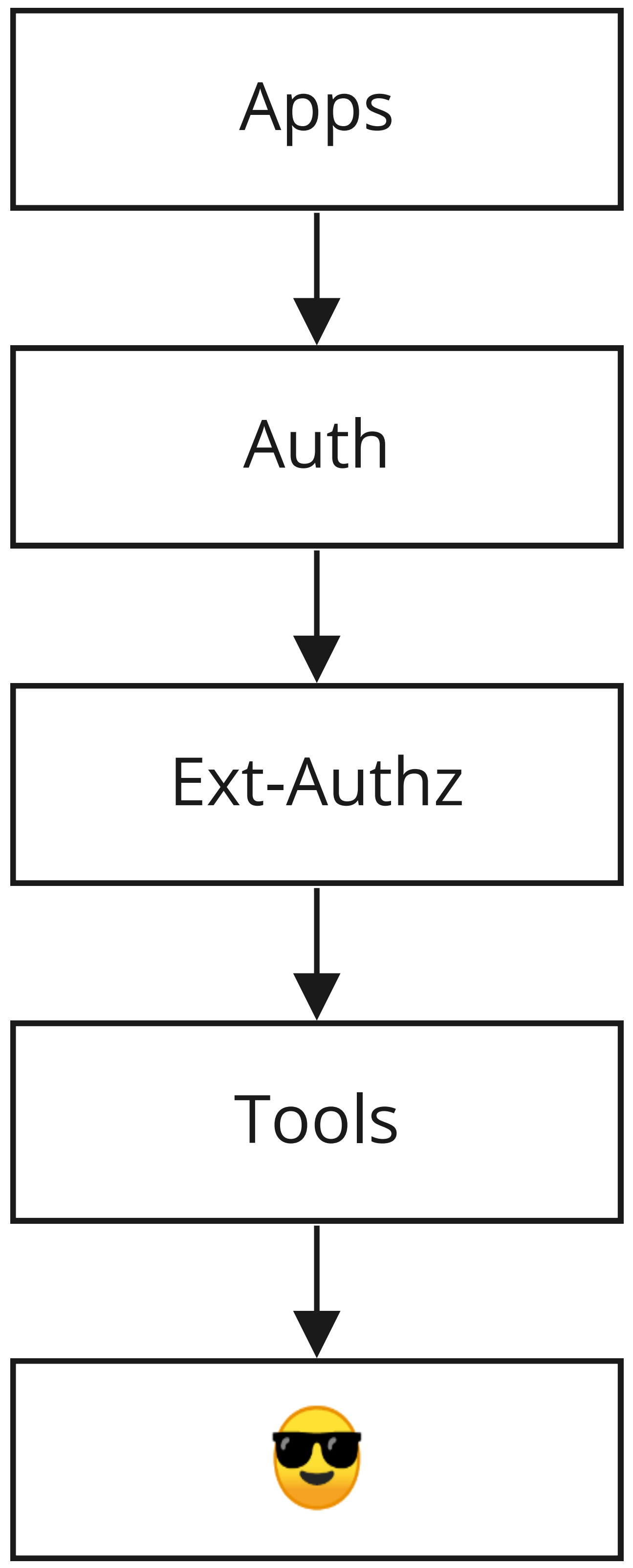
Thursday 8/12

DEVSTAFF
COMMUNITY

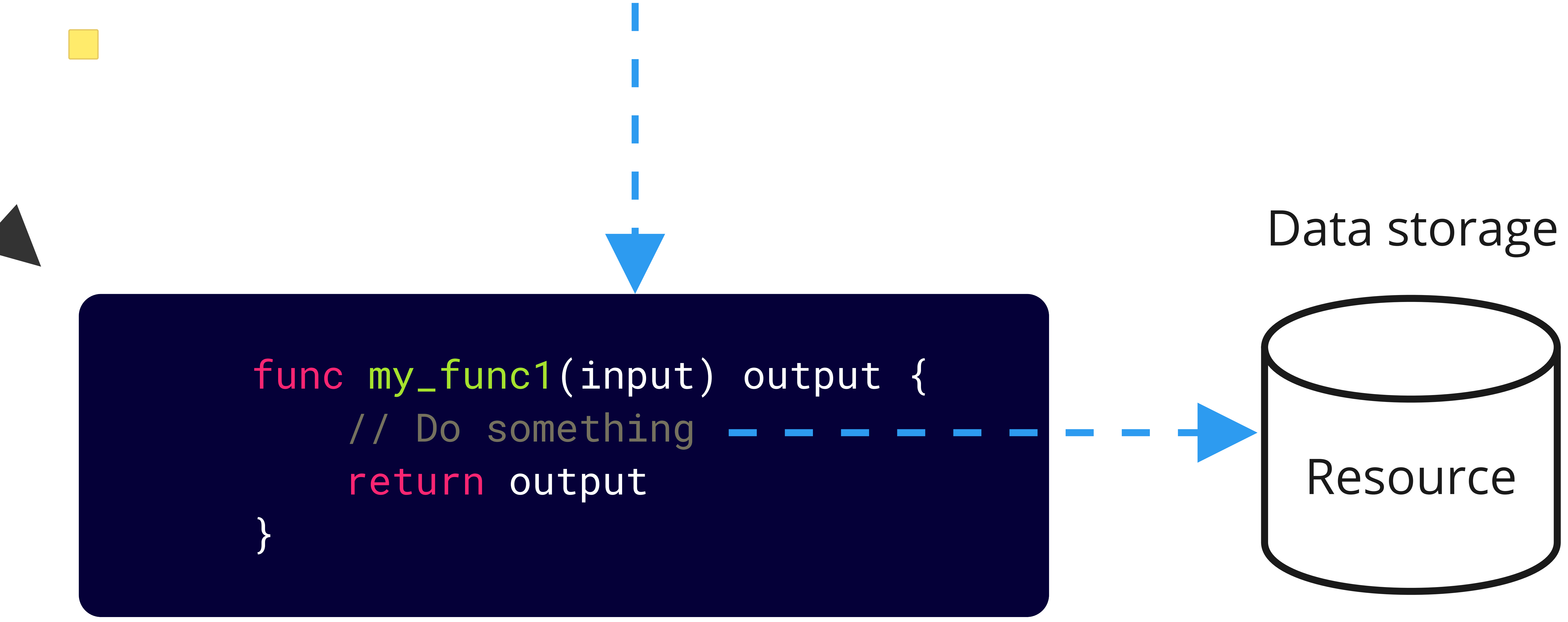
Authentication & Authorization

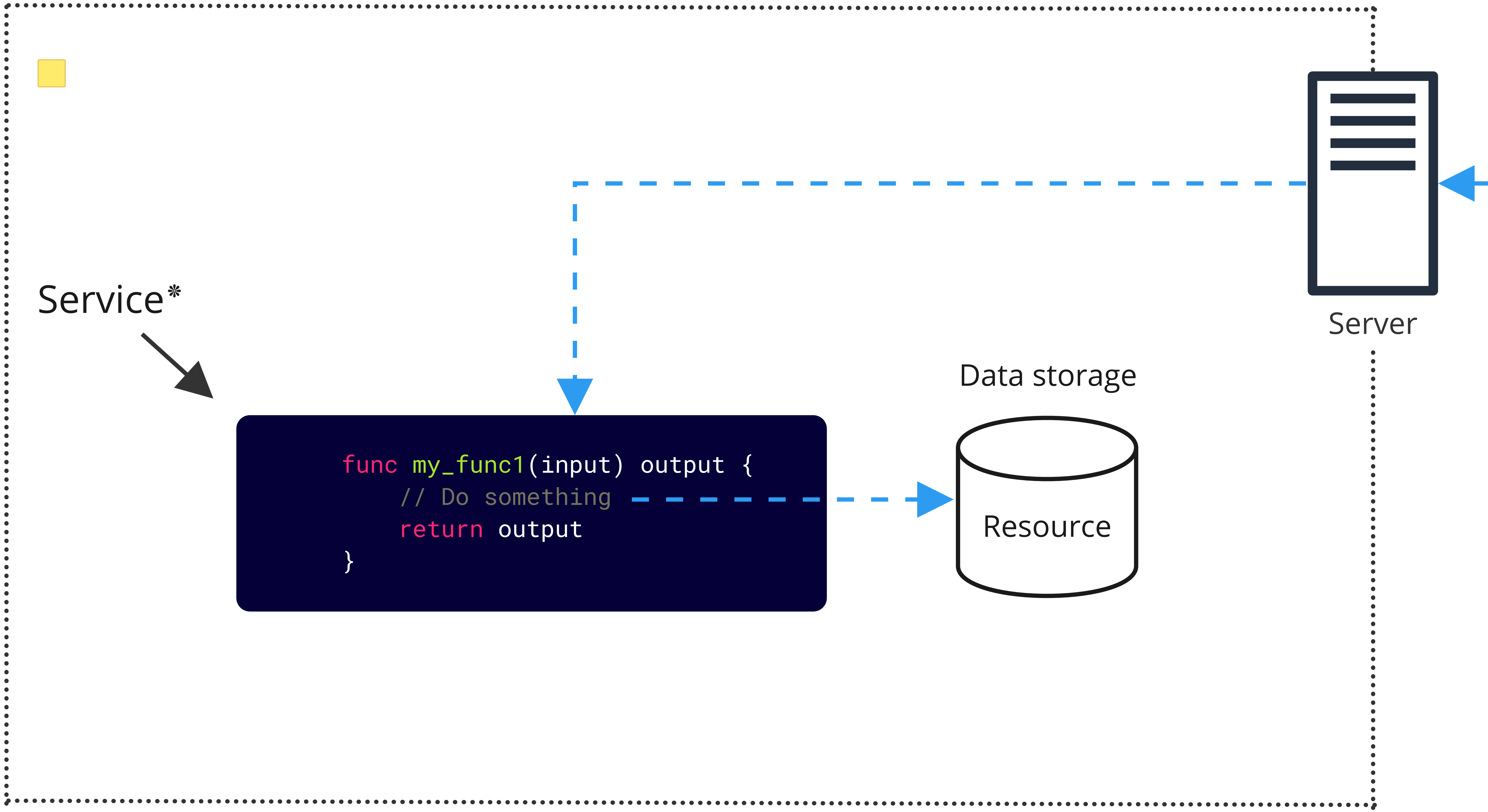
Intro to AuthN/Z & External AuthZ to the rescue

Guilherme Cassolato (github.com/guicassolato)

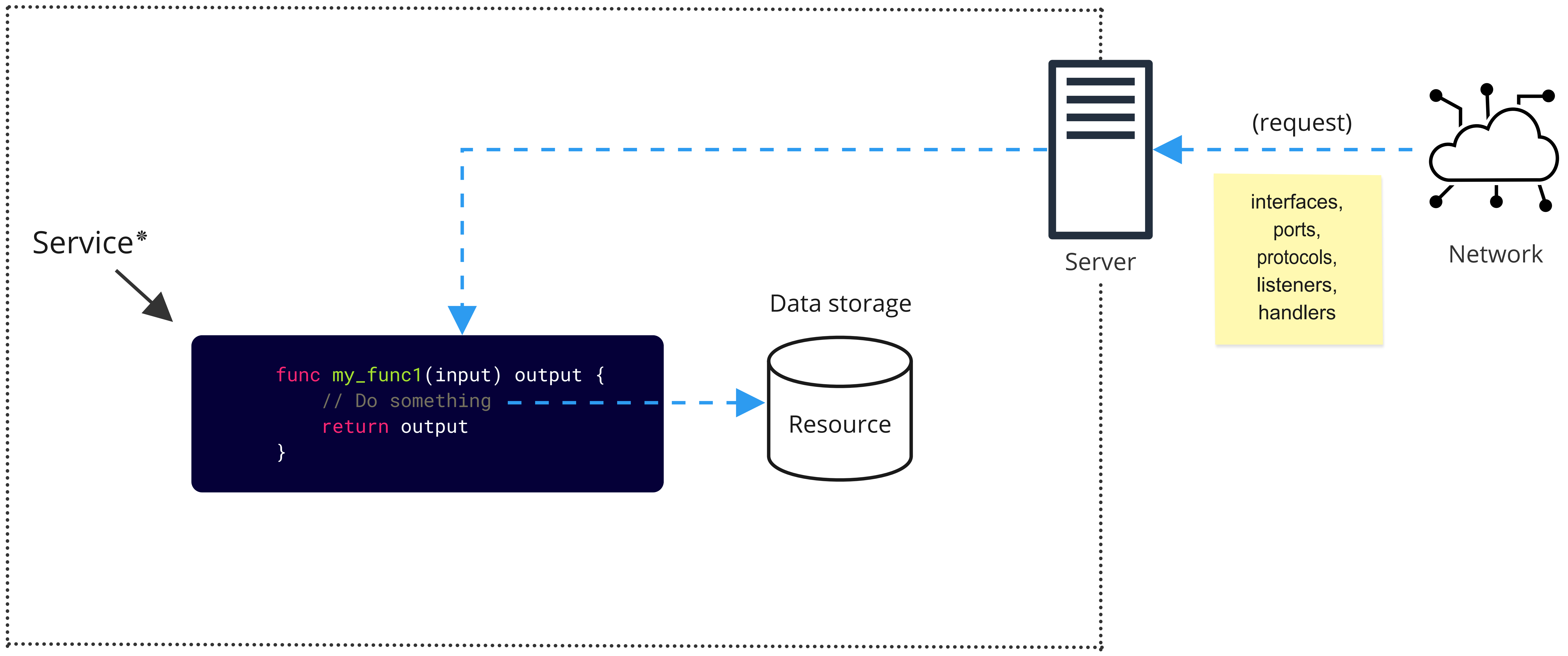

Apps
Auth
Ext-Authz
Tools
😎

```
func my_func1(input) output {
    // Do something
    return output
}
```

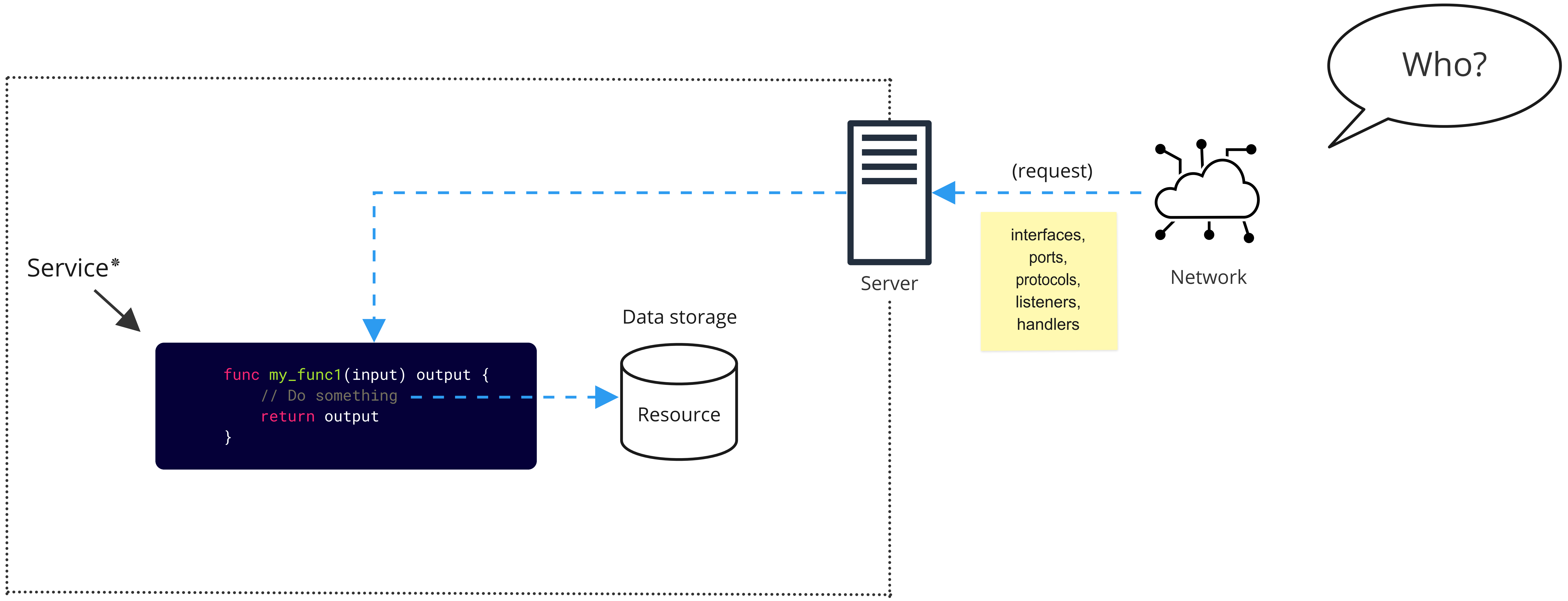

```
func my_func1(input) output {
    // Do something
    return output
}
```



```
func my_func1(input) output {
    // Do something
    return output
}
```



```
func my_func1(input) output {
    // Do something
    return output
}
```



```
func my_func1(input) output {
    // Do something
    return output
}
```

```
input = input + "&username=%{username}"
input = input + "&username=%{username}"
              + "&password=%{secret_password}"
```

```
func my_func1(input) output {
    user = sql.exec(`SELECT * FROM users WHERE username='%{input["username"]}' AND password=salted('%{input["password"]}', salt)`)
    if user == nil {
        return error(401, "authentication error")
    }

    // Do something
    return success(output)
}
```

username+ password ("Basic Auth")

API Keys

JSON Web Tokens (JWT)

OAuth2

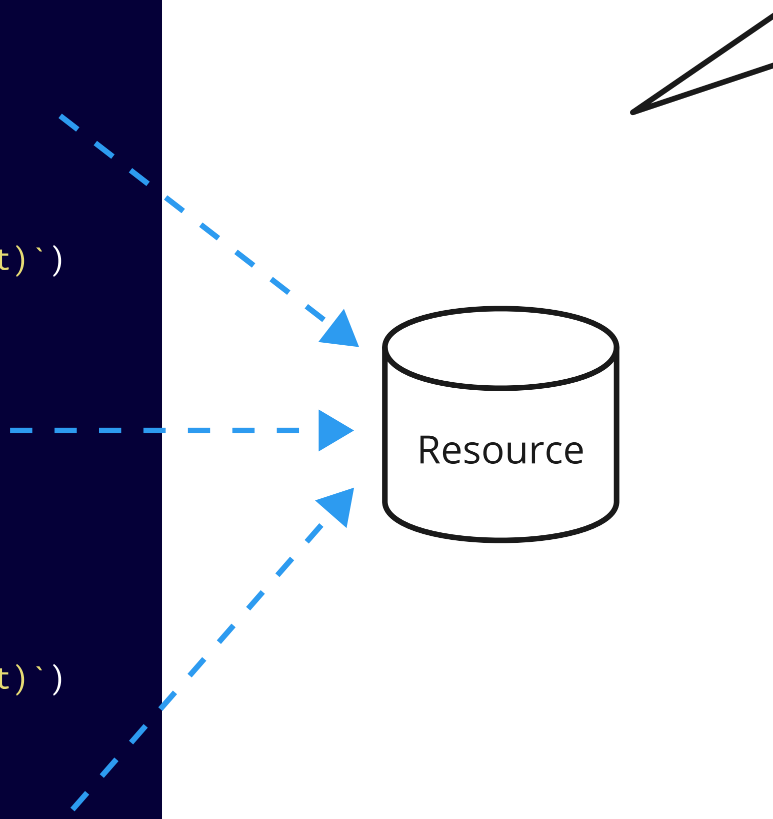
OpenId Connect (OIDC)

x.509 certificates (TLS certs)

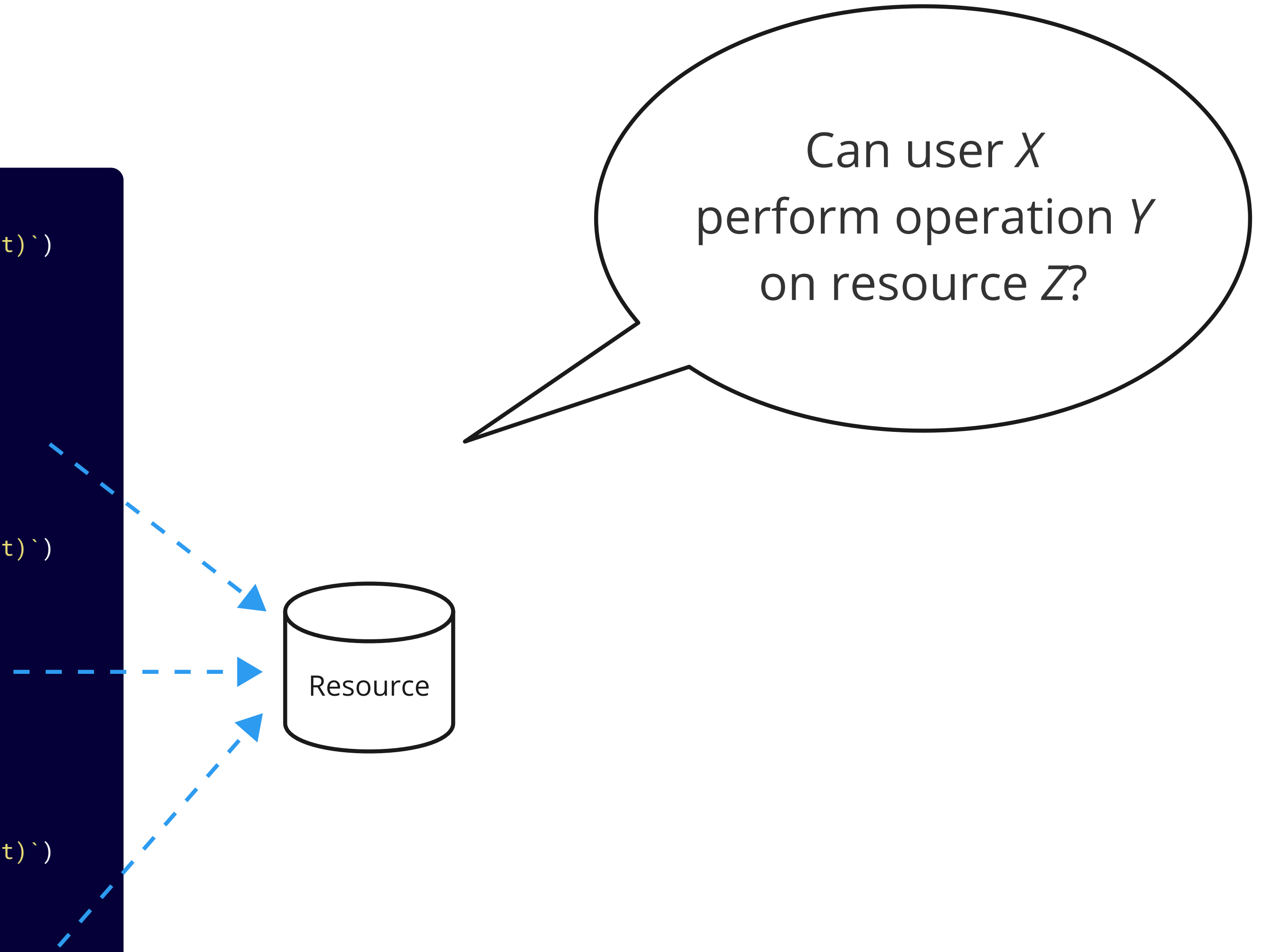
Hash Message Authentication Code (HMAC)

```
func my_func1(input) output {
    user = sql.exec(`SELECT * FROM users WHERE username='%{input["username"]}' AND password=salted('%{input["password"]}', salt)`)
    if user == nil {
        return error(401, "authentication error")
    }

    // Do something
    return success(output)
}

func my_func2(input) output {
    user = sql.exec(`SELECT * FROM users WHERE username='%{input["username"]}' AND password=salted('%{input["password"]}', salt)`)
    if user == nil {
        return error(401, "authentication error")
    }

    // Do something else
    return success(output)
}

func my_func3(input) output {
    user = sql.exec(`SELECT * FROM users WHERE username='%{input["username"]}' AND password=salted('%{input["password"]}', salt)`)
    if user == nil {
        return error(401, "authentication error")
    }

    // Do yet something else
    return success(output)
}
```

```
func my_func1(input) output {
    user = sql.exec(`SELECT * FROM users WHERE username='%{input["username"]}' AND password=salted('%{input["password"]}', salt)`)
    if user == nil {
        return error(401, "authentication error")
    }

    // Do something
    return success(output)
}

func my_func2(input) output {
    user = sql.exec(`SELECT * FROM users WHERE username='%{input["username"]}' AND password=salted('%{input["password"]}', salt)`)
    if user == nil {
        return error(401, "authentication error")
    }

    // Do something else
    return success(output)
}

func my_func3(input) output {
    user = sql.exec(`SELECT * FROM users WHERE username='%{input["username"]}' AND password=salted('%{input["password"]}', salt)`)
    if user == nil {
        return error(401, "authentication error")
    }

    // Do yet something else
    return success(output)
}
```



```
func my_func1(input) output {
    user = sql.exec(`SELECT * FROM users WHERE username='%{input["username"]}' AND password=salted('%{input["password"]}', salt)`)
    if user == nil {
        return error(401, "authentication error")
    }

    // Do something
    return success(output)
}

func my_func2(input) output {
    user = sql.exec(`SELECT * FROM users WHERE username='%{input["username"]}' AND password=salted('%{input["password"]}', salt)`)
    if user == nil {
        return error(401, "authentication error")
    }

    // Do something else
    return success(output)
}

func my_func3(input) output {
    user = sql.exec(`SELECT * FROM users WHERE username='%{input["username"]}' AND password=salted('%{input["password"]}', salt)`)
    if user == nil {
        return error(401, "authentication error")
    }

    // Do yet something else
    return success(output)
}
```



```
func my_func1(input) output {
    user = sql.exec(`SELECT * FROM users WHERE username='%{input["username"]}' AND password=salted('%{input["password"]}', salt)`)
    if user == nil {
        return error(401, "authentication error")
    }

    permissions = sql.exec(`SELECT * FROM permissions WHERE username=%{input["username"]} AND operation='op1'`)
    if permissions == nil {
        return error(403, "forbidden")
    }

    // Do something
    return success(output)
}

func my_func2(input) output {
    user = sql.exec(`SELECT * FROM users WHERE username='%{input["username"]}' AND password=salted('%{input["password"]}', salt)`)
    if user == nil {
        return error(401, "authentication error")
    }

    permissions = sql.exec(`SELECT * FROM permissions WHERE username=%{input["username"]} AND operation='op2'`)
    if permissions == nil {
        return error(403, "forbidden")
    }

    // Do something else
    return success(output)
}

func my_func3(input) output {
    user = sql.exec(`SELECT * FROM users WHERE username='%{input["username"]}' AND password=salted('%{input["password"]}', salt)`)
    if user == nil {
        return error(401, "authentication error")
    }

    permissions = sql.exec(`SELECT * FROM permissions WHERE username=%{input["username"]} AND operation='op3'`)
    if permissions == nil {
        return error(403, "forbidden")
    }

    // Do yet something else
    return success(output)
}
```

Access Control List (ACL)

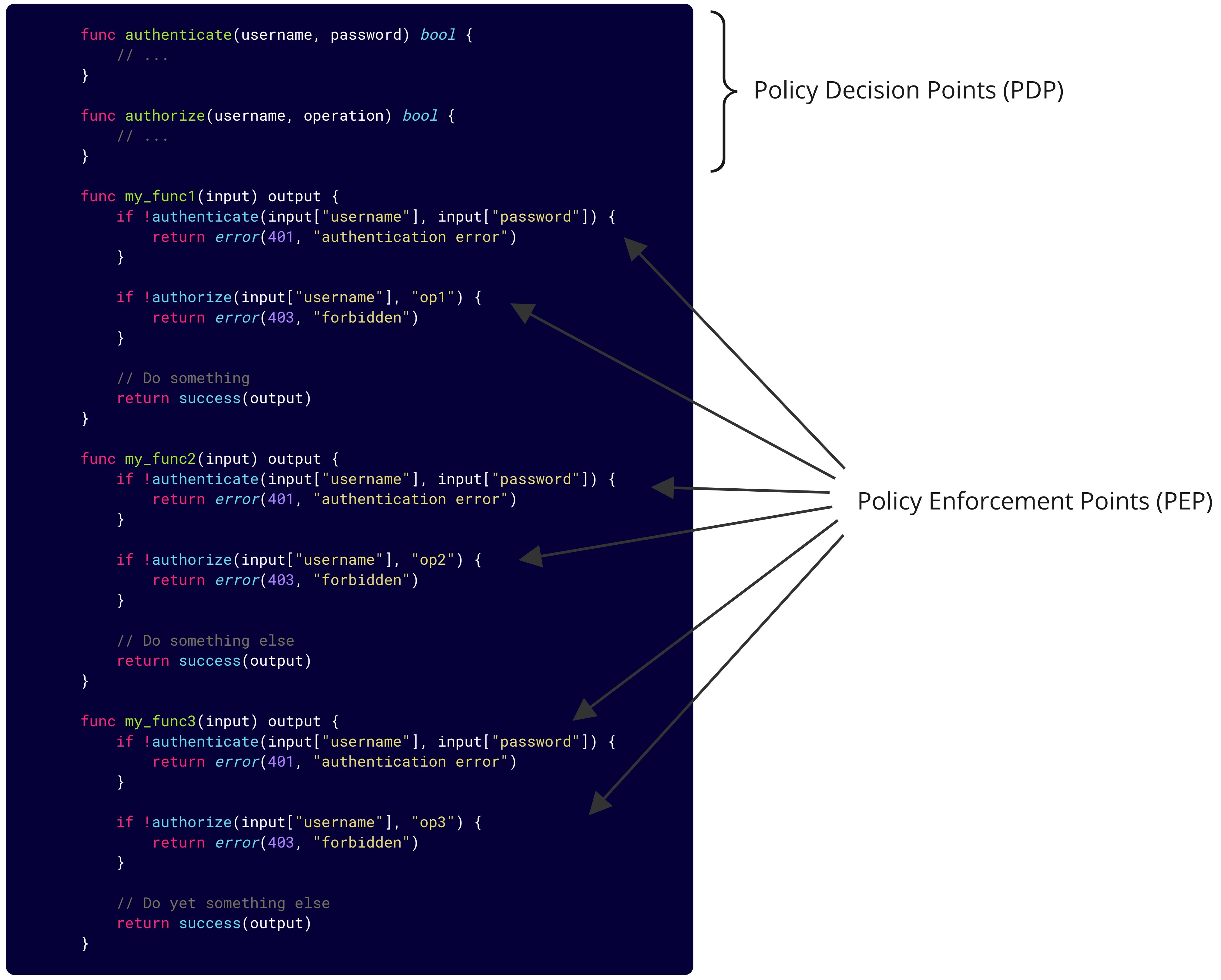
Role-Based Access Control (RBAC)

Attribute-Based Access Control (ABAC)

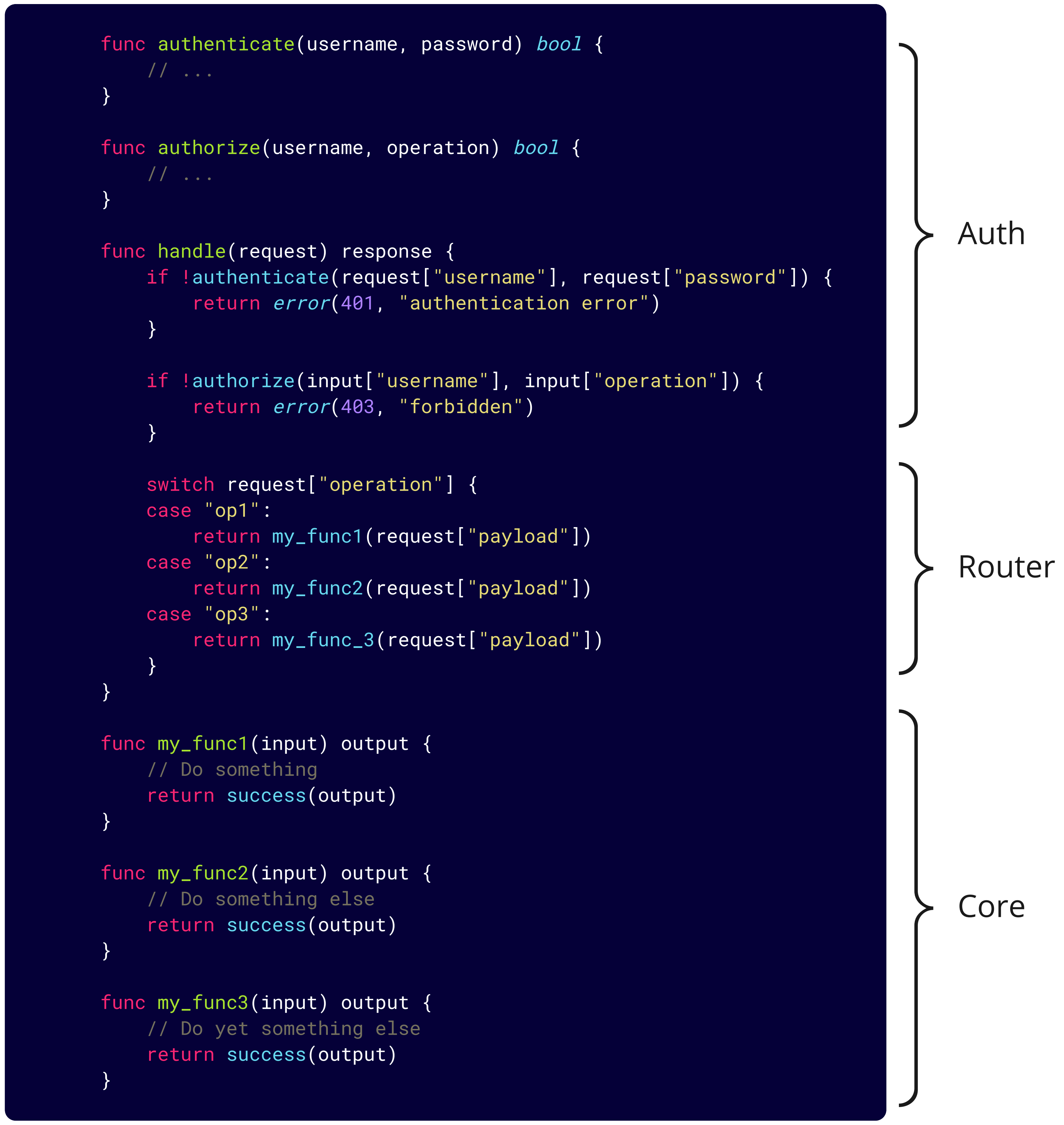
Relationship-Based Access Control (ReBAC)

Time-Based Access Control (TBAC)

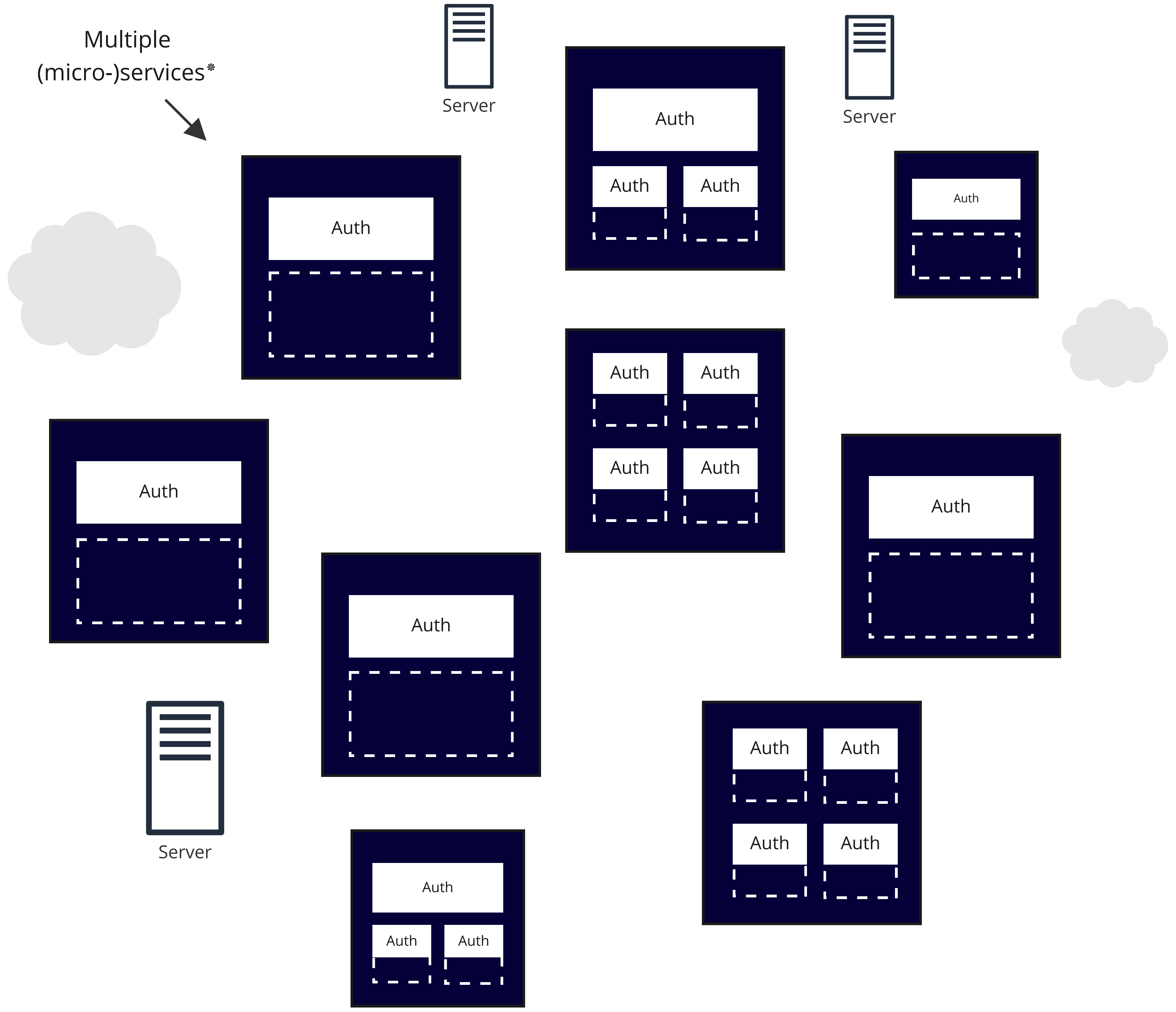
Context-Based Access Control (CBAC)

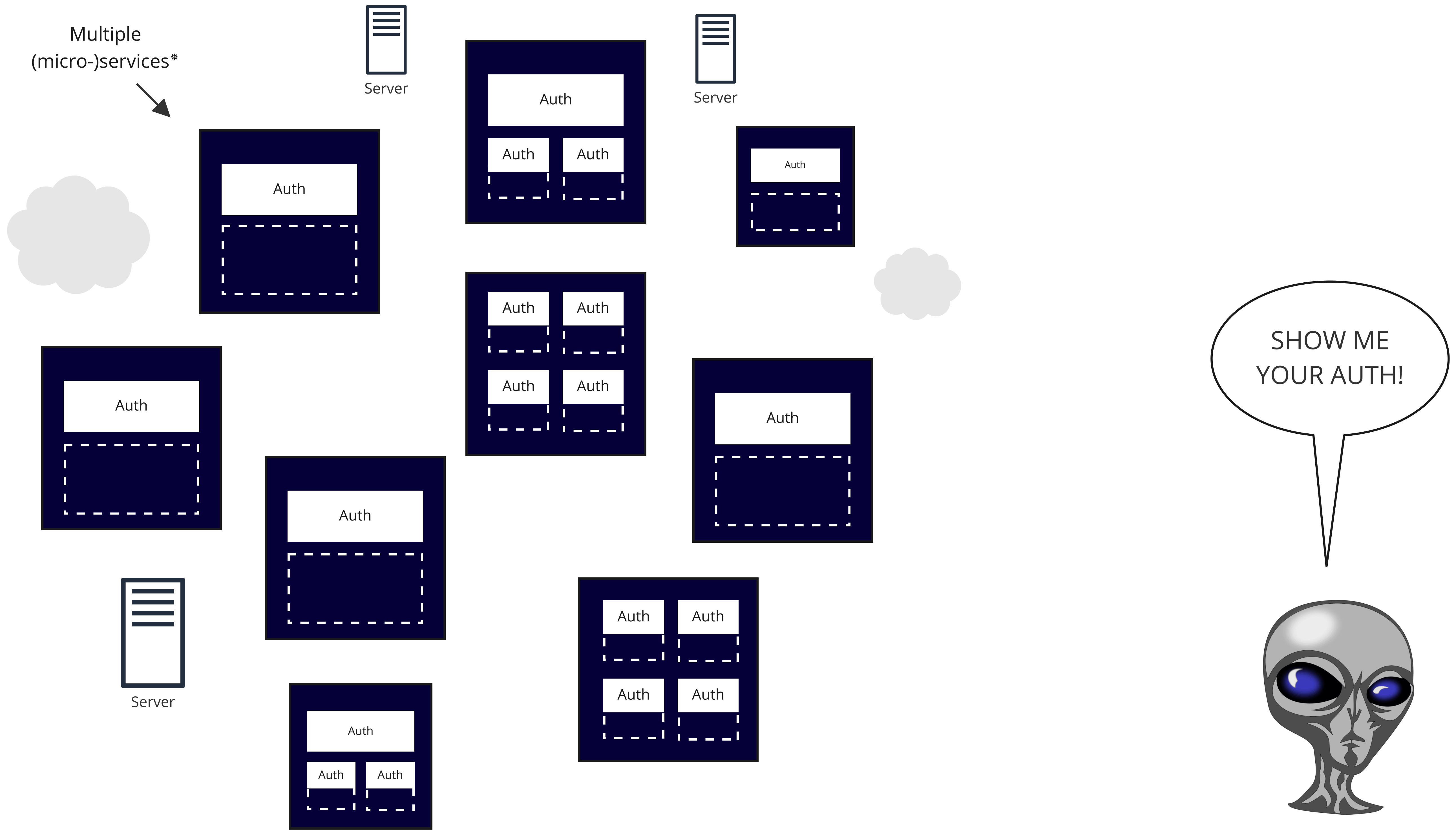


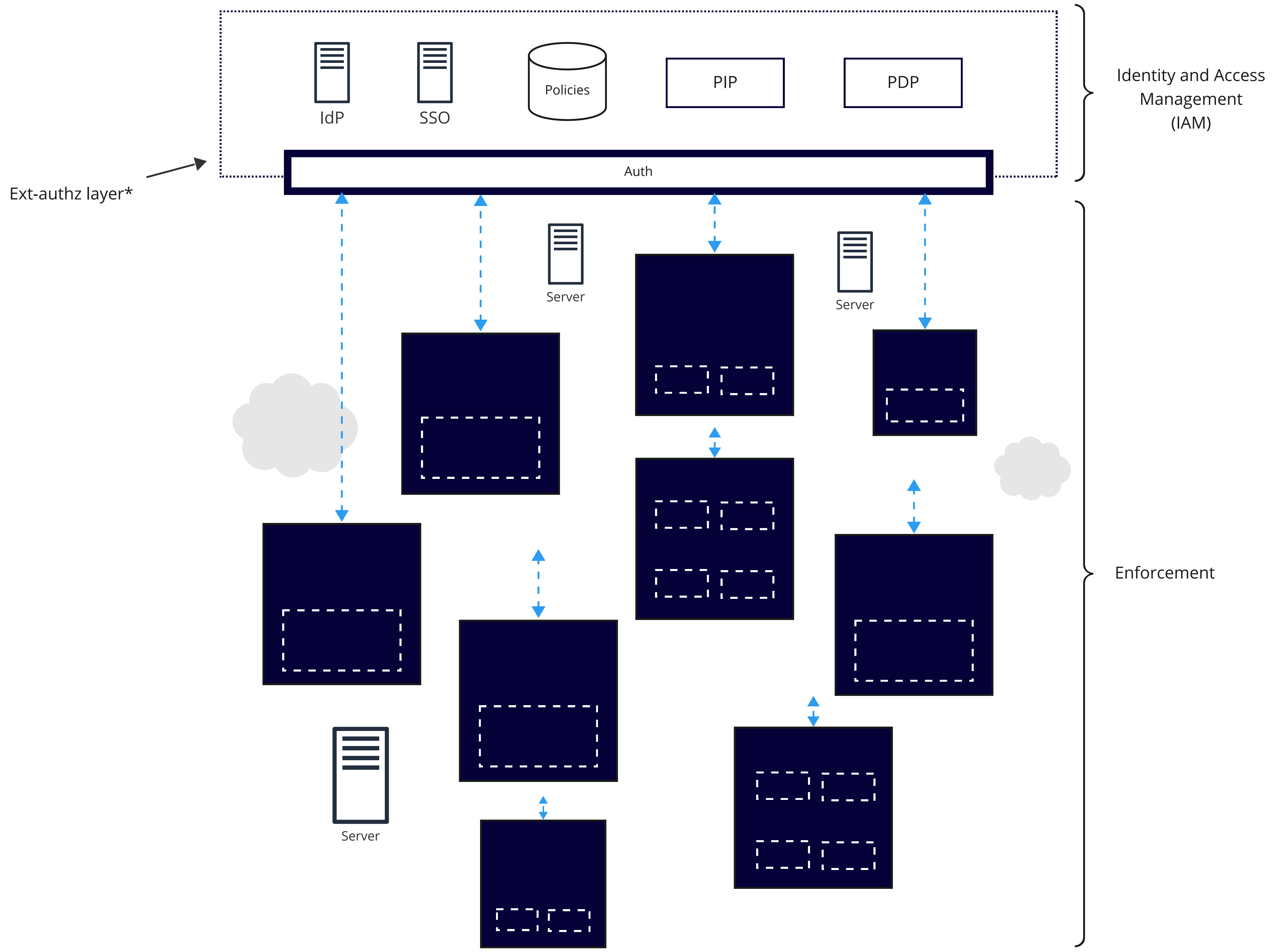
```
func authenticate(username, password) bool {
    // ...
}

func authorize(username, operation) bool {
    // ...
}

func my_func1(input) output {
    if !authenticate(input["username"], input["password"]) {
        return error(401, "authentication error")
    }

    if !authorize(input["username"], "op1") {
        return error(403, "forbidden")
    }

    // Do something
    return success(output)
}

func my_func2(input) output {
    if !authenticate(input["username"], input["password"]) {
        return error(401, "authentication error")
    }

    if !authorize(input["username"], "op2") {
        return error(403, "forbidden")
    }

    // Do something else
    return success(output)
}

func my_func3(input) output {
    if !authenticate(input["username"], input["password"]) {
        return error(401, "authentication error")
    }

    if !authorize(input["username"], "op3") {
        return error(403, "forbidden")
    }

    // Do yet something else
    return success(output)
}
```

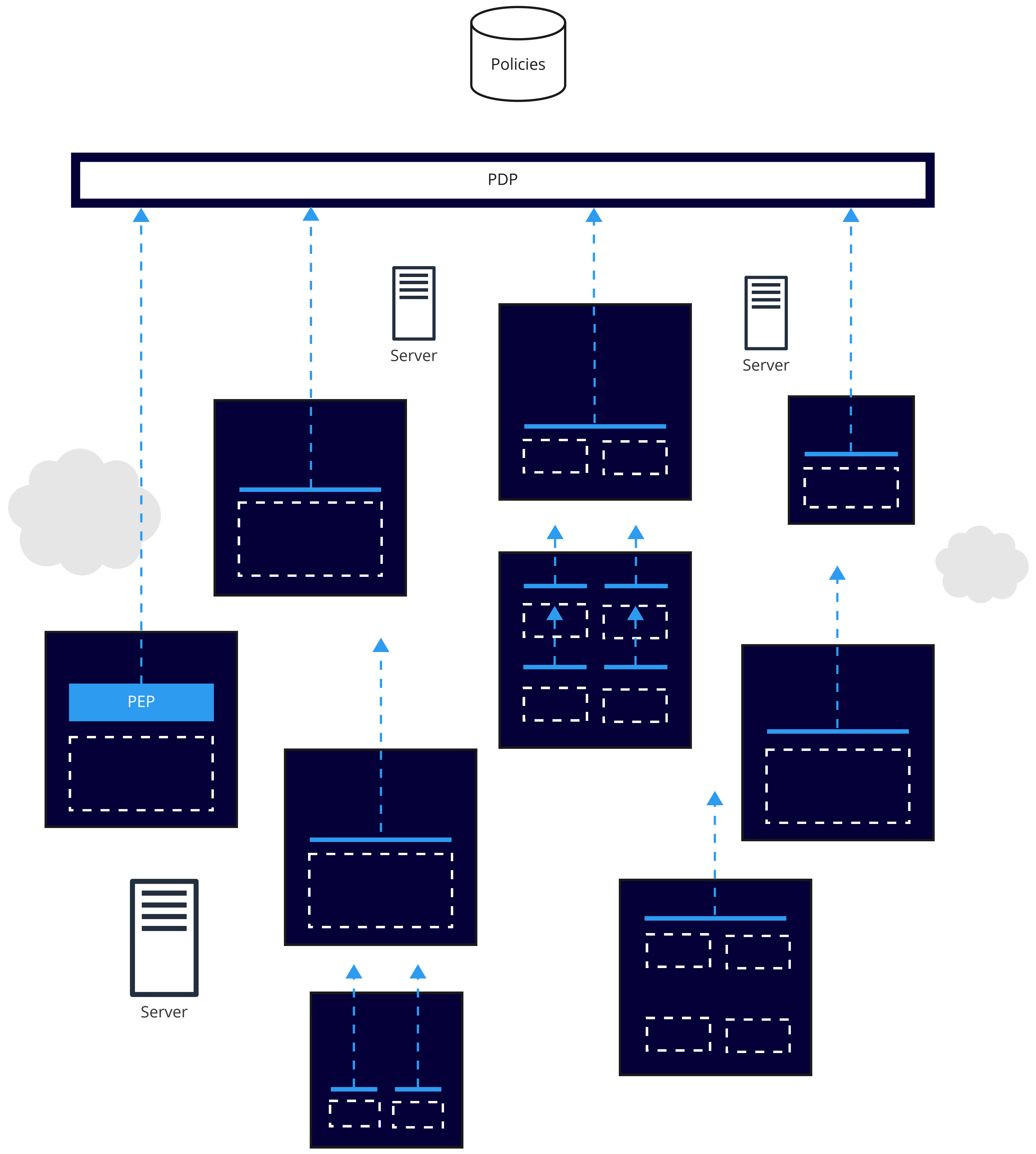


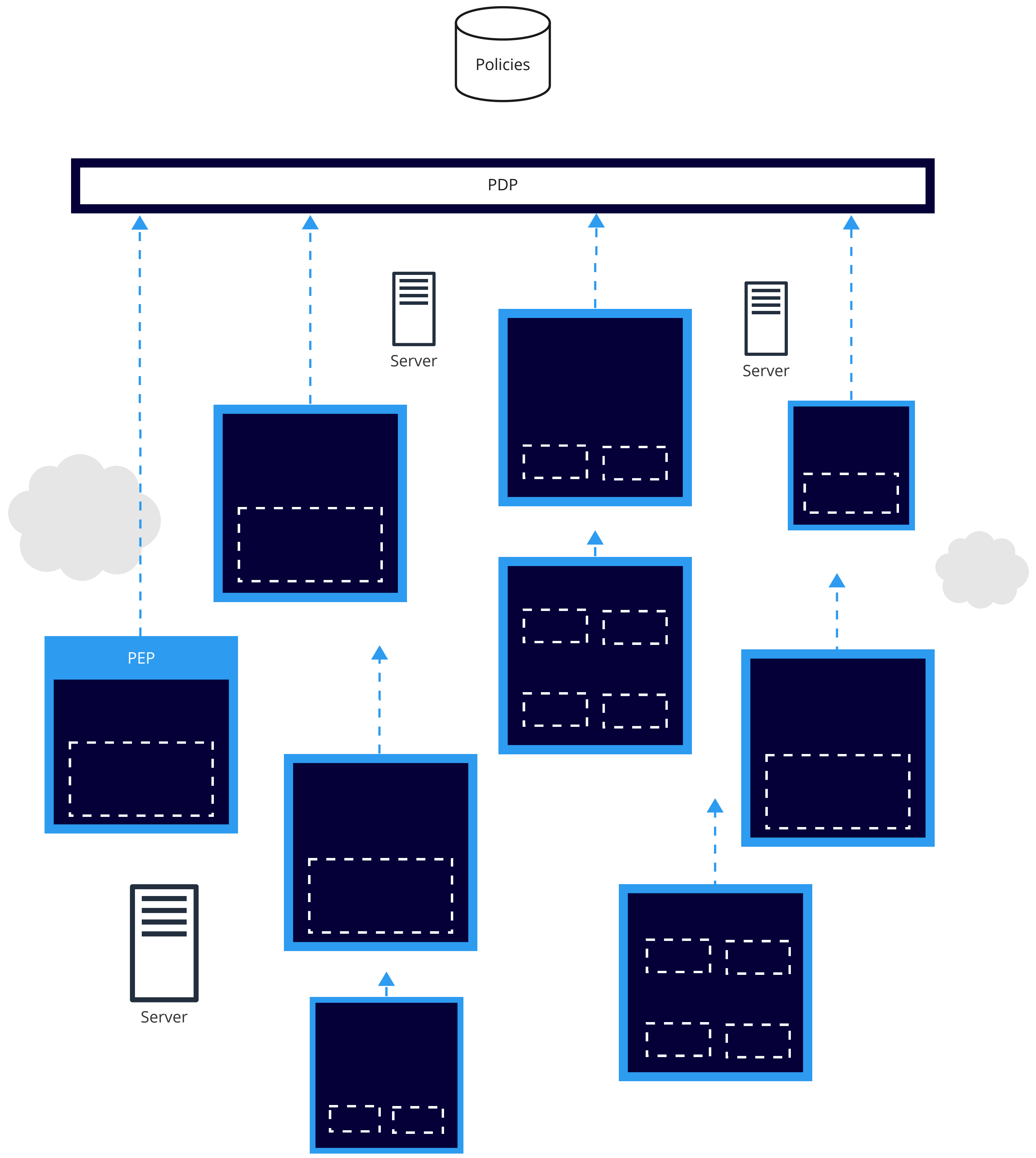
```
func authenticate(username, password) bool {
    // ...
}

func authorize(username, operation) bool {
    // ...
}

func handle(request) response {
    if !authenticate(request["username"], request["password"]) {
        return error(401, "authentication error")
    }

    if !authorize(input["username"], input["operation"]) {
        return error(403, "forbidden")
    }

    switch request["operation"] {
    case "op1":
        return my_func1(request["payload"])
    case "op2":
        return my_func2(request["payload"])
    case "op3":
        return my_func_3(request["payload"])
    }
}

func my_func1(input) output {
    // Do something
    return success(output)
}

func my_func2(input) output {
    // Do something else
    return success(output)
}

func my_func3(input) output {
    // Do yet something else
    return success(output)
}
```

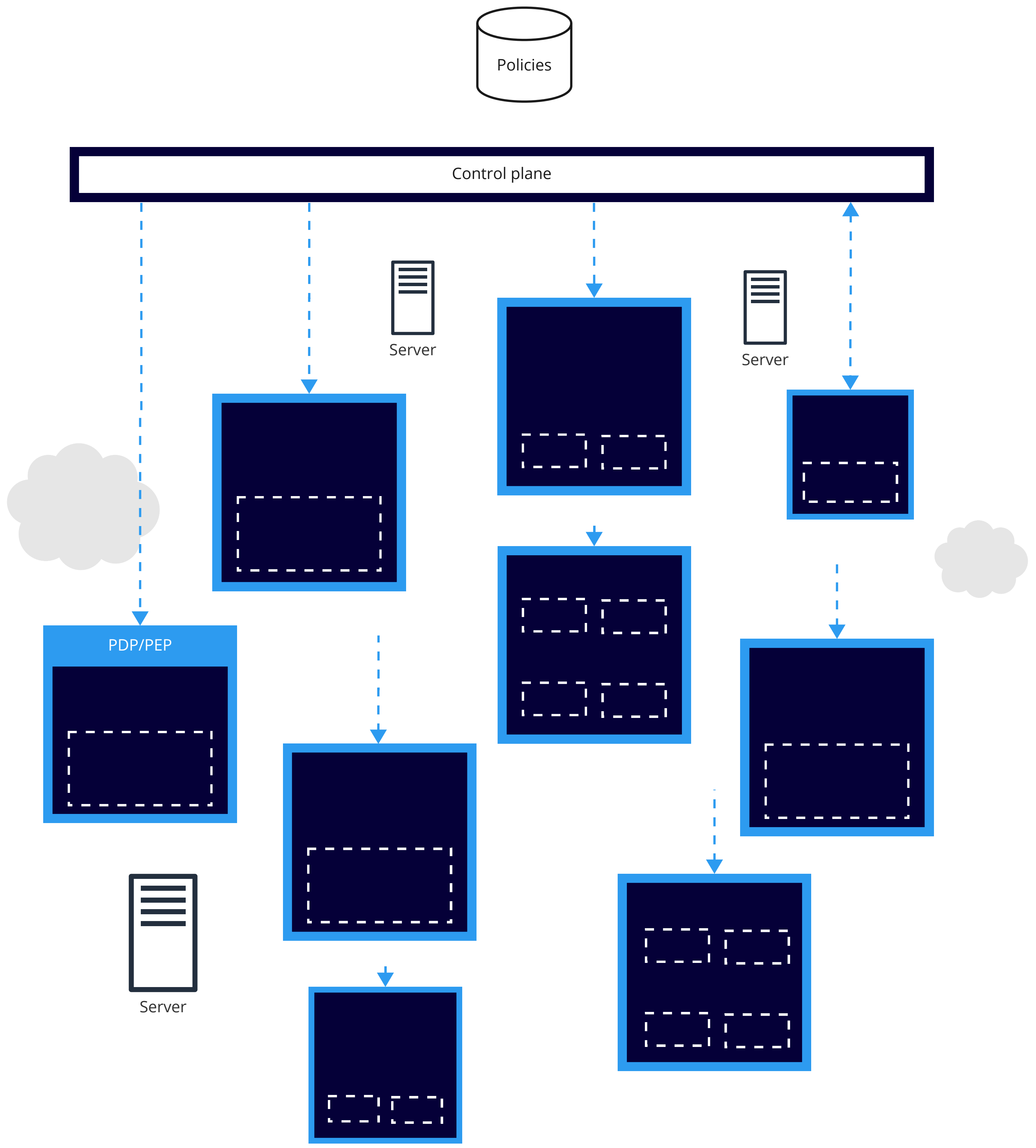

Multiple
(micro-)services*
Server
Server
Server
Auth
Auth
Auth
Auth
Auth
Auth
Auth
Auth
Auth
Auth
Auth
Auth
Auth
Auth
Auth
Auth
Auth
Auth
Auth
Auth
Auth


Multiple (micro-)services*
Server
Server
Server
Auth
SHOW ME YOUR AUTH!


IdP
SSO
Policies
PIP
PDP
Auth
Ext-authz layer*
Identity and Access Management (IAM)
Server
Server
Server
Enforcement


Policies
PDP
Server
Server
PEP
Server


Policies
PDP
Server
Server
PEP
Server



Separation of concerns

Decoupling (code, scalability)

DRY

Flexibility

Standardization

Governance

Best practices

Protocols

AWARDS 2017
nickelodeon
KIDS' CHOICE AWARDS 2017
THE TIME IS NOW!

Platform

AuthN

RBAC

kubernetes

Connectivity

Extensibility

Transparency

envoy

Istio

Control plane

Ubiquity

Open Policy Agent

language