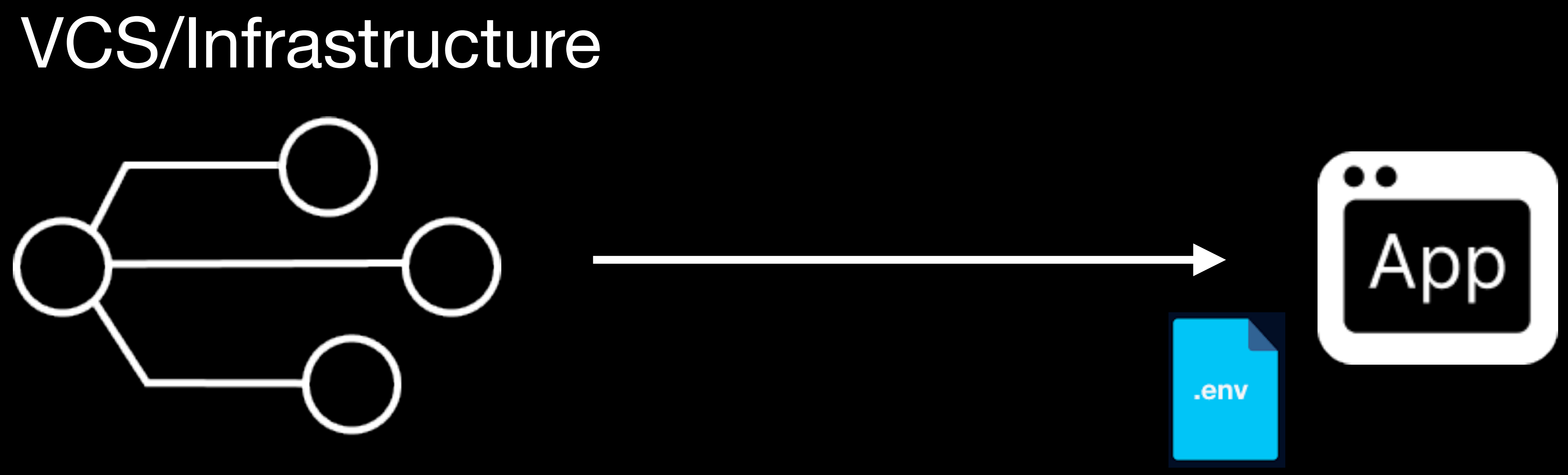
Building a Vault Client in Swift with OpenAPI and Pkl

Or How I Learned to Stop worrying and love the Bao

Javier Cuesta 30.01.2026 - Swift @ FOSDEM

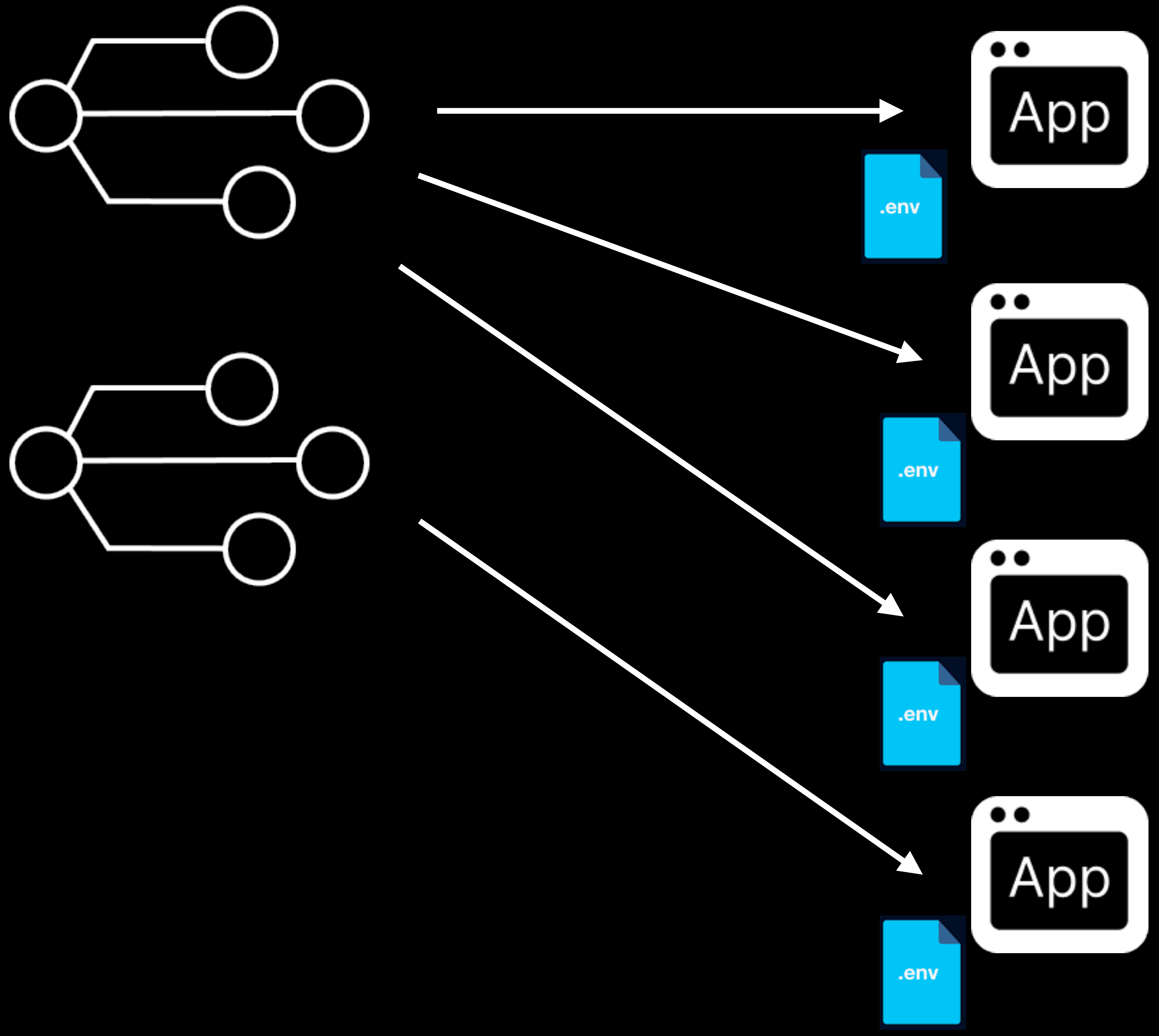
Injecting Secrets

Injecting Secrets



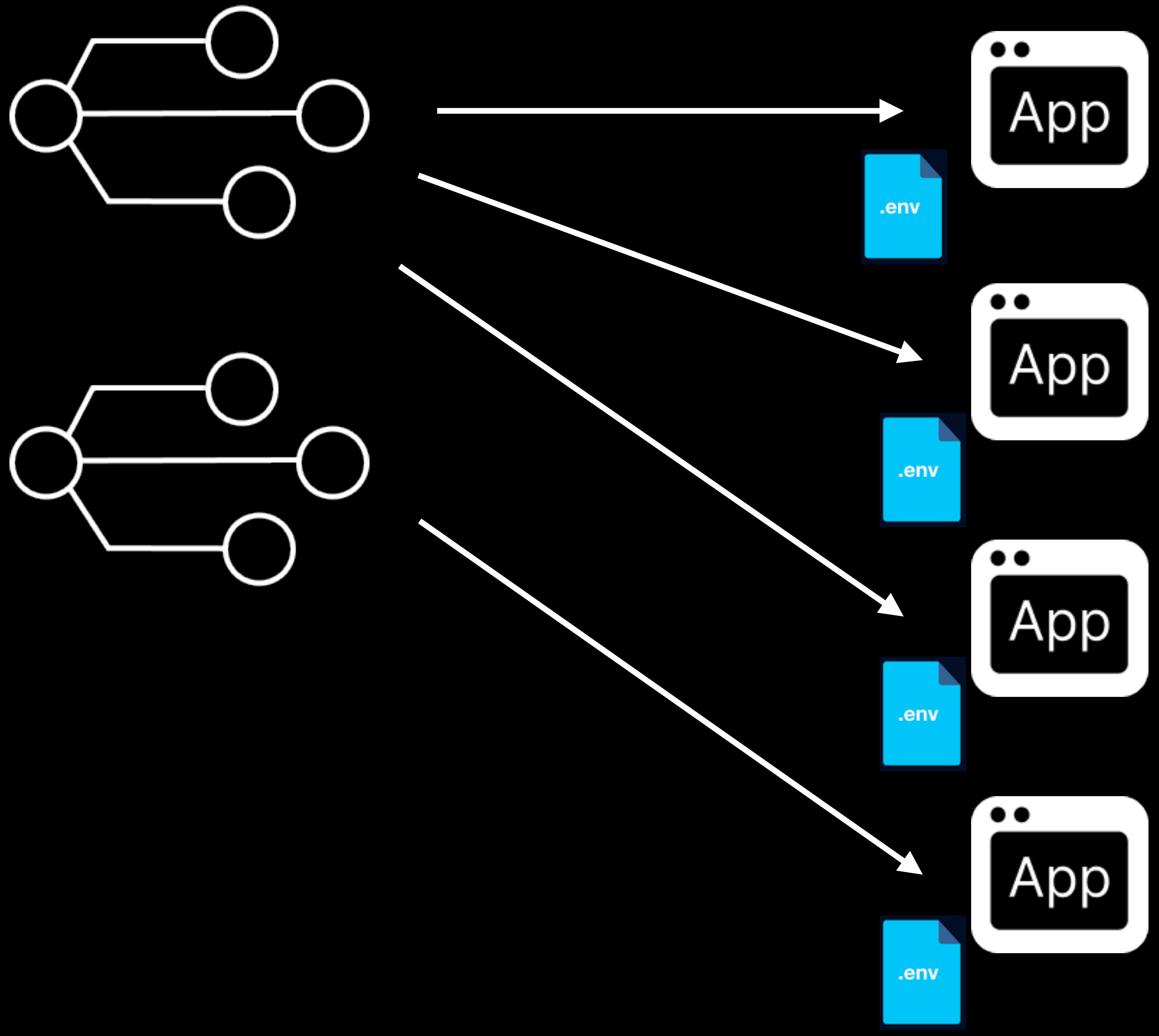
Injecting Secrets

VCS/Infrastructure



Injecting Secrets

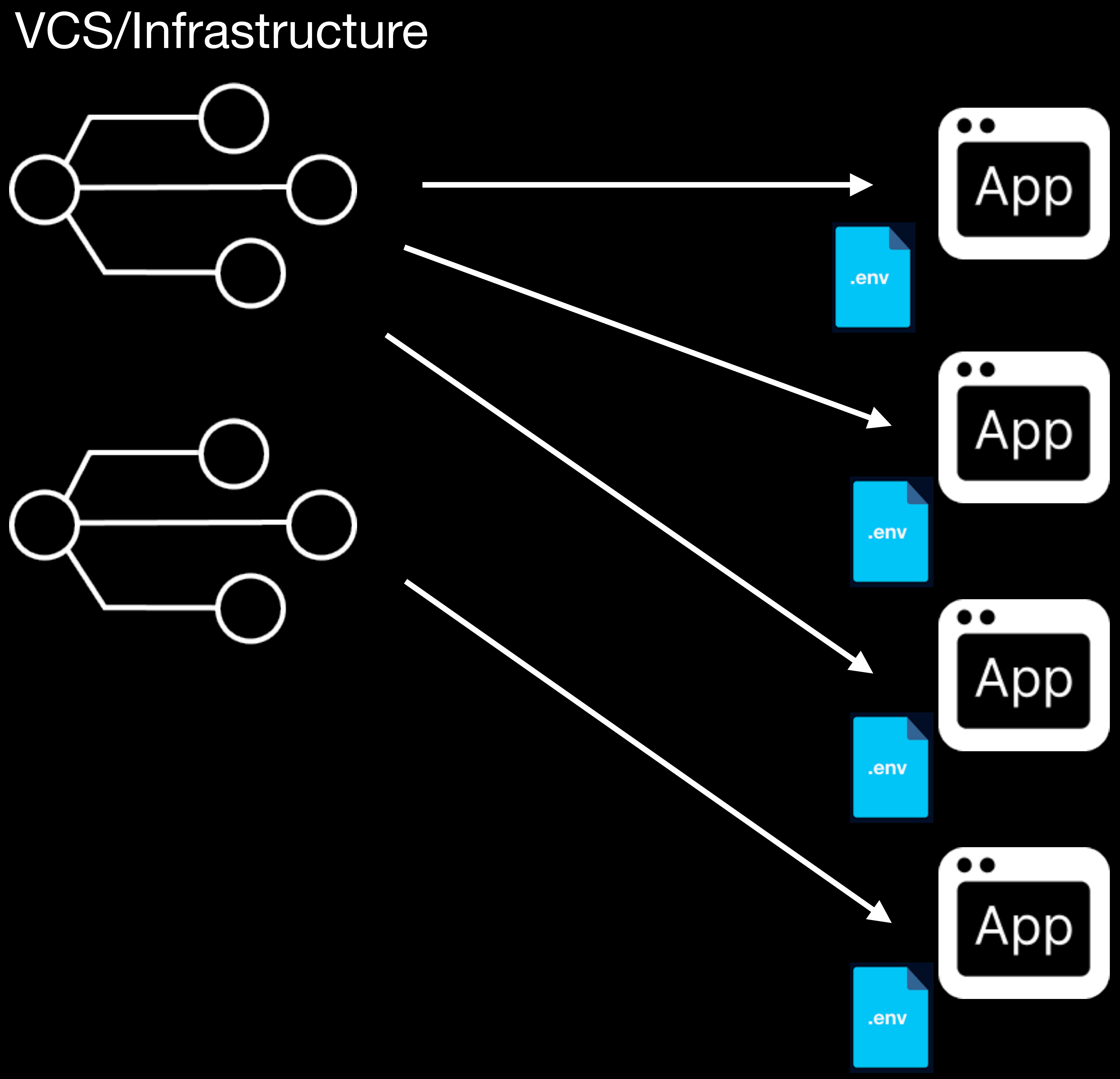
VCS/Infrastructure



Issues with this approach

Secret Sprawl

Injecting Secrets

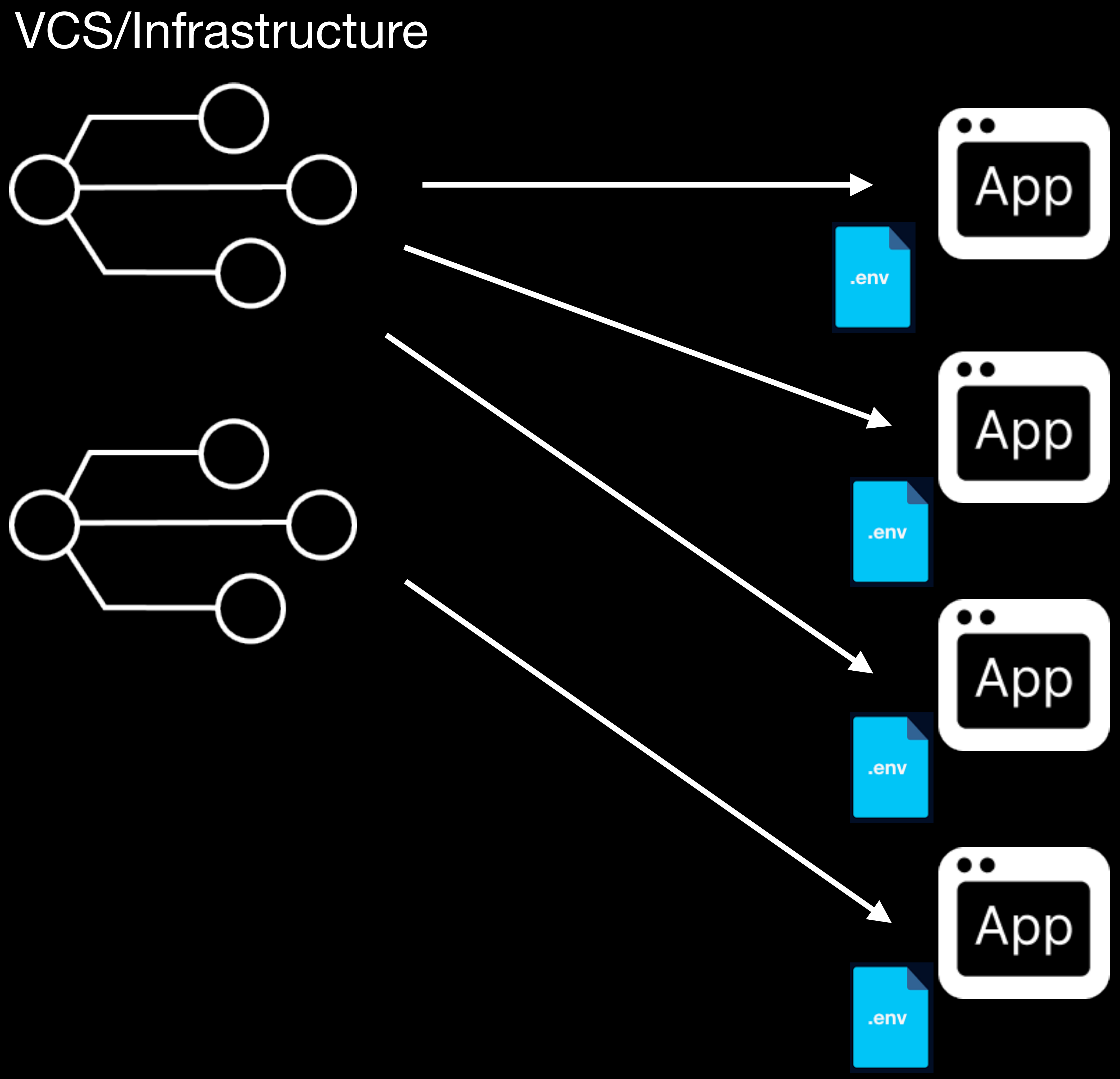


Issues with this approach

Secret Sprawl

Lack of Auditability

Injecting Secrets



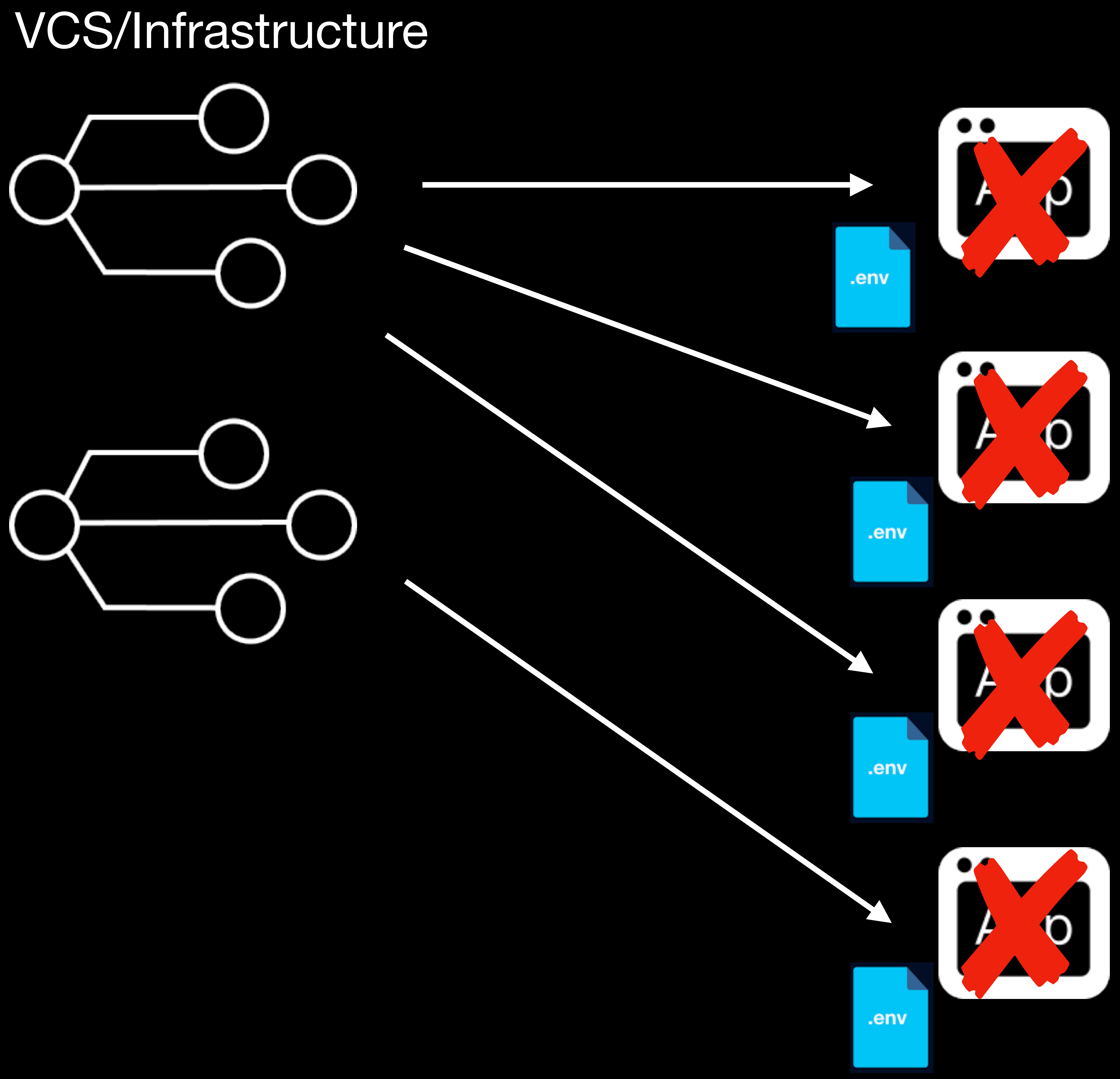
Issues with this approach

Secret Sprawl

Lack of Auditability

Difficult Secret Rotation

Injecting Secrets



Issues with this approach

Secret Sprawl

Lack of Auditability

Difficult Secret Rotation

Modern secret management

HashiCorp Vault OpenBao

Centralized secret management

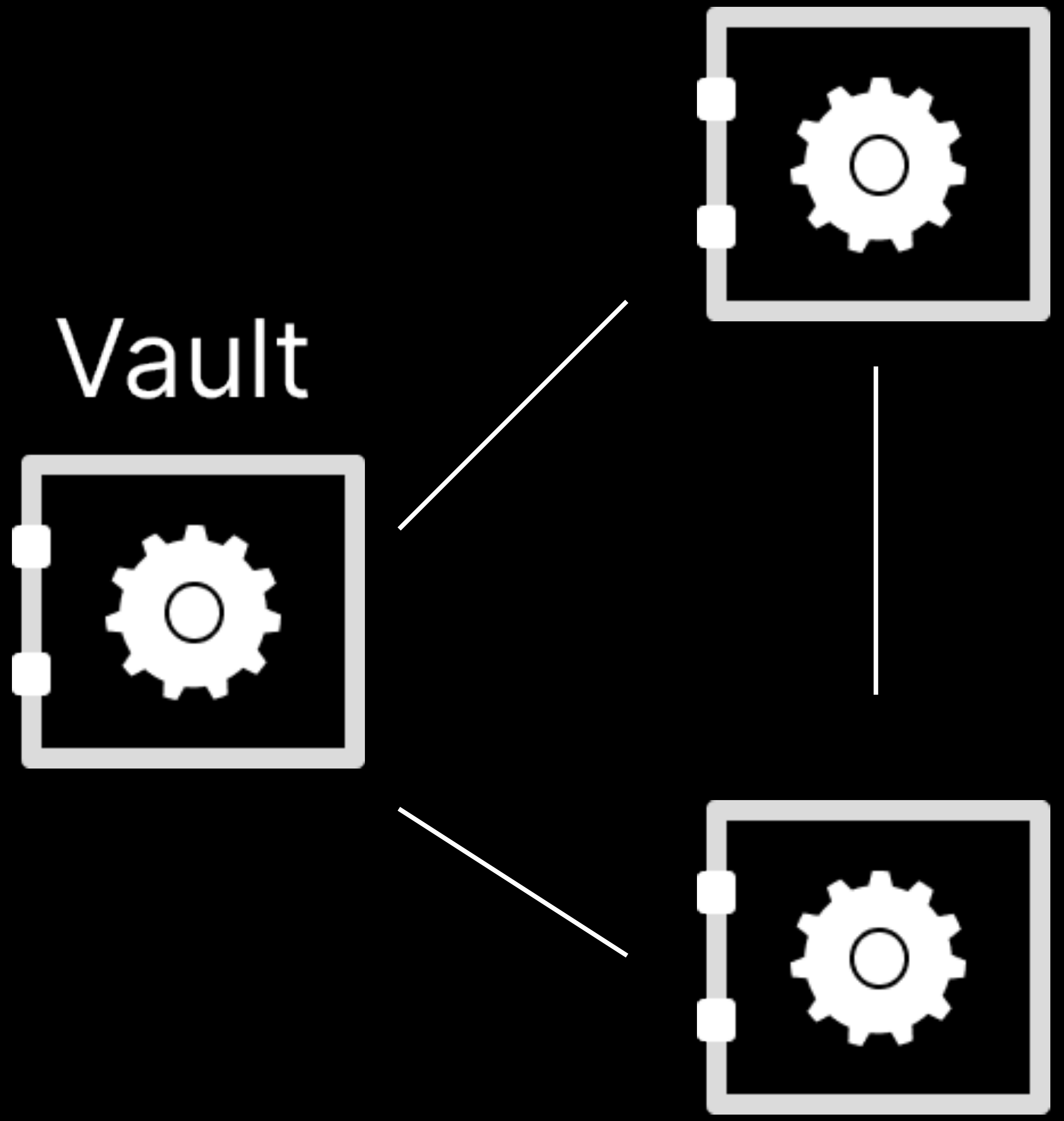
Audit trail

Fine-grained access control via ACL/RBAC

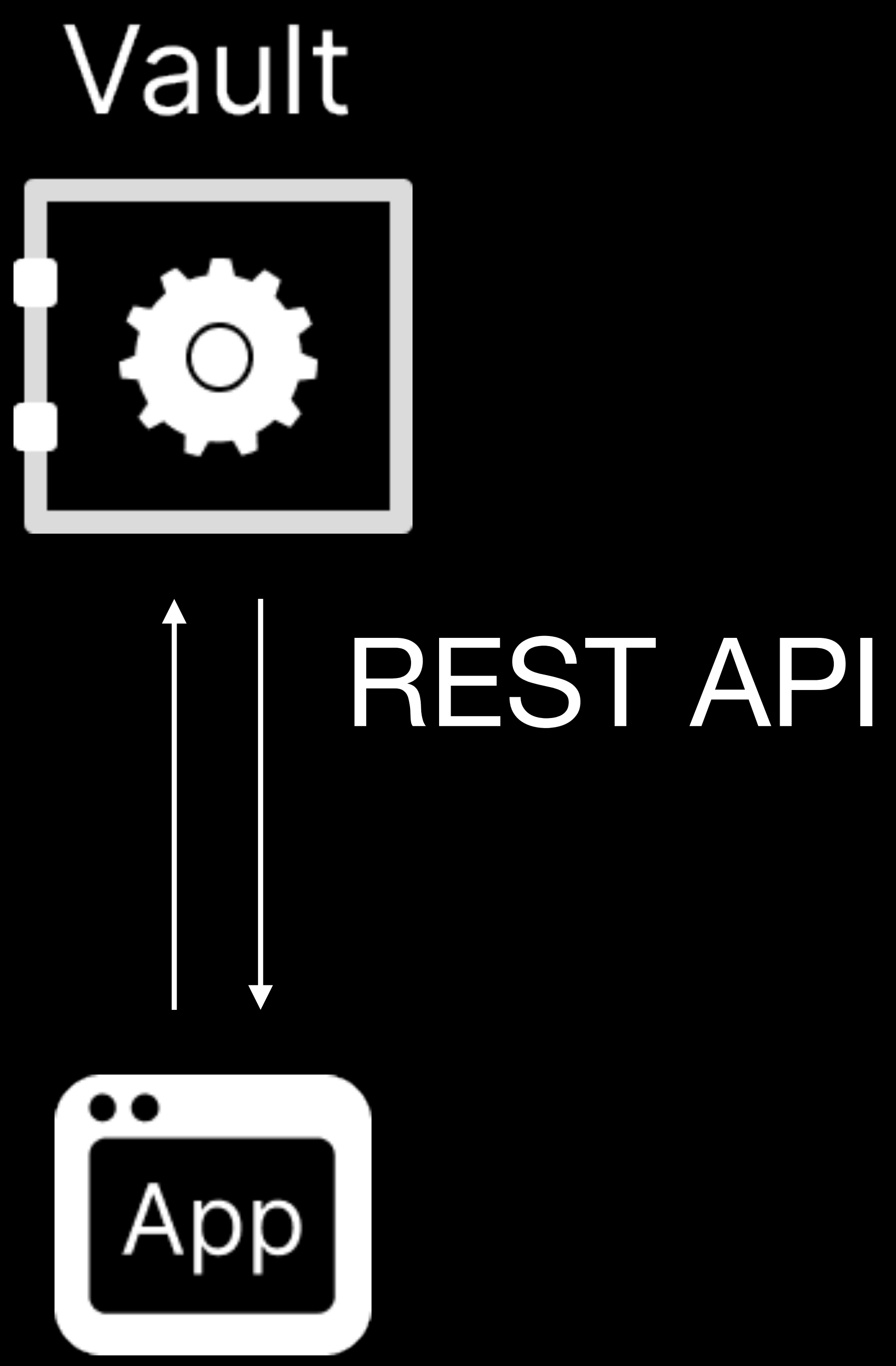
Vault

Vault

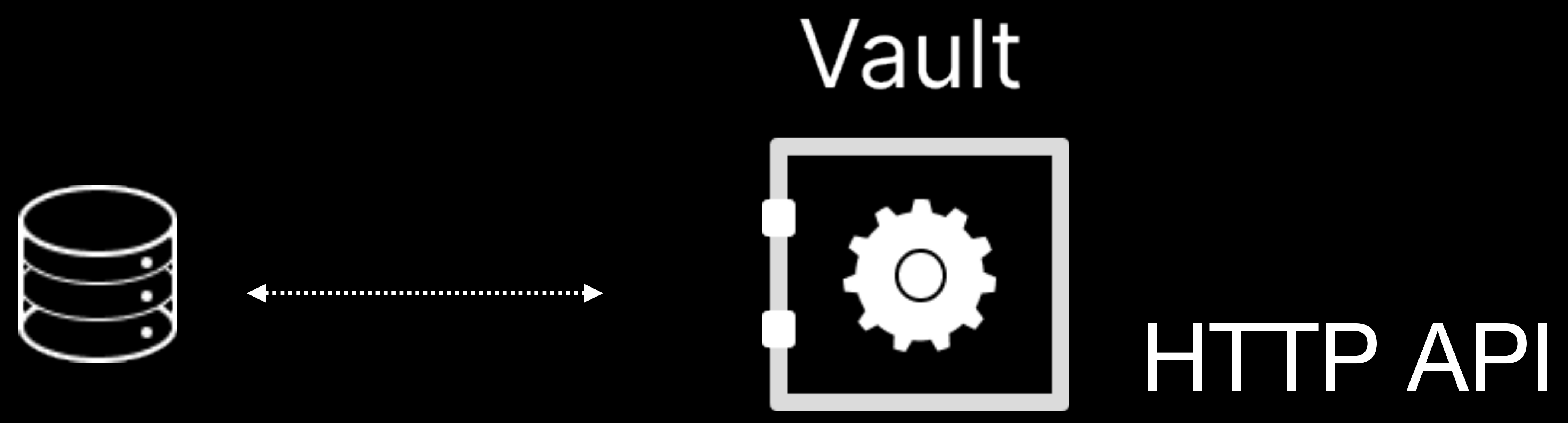
Vault



Vault



Vault



Manage 3rd Party Secrets

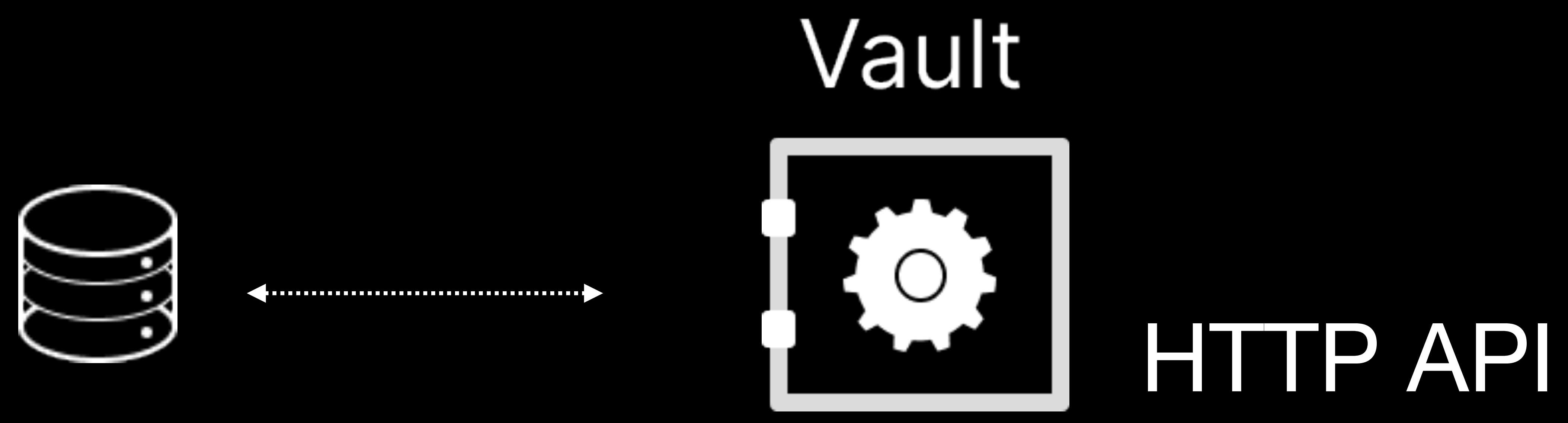
Generate, rotate and revoke Certificates

Encryption as a service

Manage Identities and Authentication

Multi-tenancy and fine-grained isolation

Vault



Manage 3rd Party Secrets

Generate, rotate and revoke Certificates

Encryption as a service

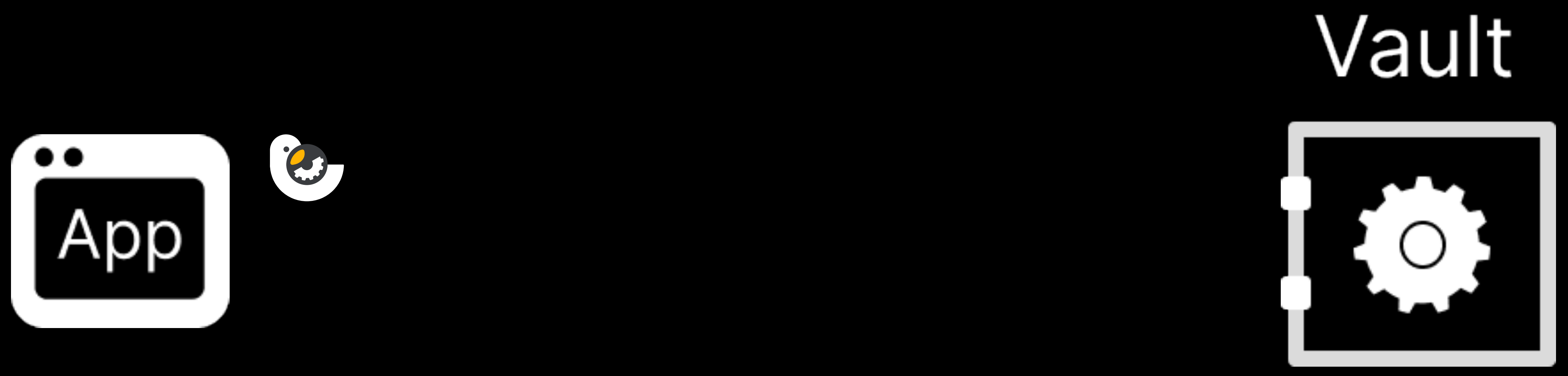
Manage Identities and Authentication

Multi-tenancy and fine-grained isolation

But, what about Swift?

Vault Courier

Native Swift client for OpenBao and HC-Vault



Swift 6 and OpenAPI: runs on Linux and Darwin

Manage 3rd Party Secrets: PostgreSQL and Valkey

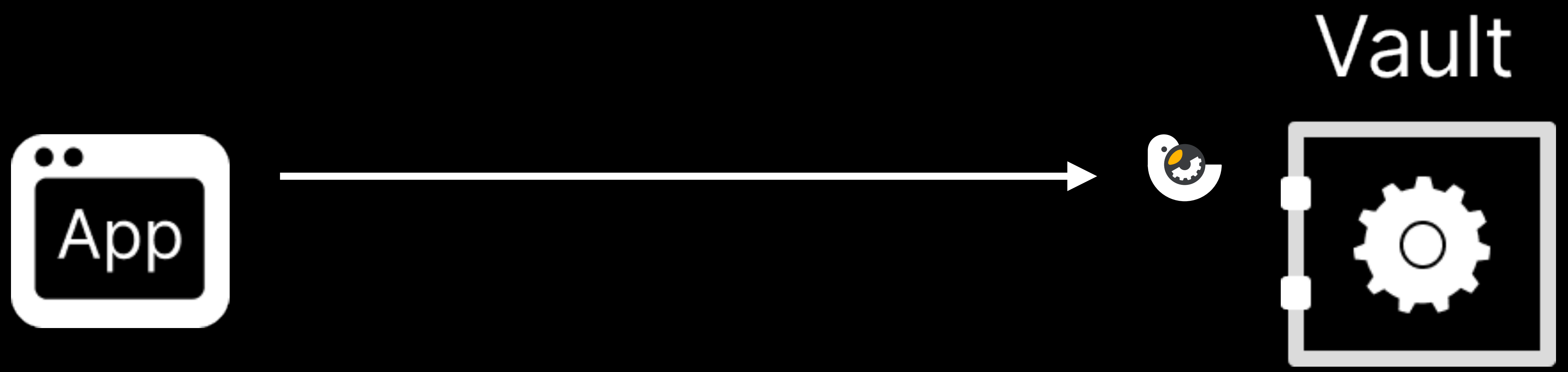
Static and dynamic secrets

Tracing support with swift-distributed-tracing

Experimental Configuration providers: Pkl and swift-configuration

Vault Courier

Native Swift client for OpenBao and HC-Vault



Swift 6 and OpenAPI: runs on Linux and Darwin

Manage 3rd Party Secrets: PostgreSQL and Valkey

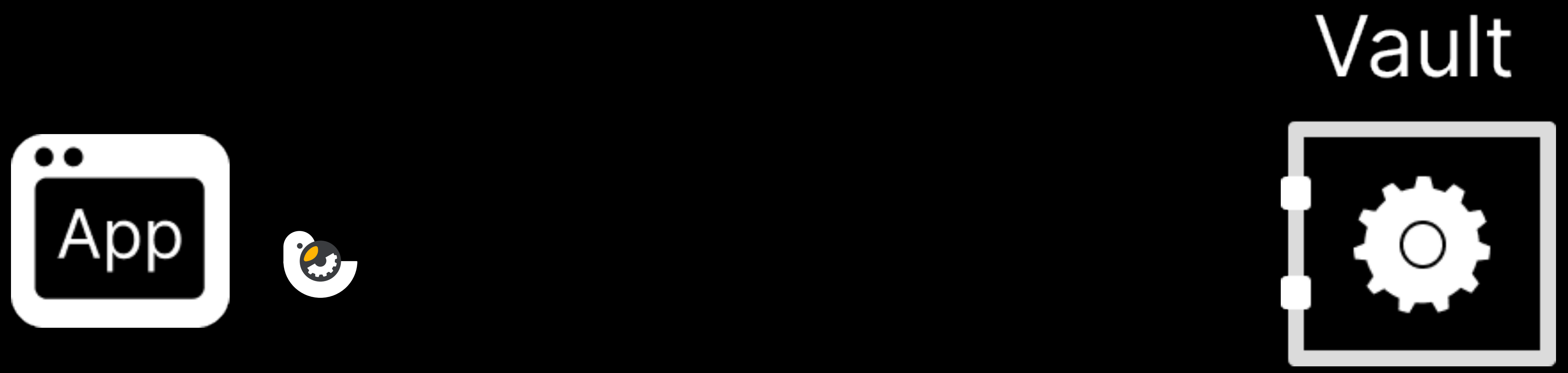
Static and dynamic secrets

Tracing support with swift-distributed-tracing

Experimental Configuration providers: Pkl and swift-configuration

Vault Courier

Native Swift client for OpenBao and HC-Vault



Swift 6 and OpenAPI: runs on Linux and Darwin

Manage 3rd Party Secrets: PostgreSQL and Valkey

Static and dynamic secrets

Tracing support with swift-distributed-tracing

Experimental Configuration providers: Pkl and swift-configuration

Part II
Lessons learned so far

First Lesson

Use swift-openapi-generator and **Package traits** SE-0450

Transport agnostic

Spec-driven development

API is documented

Much easier to have an overview of the API

Boilerplate reduced

This does not mean that there is no extra code after generating the client

Client mock available

Second Lesson:

Pkl enters the room

Found a treasure! 💎

Recommendation:
Consider using `org.openapis.v3` **Pkl package for writing complex OpenAPI specifications**

https://pkl-lang.org/package-docs/pkg.pkl-lang.org/pkl-pantry/org.openapis.v3/current/index.html

Second Lesson:

Pkl enters the room

Pkl: a modern, safe *configuration* language.
Renders into “raw-formats” like JSON and YAML

Benefits of using openapis.v3 Pkl package:

1. Modularize the specification

```
paths {
  ...PoliciesACL.paths

  ...Mounts.paths

  ...SystemAuth.paths

  ...Wrapping.paths

  ...TokenAuth.paths

  ...AppRoleAuth.paths

  ...KeyValueEngine.paths

  ...PostgresDatabaseConfiguration.paths
}
```

```
paths {
  ["/auth/token/create"] {
    description = "The token create path is used to create new tokens."
    parameters {
      new Reference {
        `$ref` = "#/components/parameters/VaultTokenHeader"
      }
      new Reference {
        `$ref` = "#/components/parameters/WrapTTLHeader"
      }
    }
    post {
      operationId = "token-create"
      tags { “auth" }
      requestBody {
        content {
          ["application/json"] {
            schema = new Reference {
              `$ref` = "#/components/schemas/TokenCreateRequest"
            }
          }
        }
        required = true
      }

     // …

    }
```

Benefits of using openapis.v3 Pkl package:

2. Help with verification. You can snapshot-test the output

Benefits of using openapis.v3 Pkl package:

2. Help with verification. You can snapshot-test the output

3. You can introduce Objects/Pkl modules to simplify almost identical types.
“Amending” is a powerful mechanism here

```
tokenCommon = (Schema.PropertySchema) {
  type = "object"
  properties {
    ["token_bound_cidrs"] {
      type = "array"
      description = "List of CIDR blocks"
      items = new Schema {
        type = "string"
      }
    }
  }
}

createAppRole = (tokenCommon) {
  properties {
    ["token_ttl"] {
      type = "string"
      description = "The incremental lifetime for generated tokens"
    }
  }
}
```

Benefits of using openapis.v3 Pkl package:

2. Help with verification. You can snapshot-test the output

3. You can introduce Objects/Pkl modules to simplify almost identical types. “Amending” is a powerful mechanism here

```
tokenCommon = (Schema.PropertySchema) {
  type = "object"
  properties {
    ["token_bound_cidrs"] {
      type = "array"
      description = "List of CIDR blocks"
      items = new Schema {
        type = "string"
      }
    }
  }
}

createAppRole = (tokenCommon) {
  properties {
    ["token_ttl"] {
      type = "string"
      description = "The incremental lifetime for generated tokens"
    }
  }
}
```

How can we make it easier to access Vault secrets from configuration?

Third Lesson:

Use Pkl-bindings for integrating complex
Configuration and external resources

```
amends "ServerConfig.pkl"

logLevel = "debug"

database {
  hostname = "localhost"

  port = 8080
}
```

https://github.com/apple/pkl-swift

Third Lesson:

Use Pkl-bindings for integrating complex
Configuration and external resources

```
amends "ServerConfig.pkl"

logLevel = "debug"

apiToken = read(“vault.secrets:api_token?version=2”).text

database {
  hostname = "localhost"

  port = 8080

  credentials = read(“vault.database:static-creds/server_role”).text
}
```

https://github.com/apple/pkl-swift

Third Lesson:

Use Pkl-bindings for integrating complex Configuration and external resources

```
amends "ServerConfig.pkl"

logLevel = "debug"

apiToken = read(“vault.secrets:api_token?version=2”).text

database {
  hostname = "localhost"

  port = 8080

  credentials = read(“vault.database:static-creds/server_role”).text
}
```

https://github.com/apple/pkl-swift

Found another use-case for Pkl 💎

Adopting Pkl as native config format allows to read resources specified by URI schemas

In Pkl:

```
property = read("schema:path?query")
```

In Swift

```
extension VaultResourceReader: ResourceReader {
    public func read(url: URL) async throws -> [UInt8] {
      // …
    }
}
```

https://github.com/apple/pkl-swift

Check VaultCourier

Repository

Tutorials

Visit OpenBao/OpenTofu Stand at K1-C-06