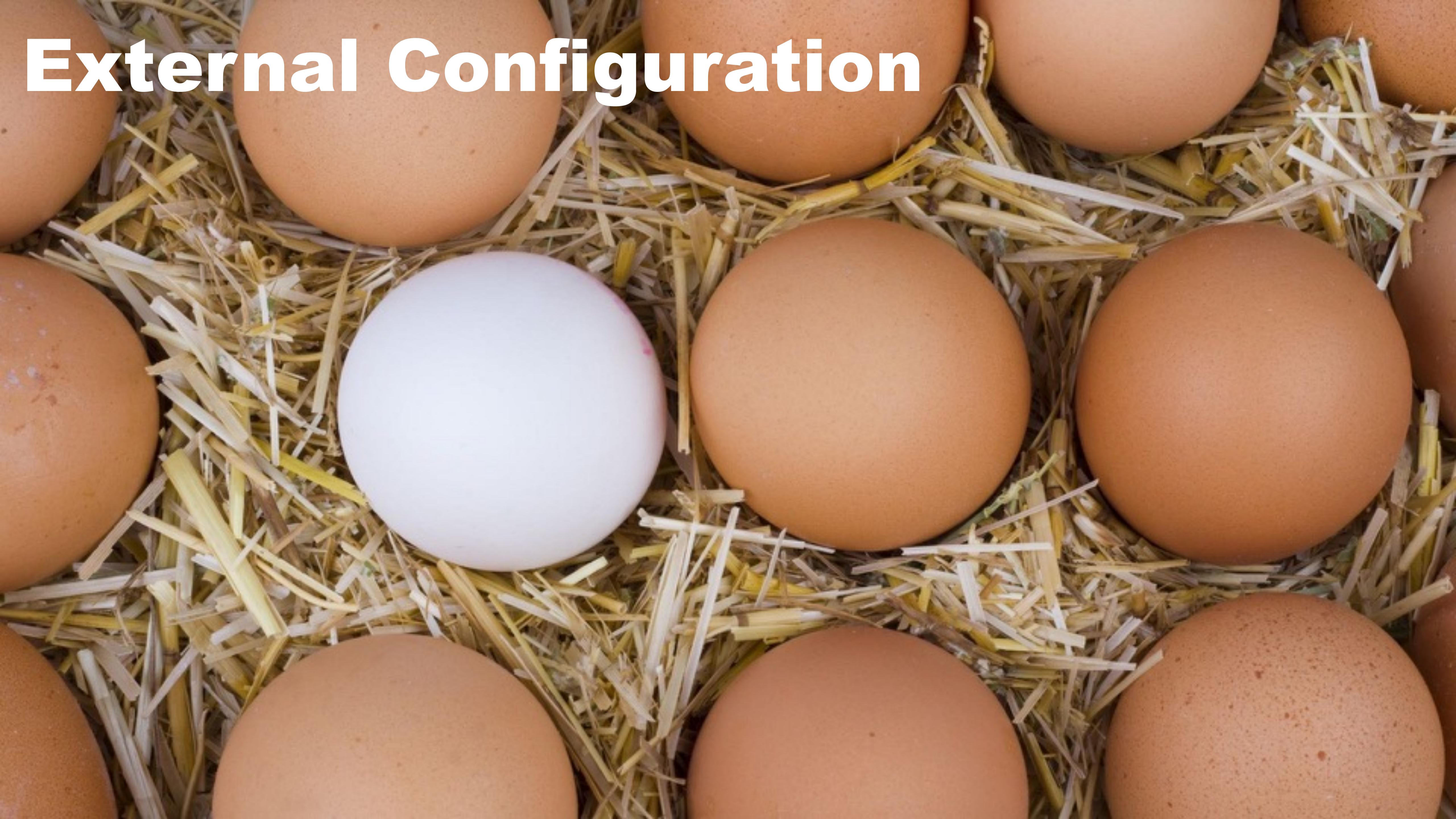
docker run

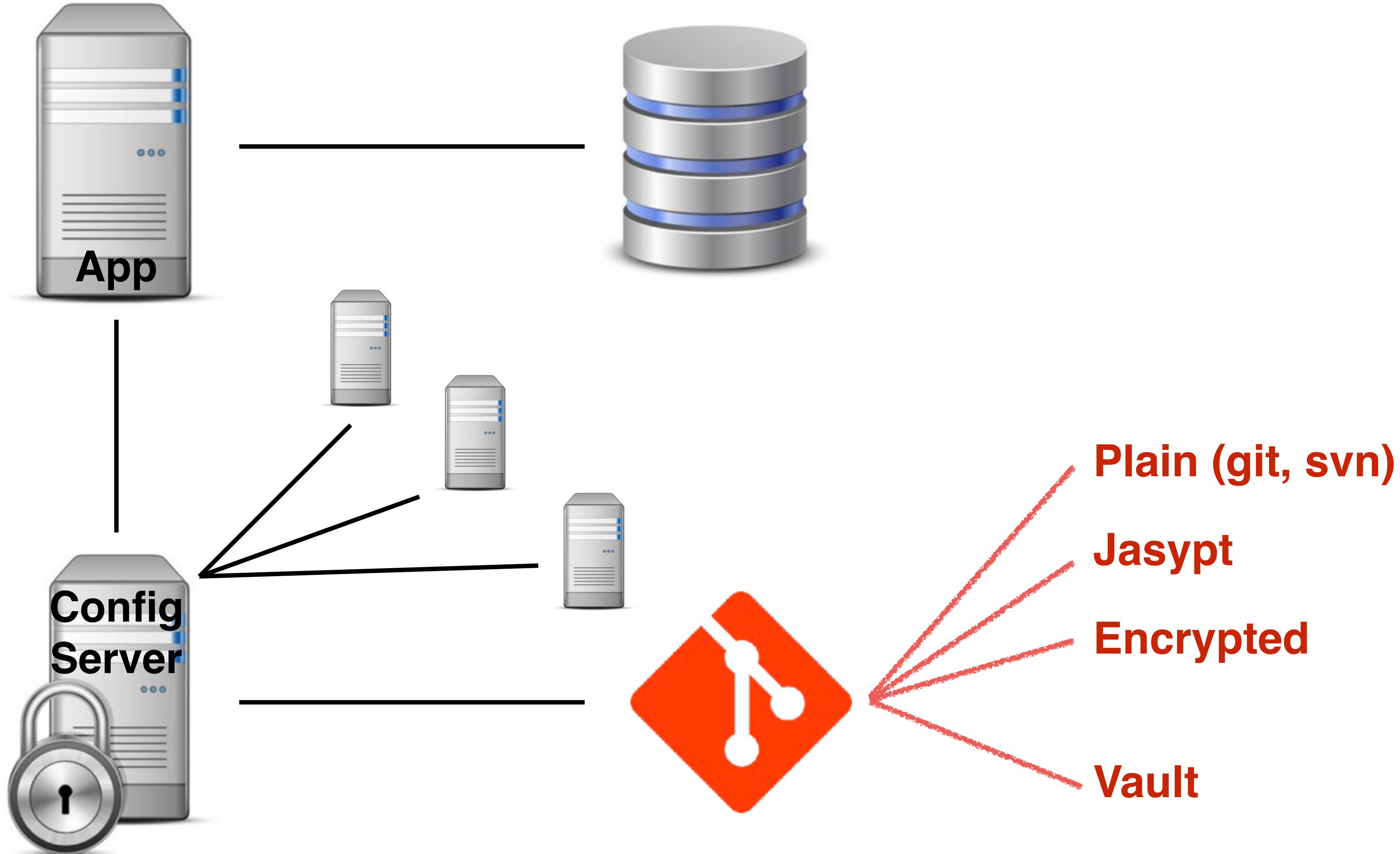
```
-e jasypt.encryptor.password=sample-password
-p 8080:8080
-t dschadow/sample
```



**Better than before, but
we are not there yet...**

External Configuration





Config Server Cryptography

Properties starting with *{cipher}* will be decrypted before returned to client

Symmetric Encryption

```
# requires key
# alternative: environment variable ENCRYPT_KEY
encrypt:
  key: mySuperSecretKey
```

Asymmetric Encryption

```
# requires keystore or pem file
encrypt:
  keyStore:
    location: classpath:/server.jks
    password: letmein
    alias: mytestkey
    secret: changeme
```

```
$ curl localhost:8888/encrypt -d config-client-db-password  
AQAGRwtH9HOM0F1CJttwZXxi/7WxGrdESX+tk8RKw4rIuB7hTiKpMljofp2LuZIxOGpXOfd/JJxt30HUUmXI++vtsGhrWsDaeD0UOo  
LRWBSQQIubC9PpPyYGLqvN7+T+smlrC0GK1PFxXoS9NA7qsVgHVq+x6083tqMAuuFrPE7UedYfYfgBYRXy5S3HjRqeat5A2NMUCpEsZ  
61vLg10PtivPoJHFk91AYc5ixY5Yfz+eIIkiV/4iHGkuygIbIJw8dqKqCJFPq+JddzfPmmI+gbDFb4ZWdrdITQBudSVPT416riMS5Lc  
TNRJVN4PHZ4uflqaeILdE/KM5vAitpd5W4EWH310zMIh6TaRr+Gf5AVEx7cjdal/m1cGJo0P3DXieASFhN8cvwkzJaI8Qr81wZz  
  
$ curl localhost:8888/decrypt -d AQAGRwtH9HOM0F1CJttwZXxi/7WxGrdESX+tk8RKw4rIuB7hTiKpMljofp2LuZIxOGpXOfd/JJxt30HUUmXI++vtsGhrWsDaeD0UOoLRWBSQQIubC9PpPyYGLqvN7+T+smlrC0GK1PFxXoS9NA7qsVgHVq+x6083tqMAuuFrPE7U  
edYfYfgBYRXy5S3HjRqeat5A2NMUCpEsZ61vLg10PtivPoJHFk91AYc5ixY5Yfz+eIIkiV/4iHGkuygIbIJw8dqKqCJFPq+JddzfPmm  
I+gbDFb4ZWdrdITQBudSVPT416riMS5LcTNRJVN4PHZ4uflqaeILdE/KM5vAitpd5W4EWH310zMIh6TaRr+Gf5AVEx7cjdal/m1cGJo  
0P3DXieASFhN8cvwkzJaI8Qr81wZz  
config-client-db-password
```

Endpoints must be secured

```
spring:
  datasource:
    name: config-client-db
    username: config-client-db-user
    password: '{cipher}
AQAPseaUfV+p34giF9jspUa475vKWV3bLwJh9sL2Gco8xB0e4GG
0z24LRIVXz5SKtESd+t9mFFkfPxJ/SgxPNOgAl+naZSay088bug
uJlpnYNHkNDafpoJmLGdeWq7ZTkdl eoCxpZWIioxz5e3GfWudsM
jmcJ6smpq6J630QkmHy6Z00av7kIKscNZksDDTikeKX02mnpGR
1BZfMYqsMF96v7o7tuAT15tTR1v6SHrUpJ83hSy8GgtWRR6egza
Tu8sYzJdkpjOmUMGXNI1flBvFd kWNt78BjzB5Lm4IiXINQFw6SO
bcTCsUv3nQbFOELVqg9ajVHrbKi3oaGwcbgpYR3VGgBzAgX/B/T
oS/WBRLtRViSXTANE9iCvqArik4Ynfti6KKROBVk9MFe8qMEiV'
```

```
{  
  name: "config-client",  
  ▼ profiles: [  
    "cipher"  
,  
    label: null,  
    version: null,  
    state: null,  
  ▼ propertySources: [  
    ▼ {  
      name: https://github.com/dschadow/CloudSecurity/config-repo/config-client-cipher.yml,  
      ▼ source: {  
        "application.name": "Config Client",  
        "application.profile": "Config Server Cipher",  
        "spring.datasource.name": "config-client-db",  
        "spring.datasource.username": "config-client-db-user",  
        "spring.h2.console.enabled": true,  
        "invalid.spring.datasource.password": "<n/a>"  
      }  
    }  
  ]  
}
```

decryption failure

```
{  
    name: "config-client",  
    ▼ profiles: [  
        "cipher"  
    ],  
    label: null,  
    version: null,  
    state: null,  
    ▼ propertySources: [  
        ▼ {  
            name: https://github.com/dschadow/CloudSecurity/config-repo/config-client-cipher.yml,  
            ▼ source: {  
                "application.name": "Config Client",  
                "application.profile": "Config Server Cipher",  
                "spring.datasource.name": "config-client-db",  
                "spring.datasource.username": "config-client-db-user",  
                "spring.h2.console.enabled": true,  
                "spring.datasource.password": "config-client-db-password"  
            }  
        }  
    ]  
}
```

Using Multiple Keys

Select existing key with
{key:name} in properties file

```
spring:  
  datasource:  
    password: '{cipher}{key:myFirstKey}AQUfV+...'
```

Demo



AU L T

A tool for managing secrets.

,,.. centrally store, secure, and tightly control access to secrets across distributed infrastructure, applications, and humans.“

Data is Always Encrypted

Internal key encrypts all data with AES

This key never leaves the system

Configured storage backend never sees plain text

File, Amazon DynamoDB, Consul, ...

High availability

Secret Storage

- Store existing secrets
- Dynamically create new secrets
- Lease time for new secrets, automatic revocation
- Access control policies for secrets

Audit Logs

- Not active by default
- Detailed audit log of all authenticated client interaction
- Trace lifetime and origin of any secret
- Hashes sensitive information with HMAC-SHA256

Accessible via HTTP API or CLI

Unsealing requires configured number of master keys

```
vault server -config vault.conf
```

```
export VAULT_ADDR=http://127.0.0.1:8200
```

DEMO!

```
vault init -key-shares=5 -key-threshold=2
```

Shamir's Secret Sharing

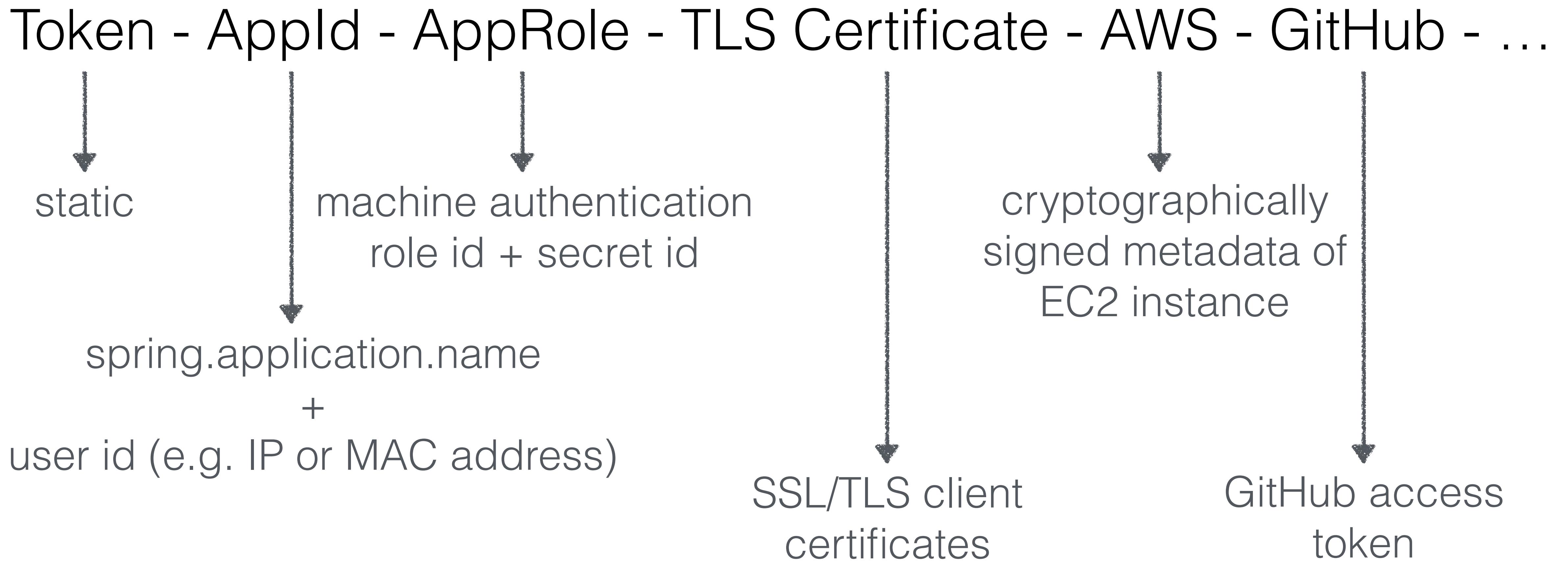
Unseal Key 1: Pv/Xx49co4Zmed2McapSOr4jC4iiAvfd5EjvILMySJUB
Unseal Key 2: T00pjFgitbcy+JKOGI6DFgW/0jBdyrVriLdGu7PENbsC
Unseal Key 3: YCOKtRUIT1H3h155P5LM+2zLbFgIe4vwrOIhO70WHqED
Unseal Key 4: rTLOGu3emdWa4QyKysY6Tmicelu4QTEcUFIPlrMzz+cE
Unseal Key 5: glxtI6D0YjNfnsB97dp1owHoxTPt8A+HdAdoFrNh5P0F
Initial Root Token: efe88b79-cf8b-825a-0f6f-ef1ca142782b

Only visible after initialization

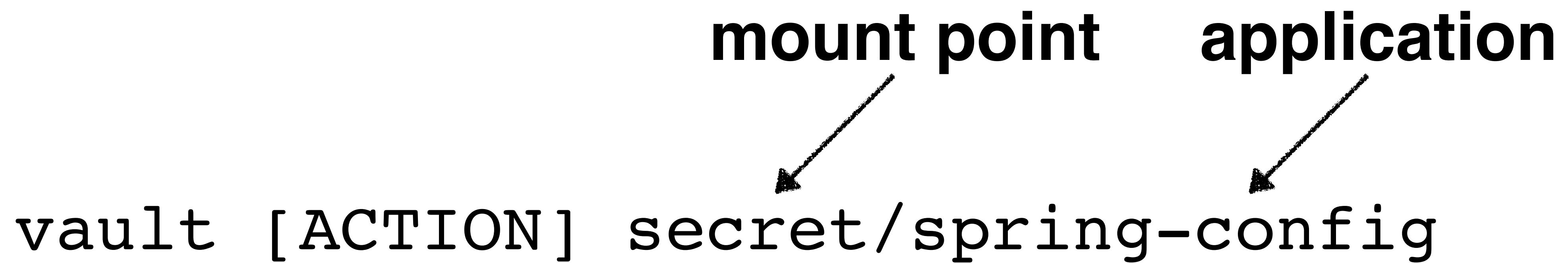
```
vault unseal Pv/Xx49co4Zmed2McapSOr4jc4iiAvfd5EjvILMySJUB
Sealed: true
Key Shares: 5
Key Threshold: 2
Unseal Progress: 1
Unseal Nonce: 87f350d5-2a25-a821-dc7f-2962fc49fe03
```

```
vault unseal rTLOGu3emdWa4QyKysY6Tmice1u4QTEcUFIPlrMzz+cE
Sealed: false
Key Shares: 5
Key Threshold: 2
Unseal Progress: 0
Unseal Nonce:
```

Authenticated Access Required



Path Based Secret Storage



Each ,User‘ is Assigned a Policy

Policies use path based matching to apply rule

Policy may constrain actions and paths

```
path "secret/*" {  
    policy = "write"  
}  
  
path "auth/token/lookup-self" {  
    policy = "read"  
}
```

CLI

```
export VAULT_TOKEN
```

```
=
```

```
efe88b79-cf8b-825a-0f6f-ef1ca142782b
```

HTTP API

```
X-Vault-Token
```

```
vault write secret/spring-config  
db.password=config-client-db-password
```



key/value format (generic backend)

```
$ vault read secret/spring-config  
Key          Value  
---          ----  
refresh_interval    768h0m0s  
db.password        config-client-db-password
```

Spring Cloud Vault

Secrets are picked up during application startup

Path based on application name and active contexts

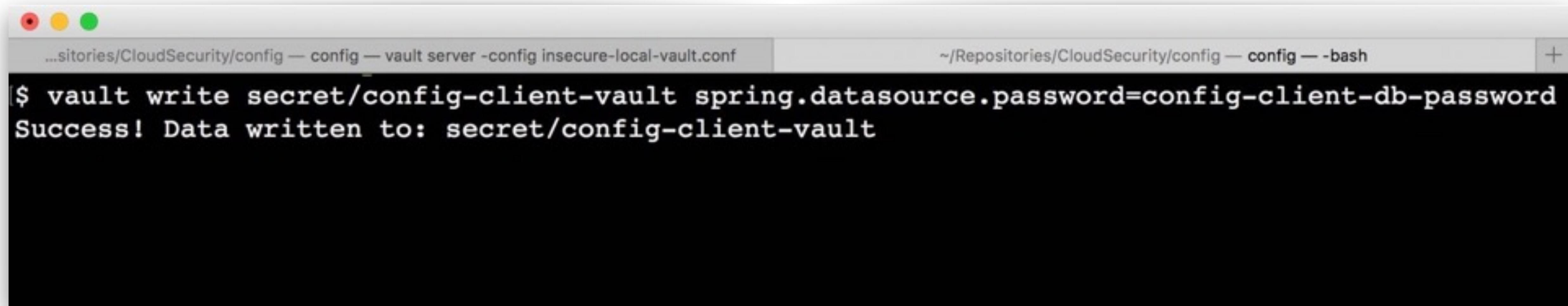
/secret/{application}/{profile}

/secret/{application}

/secret/{defaultContext}/{profile}

/secret/{defaultContext}

Reading and Writing Secrets



A screenshot of a macOS terminal window. The window title bar shows two tabs: "...sitories/CloudSecurity/config — config — vault server -config insecure-local-vault.conf" and "~/Repositories/CloudSecurity/config — config — -bash". The main pane of the terminal displays the following command and its output:

```
$ vault write secret/config-client-vault spring.datasource.password=config-client-db-password
Success! Data written to: secret/config-client-vault
```

Actuator Health Endpoint

```
{  
    status: "UP",  
    ▼ diskSpace: {  
        status: "UP",  
        total: 999250984960,  
        free: 669155143680,  
        threshold: 10485760  
    },  
    ▼ db: {  
        status: "UP",  
        database: "H2",  
        hello: 1  
    },  
    ▼ refreshScope: {  
        status: "UP"  
    },  
    ▼ vault: {  
        status: "UP"  
    }  
}
```

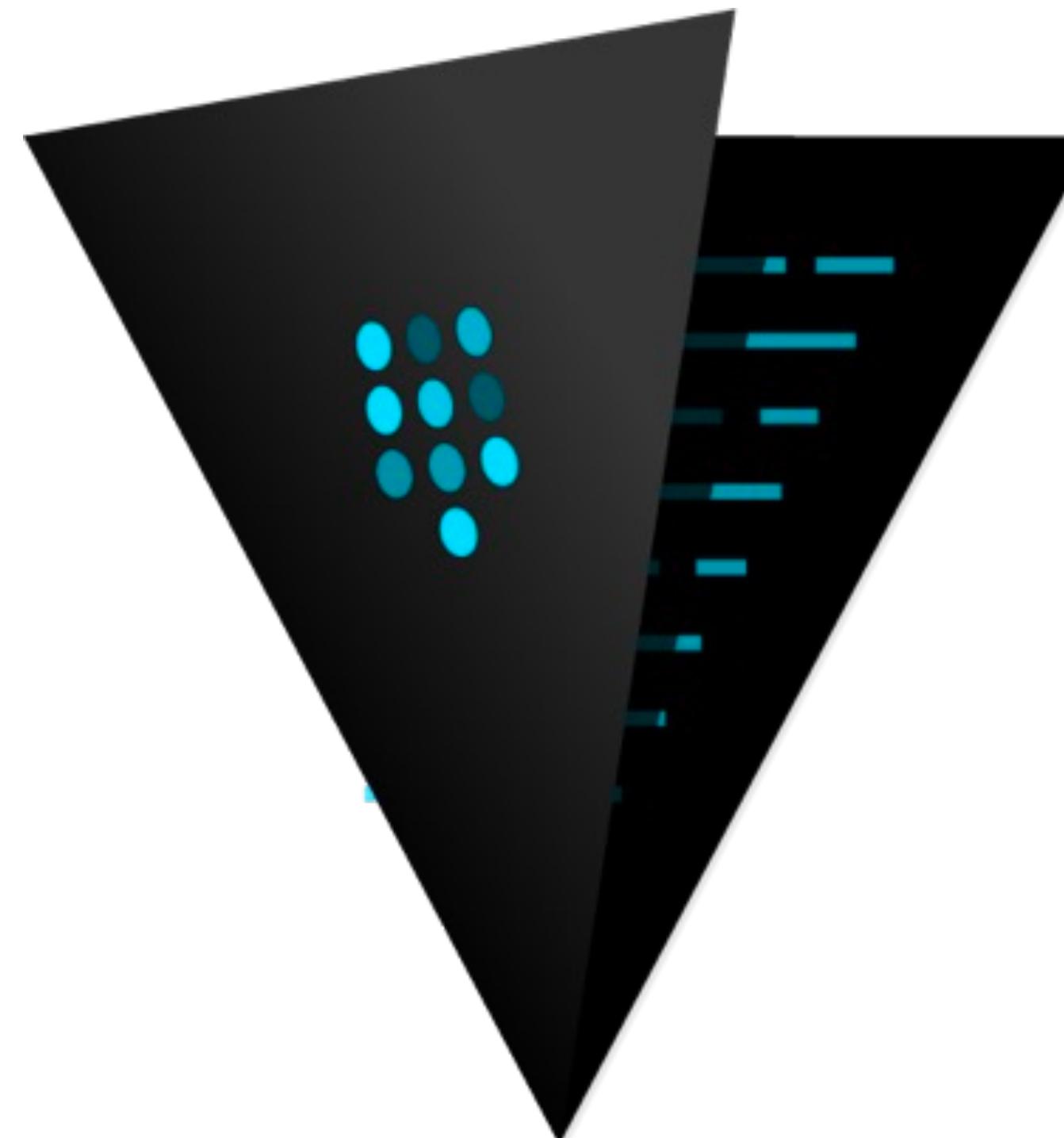
Demo



A series of modern, metallic turnstiles are lined up in a public area, likely a subway or train station. Each turnstile features a large, curved, illuminated display screen at the top and a circular sensor or card reader in the center. The floor is a polished, light-colored material, and the background shows a glass wall reflecting the interior of the building.

Personal Credentials

API for Secret Management



VaultTemplate to read, write, list and delete secrets

Similar to RestTemplate

```
vaultTemplate.read(PATH);
```

Demo



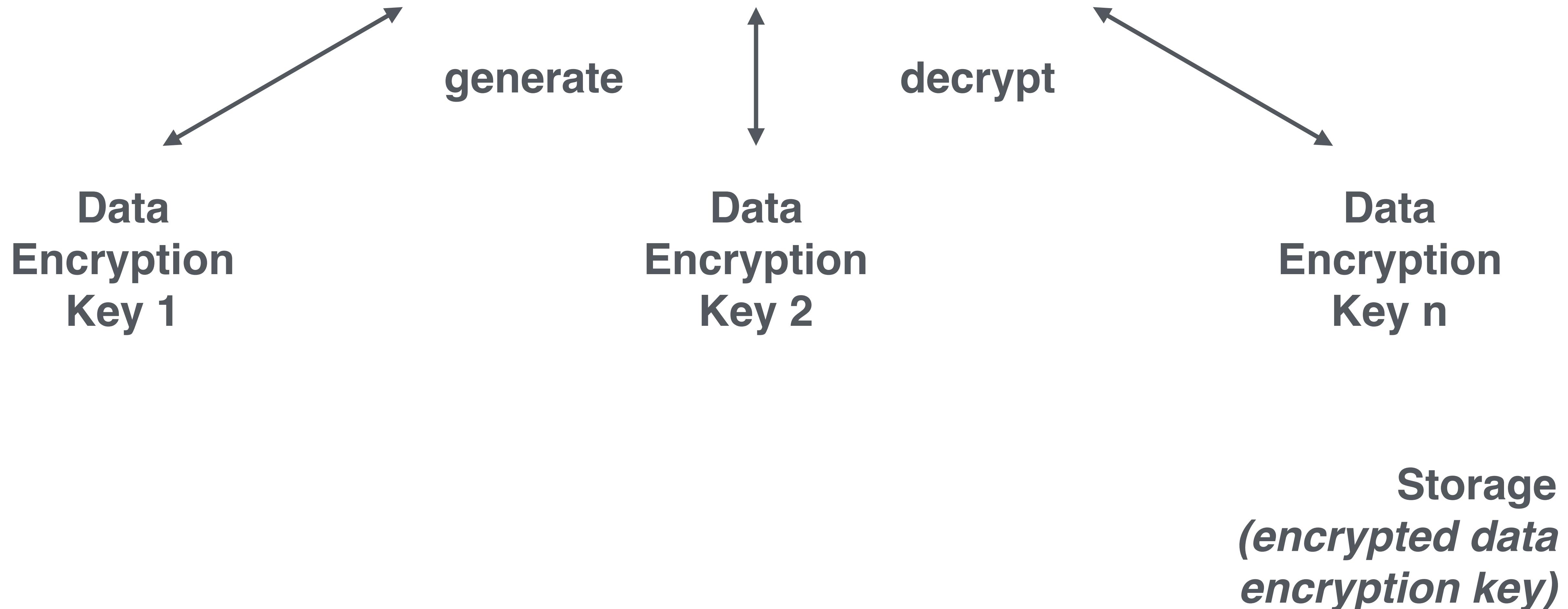
**We have created
another system
to monitor and
maintain...**



AWS Key Management Service (KMS)

- Master key to protect all keys
- Service for storing encryption keys
 - Supporting cryptographic operations with these keys
- Access control and auditing functionality

Key Encryption Key (a.k.a. master key)



Microsoft Azure
Key Vault

IBM Bluemix
Key Protect



Summary

- Multiple options to protect sensitive data exist
- Keep it as simple as possible
 - **Jasypt** for simple applications
 - **Config Server** cipher for distributed applications
 - **Vault** when required
- Consider cloud provider services



Marienstr. 17
70178 Stuttgart

dominik.schadow@bridging-it.de
www.bridging-it.de

Blog blog.dominiksshadow.de
Twitter @dschadow

Demo Project
<https://github.com/dschadow/CloudSecurity>

AWS KMS
<https://aws.amazon.com/de/kms>

Jasypt integration for Spring boot
<https://github.com/ulisesbocchio/jasypt-spring-boot>

Spring Cloud
<http://projects.spring.io/spring-cloud>

Vault
<https://www.vaultproject.io>

Pictures
<http://www.dreamstime.com>

