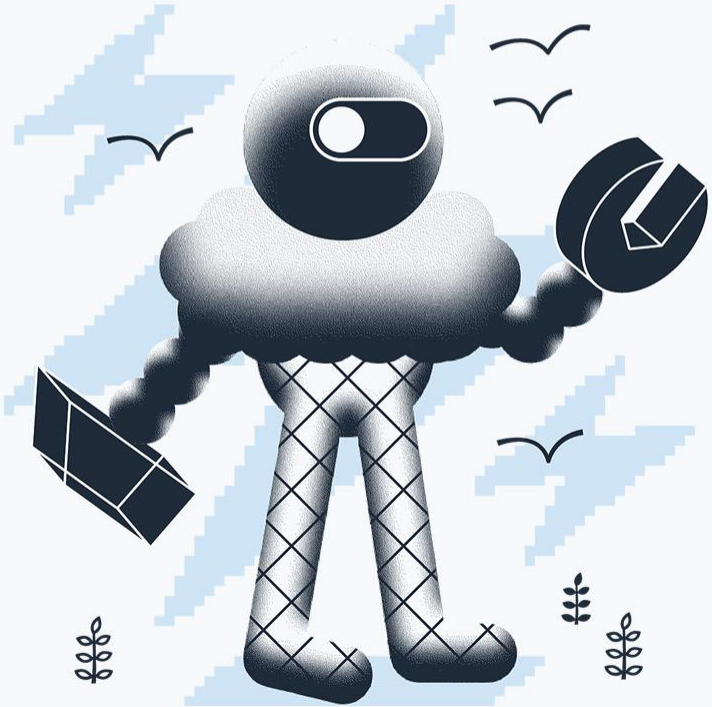
3fs

Secrets handling in cloud environments



Jernej Porenta
Staff infrastructure engineer

2025-06-26

"In cloud environments, a secret refers to sensitive information such as passwords, API keys, certificates, or tokens that are securely stored and managed to protect access to systems, applications, and data."

DeepSeek

"Tell me what is a secret in scope of cloud environments in a single sentence."

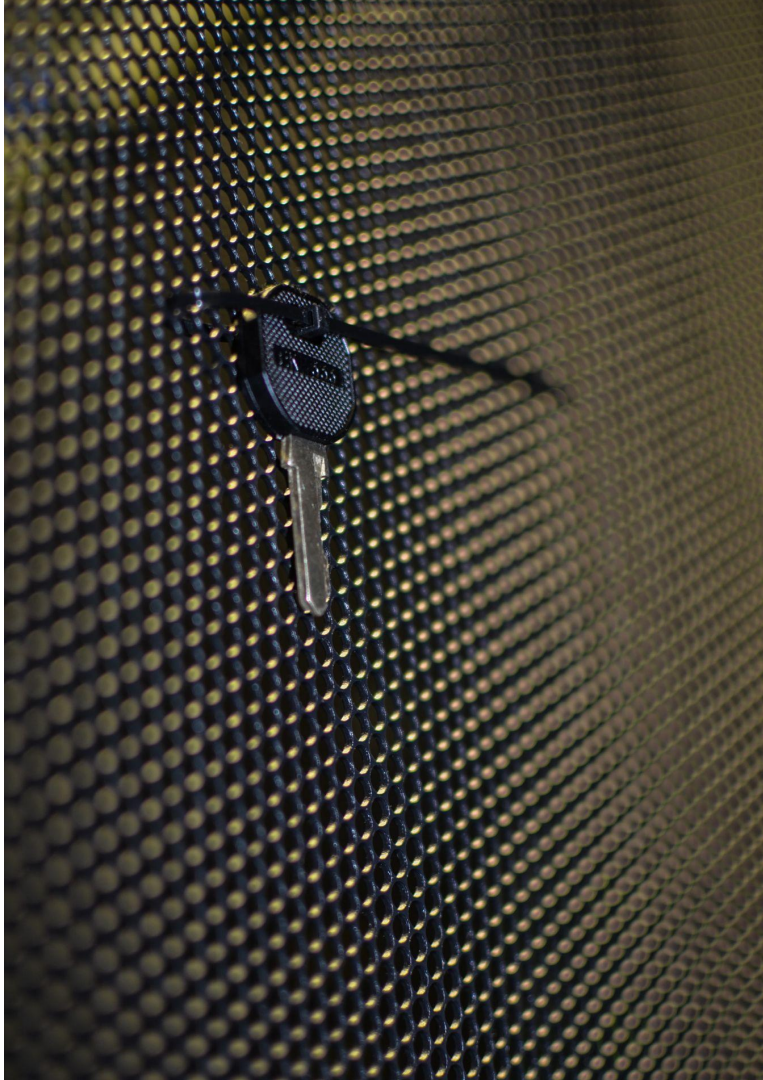
"In cloud environments, a secret refers to sensitive information such as passwords, API keys, certificates, or tokens that are securely stored and managed to protect **access to systems**, **applications**, and **data**."

DeepSeek

"Tell me what is a secret in scope of cloud environments in a single sentence."

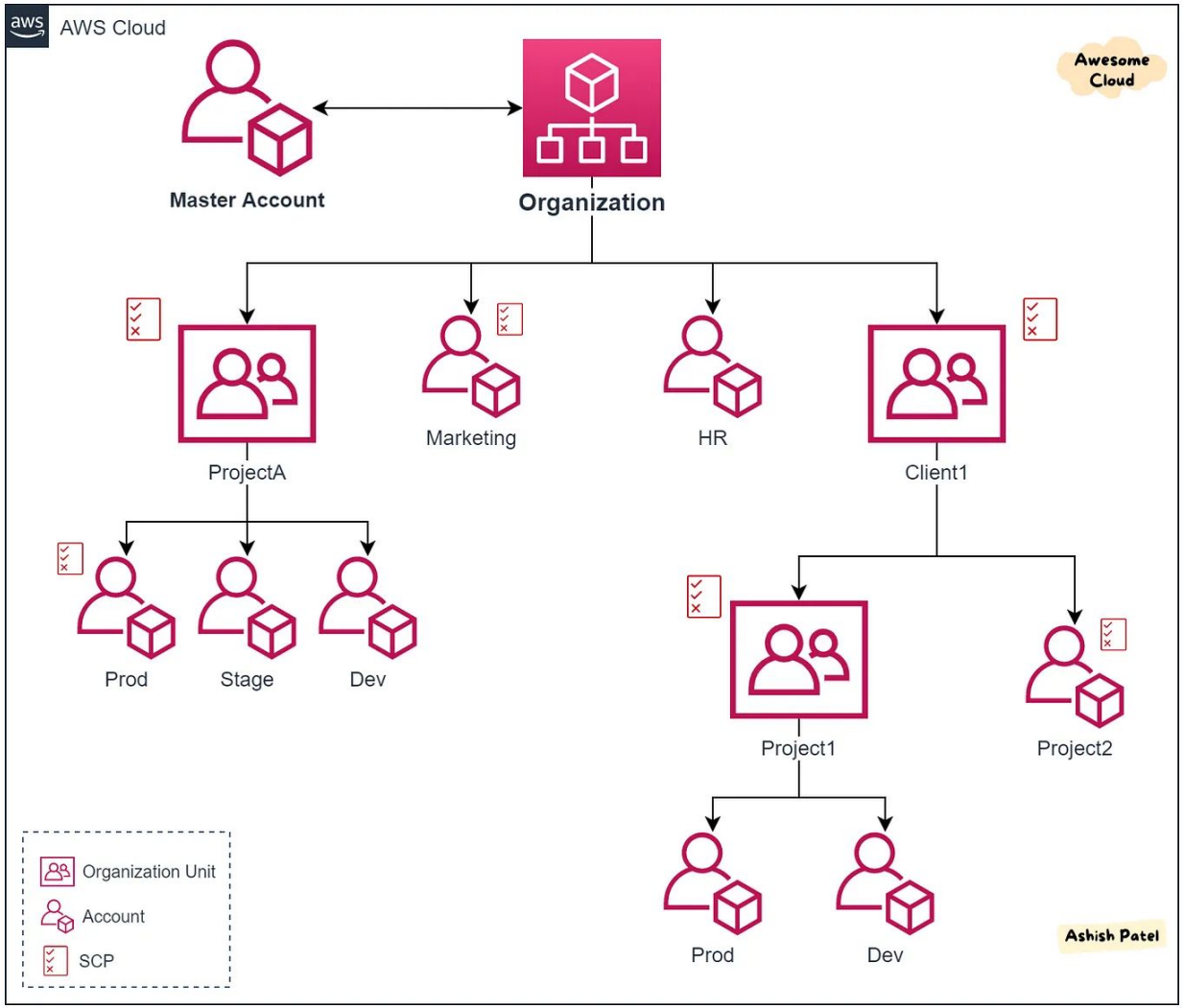
Access to systems

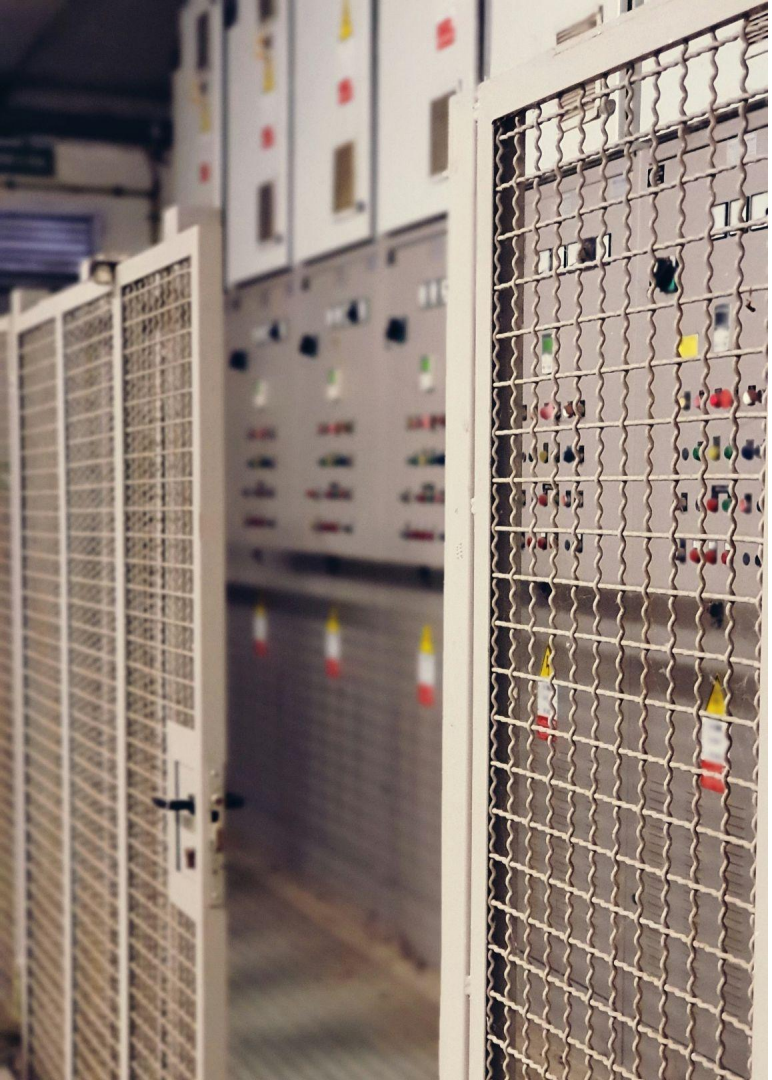
- Access to your cloud environment is access to your **datacenter**
- Authentication method
  - Passwords
  - Multi factor authentication
- Protect master access to management portal
  - Secure password storage
  - Multiple MFA
  - Four eyes principle
  - Break glass role
  - Audited access



3fs

Multiple account setup





Access to systems for users

- User access to management portals
  - Creating resources
  - Monitoring
- Best practices
  - External identity providers (AWS Identity Center (formerly known as AWS SSO))
  - **No static credentials**
  - Least privilege principle with ded
  - Dedicated roles
  - Dynamic role assignment through external attributes

Access to systems for others

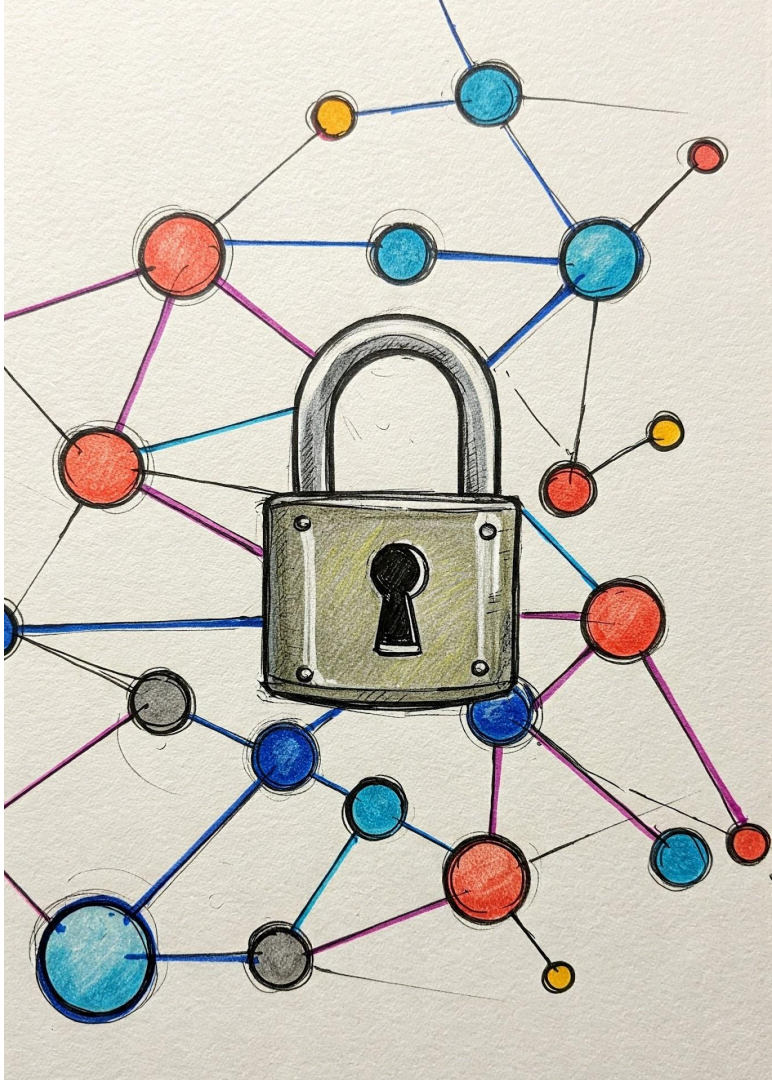
- Non-human access to management interfaces
  - Infrastructure as a Code
  - CI/CD deployments
  - SIEM systems
- Best practices
  - **Least privilege principle**
  - Continuous threat modeling
  - Drop credentials when not needed

Access to systems

- Recap
  - Offline root credentials
  - Multiple accounts
  - **No static credentials**
  - Role based access with least privilege principle applied
  - OIDC based authentication and authorization
  - Break glass role
  - No touch deployments

Application secrets

- Handling secrets in applications
  - Passwords (e.g. database access)
  - API keys (e.g. external services)
  - Certificates (private keys)
- Passing secrets to applications
  - Environment variables
  - Configuration files
  - Workload identity (IAM Roles)



Application secrets

- Workload identity
  - Assigning permissions to cloud application workload
- Benefits
  - Secure service authentication
  - Granular access control
  - Audit and compliance

Application secrets by using environment vars

```python
def connect_to_database():
    # Create a connection using environment variables
    conn = psycopg2.connect(
        host=os.getenv("DB_HOST", "localhost"),
        port=os.getenv("DB_PORT", "5432"),
        dbname=os.getenv("DB_NAME"),
        user=os.getenv("DB_USER"),
        password=os.getenv("DB_PASSWORD"),
    )

    # Use the connection to execute a simple query
    with conn.cursor() as cur:
        cur.execute("SELECT version();")
```

Application secrets by using configuration file

```python
def get_db_config(filename="db_config.ini", section="database"):
    parser = ConfigParser()
    # read the configuration out of configuration file
    parser.read(filename)
    return {key: value for key, value in parser.items(section)}

def connect_to_database():
    db_config = read_db_config()
    conn = psycopg2.connect(**db_config)

    # Use the connection to execute a simple query
    with conn.cursor() as cur:
        cur.execute("SELECT version();")
```

Application secrets by using workload identity

```python
def get_db_credentials(secret_name="my-db-credentials"):
    # use workload identity to get the secret out of AWS Secrets Manager
    client = boto3.client("secretsmanager", region_name="eu-central-1")
    response = client.get_secret_value(SecretId=secret_name)
    secret = response.get('SecretString')
    return json.loads(secret)

def connect_to_database(secret_name):
    db_config = get_db_credentials(secret_name)
    conn = psycopg2.connect(dsn=db_config['dsn'])

    # Use the connection to execute a simple query
    with conn.cursor() as cur:
        cur.execute("SELECT version();")
```

Application secrets

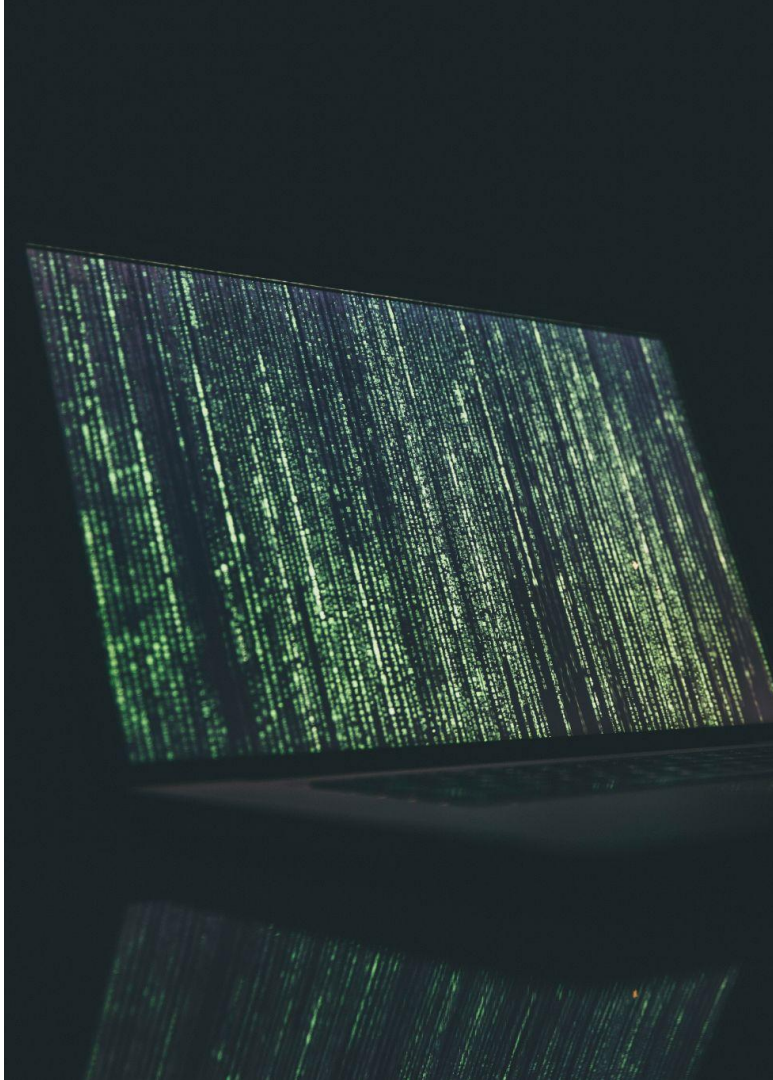
- Secure cloud secrets storage
  - AWS Secrets Manager
  - Systems Manager Parameter Store
- On prem secure storage
  - Hashicorp Vault
- Advanced types
  - Cloud hardware security modules (HSM)

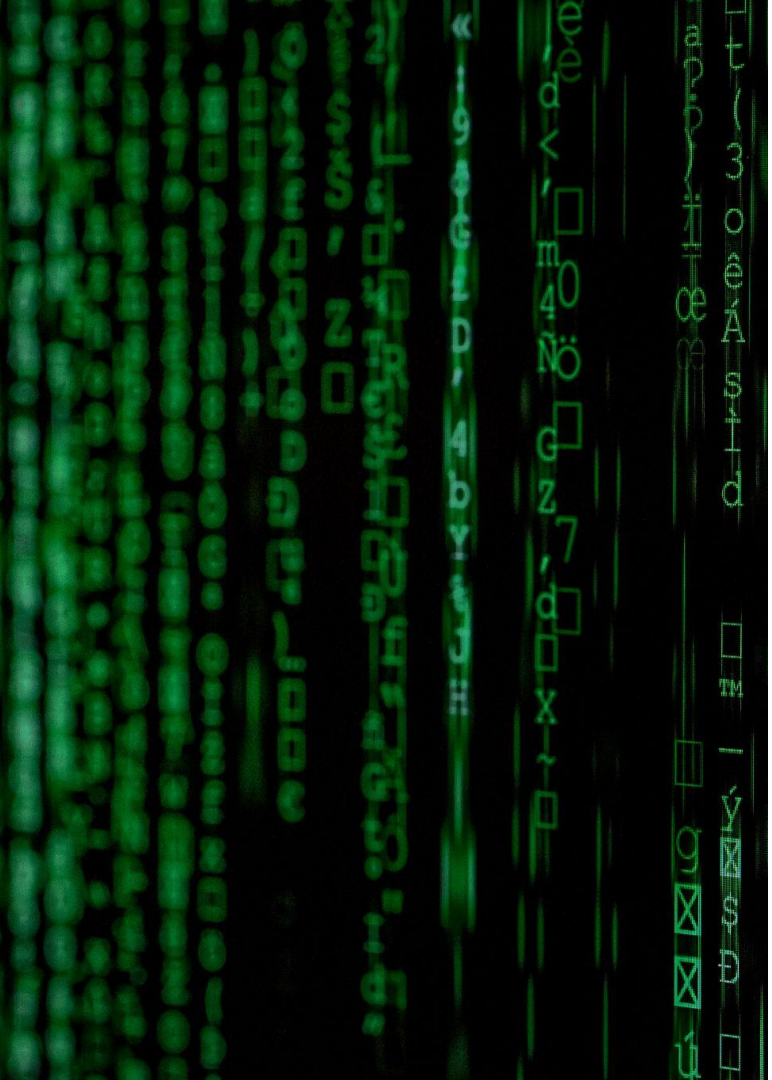
Application secrets

- Best practices
  - No static credentials
  - Workload identity
  - Let application checkout secret out of cloud secrets storage system
  - HSM for critical secrets
  - Secrets rotation

Data secrets

- Data in the cloud is like data in the datacenter
  - Encrypted
  - Data at rest / in transit
- **Encryption** is the key, but where is the key?
  - Provider managed keys
  - Customer managed keys





Data secrets

- AWS Key Management Service
- Use cases
  - Data at rest
  - Digital signatures
  - Certificate storage

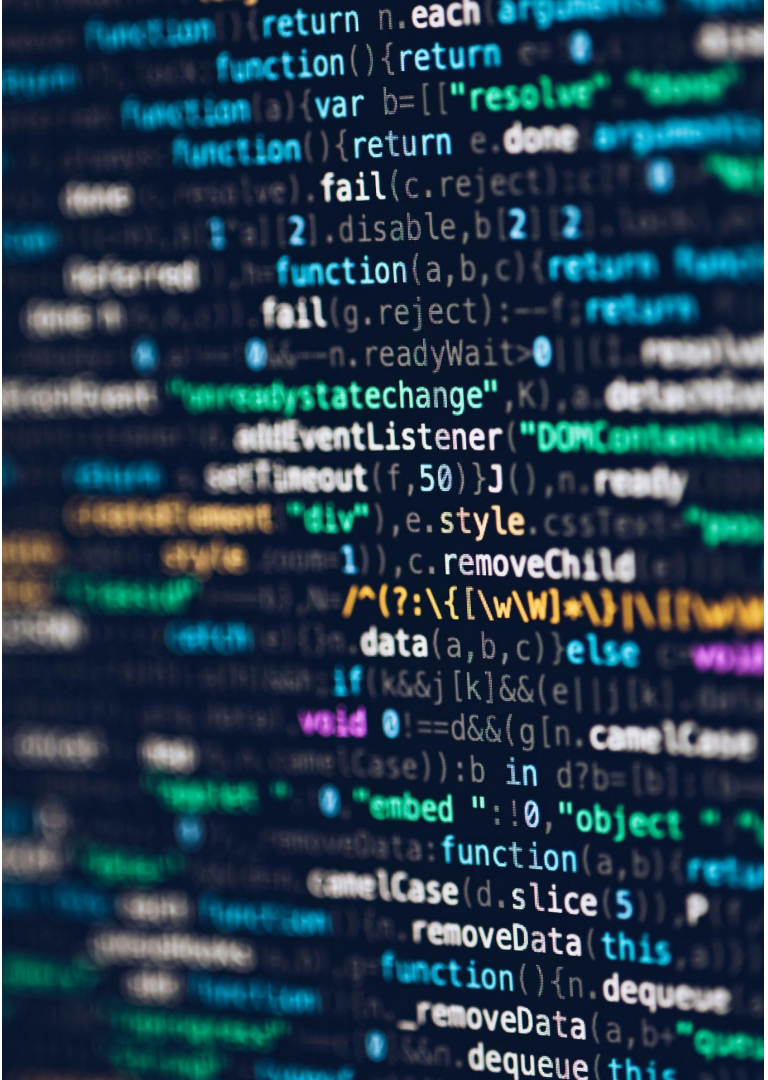
Cloud encryption services

Advantages

- Automatic secret rotation
- Role based access control
- Compliance
- Seamless integration into cloud services
- Centralized, scalable service
- Distributed key management

Disadvantages

- Location of key material
- “Bring Your Own Key” attacks
- Distributed key management
- Pricing
- Limited customization
- Disaster recovery procedures



Data secrets

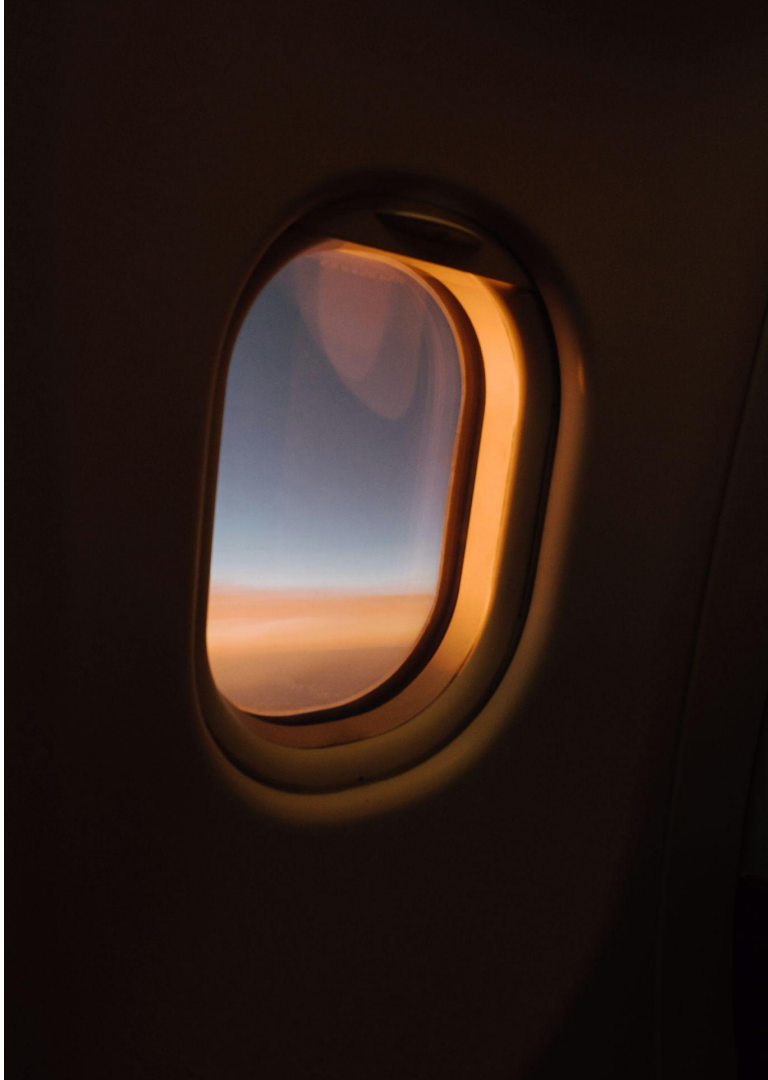
- Best practices
  - **Control access to the encryption keys**
  - Control access to the data
  - Least privilege permissions per key
  - HSM for critical secrets
  - Secrets rotation
  - Multi region deployments
  - SIEM key action monitoring

Honeytokens

- Digital resources **decoy**
  - API keys
  - Fake IAM roles
  - Certificates
- Benefits
  - Early detection
  - Minimal false positives
  - Forensics analysis
- Requirements
  - Good positions
  - Monitoring and alerting systems

Takeaways

- Treat your cloud as datacenter
- Do not store your secrets in plain text
- Use cloud provided secret storage systems
- Use cloud provided encryption services
- Rotate secrets



Resources

https://external-secrets.io/
Integrating secrets from various sources into Kubernetes objects

https://www.vaultproject.io/
Secure, store, and tightly control access to tokens, passwords, certificates, and encryption keys

https://github.com/trufflesecurity/trufflehog
Find, verify, and analyze leaked credentials

https://canarytokens.org/
Track activity and actions on your network or environment

https://github.com/getsops/sops
Simple and flexible tool for managing secrets with cloud encryption built in

13. September, FE UNI LJ
https://cfp.devopsdays.si

3fs

QUESTIONS?

jernej@3fs.si