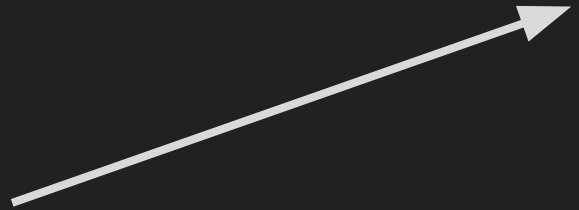
# Customizing OPA

For a “Perfect Fit” Authorization Sidecar

# Patrick East



OPA Maintainer

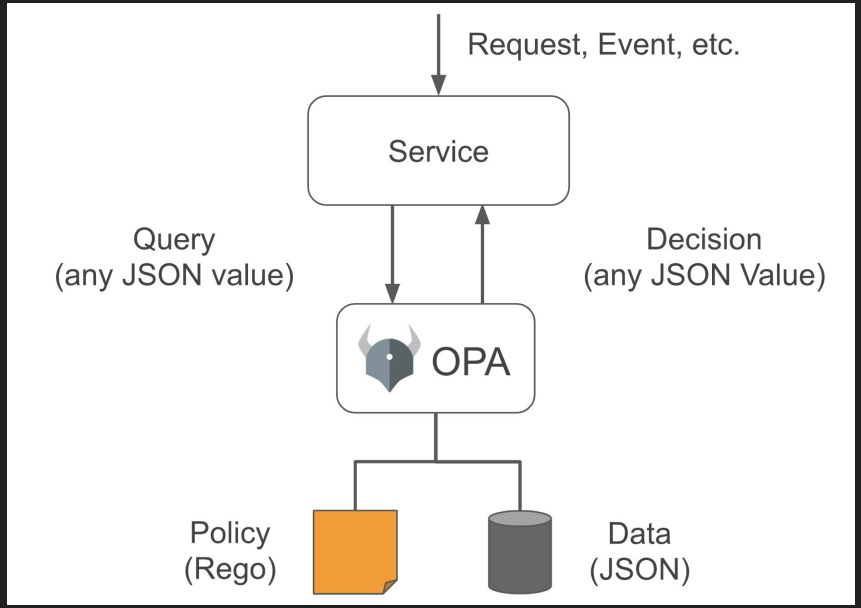
Patrick East on OPA slack

@peast907

# Open Policy Agent

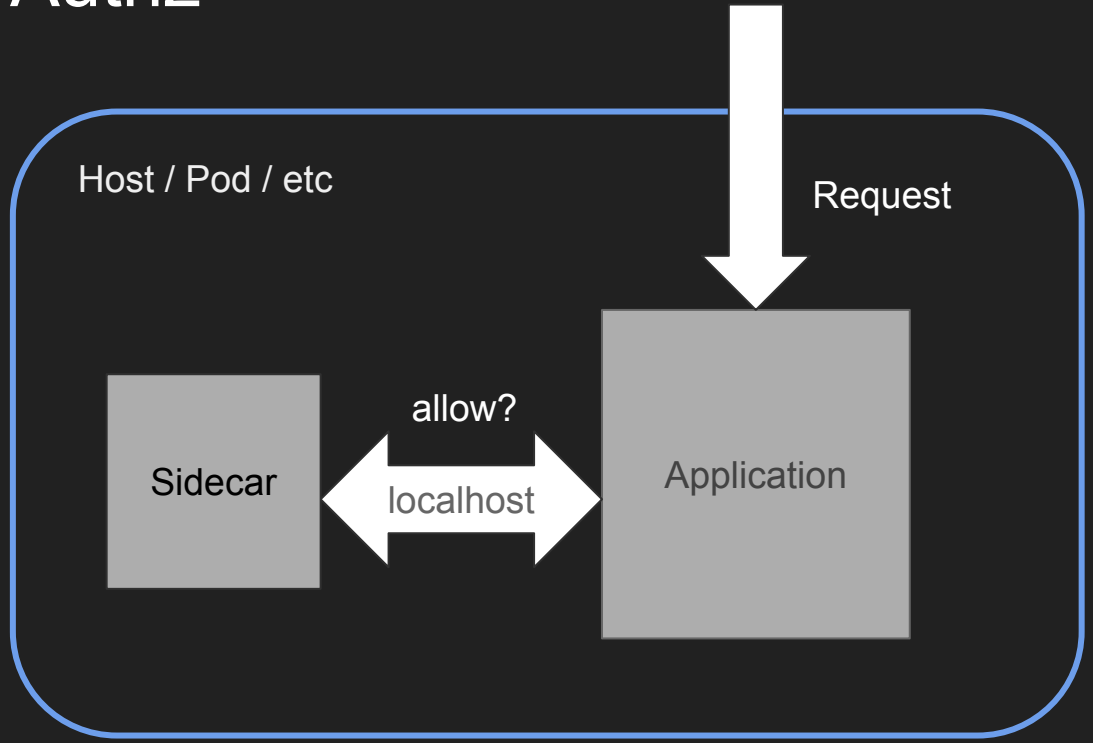
General purpose policy engine

Golang Library and/or HTTP Server

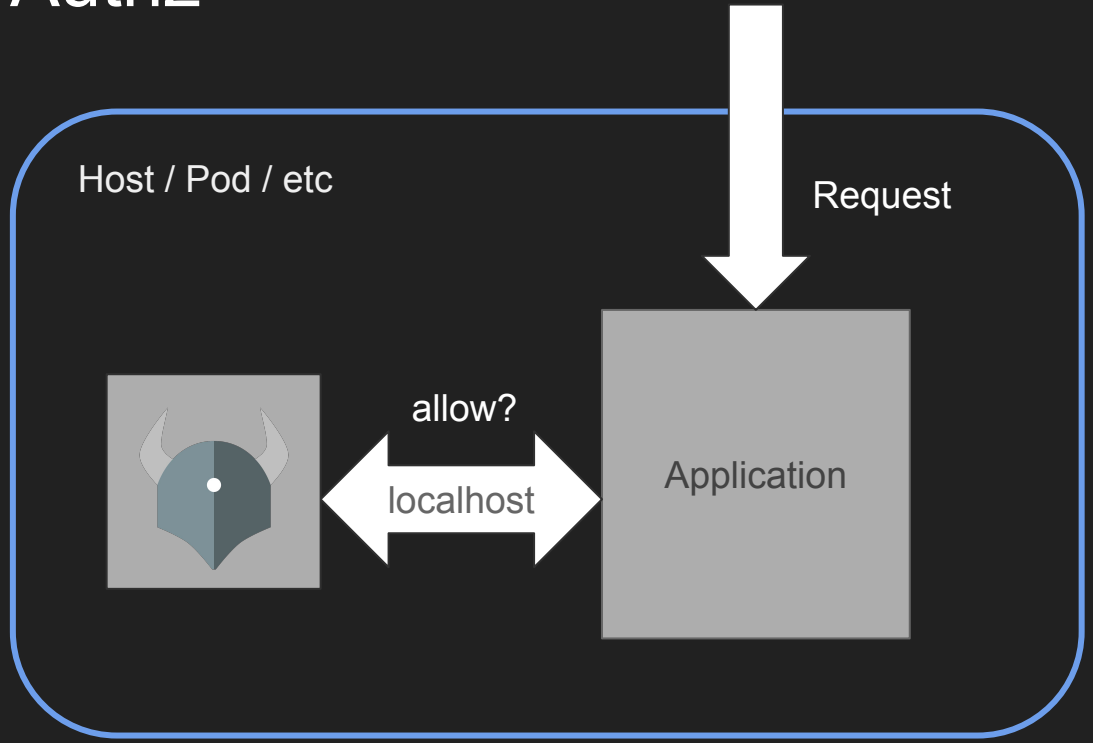



**https://www.openpolicyagent.org/**

# Sidecar Authz



# Sidecar Authz



# OPA is Perfect

# OPA is Perfect

- usually…

# OPA is Perfect

- usually…
- except when it isn’t

# No Builtin For That??

Use Case: API Authz with a JWT input

Problem: I need my JWKS from a remote location, with juuuuussstt the right error handling and caching.

# No Builtin For That??

Use Case: API Authz with a JWT input

Problem: I need my JWKS from a remote location, with juuuuussstt the right error handling and caching.

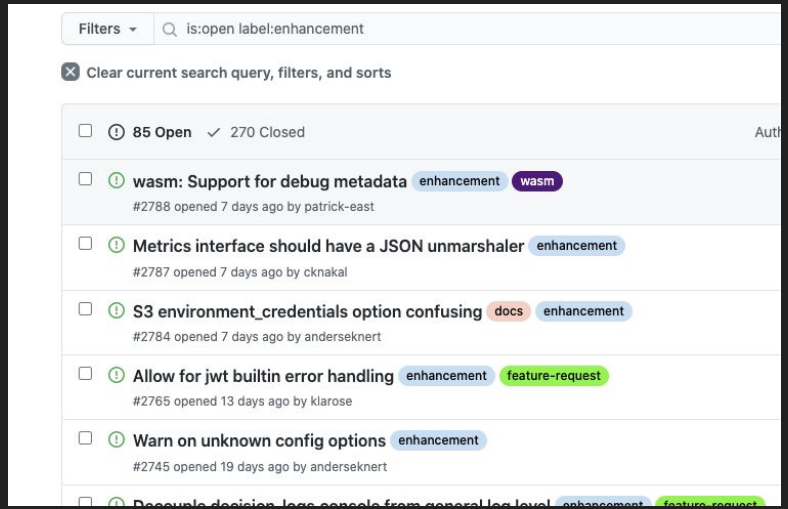
Solution: Add it to OPA!

# No Builtin For That??

Use Case: API Authz with a JWT input

Problem: I need my JWKS from a remote location, with juuuuussstt the right error handling and caching.

Solution: Add it to OPA!

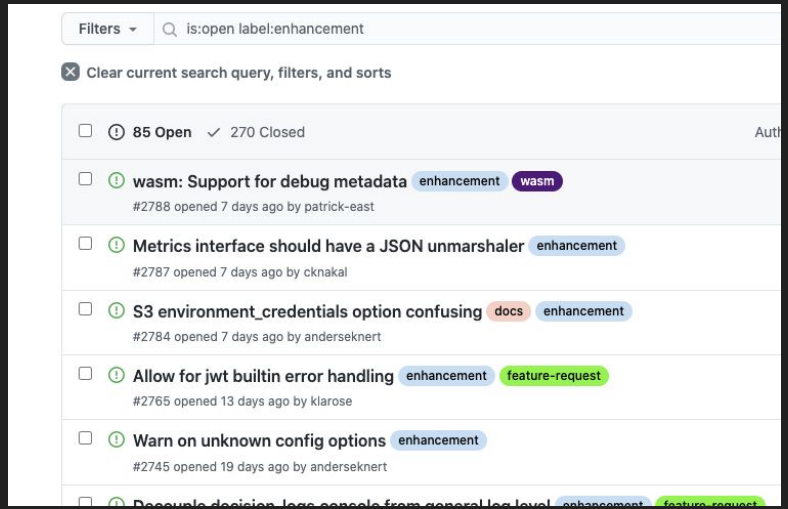


# No Builtin For That??

Use Case: API Authz with a JWT input

Problem: I need my JWKS from a remote location, with juuuuussstt the right error handling and caching.

Solution: Add it to OPA!




I'LL BUILD MY OWN OPA
WITH FETCH_JWKS AND ERROR HANDLING
imgflip.com

# Custom Built-in To Fetch JWKS

Expected behavior:

- Look for a key URI in the **`<url>/.well-known/openid-configuration`**
- Fall back to trying **`<url>/.well-known/jwks.json`**

# Custom Built-in To Fetch JWKS

Expected behavior:

Look for a key URI in the **`<url>/.well-known/openid-configuration`**

Fall back to trying **`<url>/.well-known/jwks.json`**

- What about **`http.send()`**?
  - Error handling?
  - Custom caching?
  - Complexity

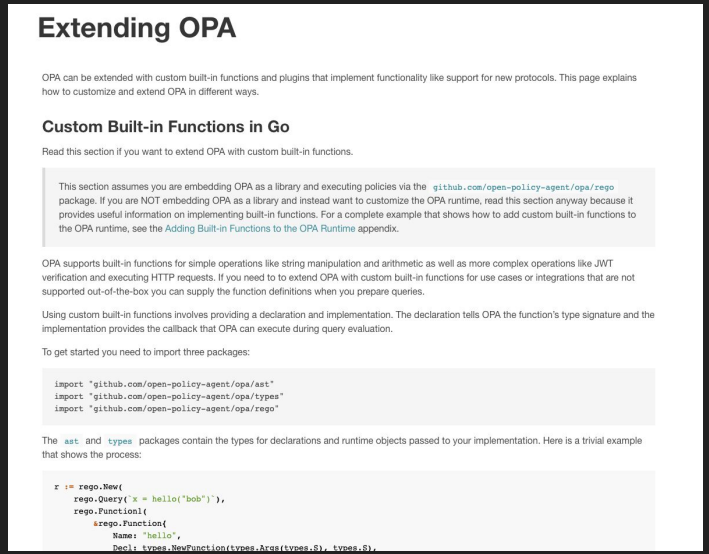
# Extending OPA - RTFM Edition

https://www.openpolicyagent.org/docs/latest/extensions/

# Extending OPA - RTFM Edition

https://www.openpolicyagent.org/docs/latest/extensions/

Looks hard though…



## Extending OPA

OPA can be extended with custom built-in functions and plugins that implement functionality like support for new protocols. This page explains how to customize and extend OPA in different ways.

### Custom Built-in Functions in Go

Read this section if you want to extend OPA with custom built-in functions.

> This section assumes you are embedding OPA as a library and executing policies via the `github.com/open-policy-agent/opa/rego` package. If you are NOT embedding OPA as a library and instead want to customize the OPA runtime, read this section anyway because it provides useful information on implementing built-in functions. For a complete example that shows how to add custom built-in functions to the OPA runtime, see the Adding Built-in Functions to the OPA Runtime appendix.

OPA supports built-in functions for simple operations like string manipulation and arithmetic as well as more complex operations like JWT verification and executing HTTP requests. If you need to to extend OPA with custom built-in functions for use cases or integrations that are not supported out-of-the-box you can supply the function definitions when you prepare queries.

Using custom built-in functions involves providing a declaration and implementation. The declaration tells OPA the function's type signature and the implementation provides the callback that OPA can execute during query evaluation.

To get started you need to import three packages:

```
import "github.com/open-policy-agent/opa/ast"
import "github.com/open-policy-agent/opa/types"
import "github.com/open-policy-agent/opa/rego"
```

The `ast` and `types` packages contain the types for declarations and runtime objects passed to your implementation. Here is a trivial example that shows the process:

```
r := rego.New(
    rego.Query(`x = hello("bob")`),
    rego.Function1(
        &rego.Function{
            Name: "hello",
            Decl: types.NewFunction(types.Args(types.S), types.S),
```

# Extending OPA - The Fear

# Extending OPA - The Fear

Custom image? :*(

# Extending OPA - The Fear

Custom image? :*(

But.. golang? Build tools? Gross..

# Extending OPA - The Fear

Custom image? :*(

But.. golang? Build tools? Gross..

More code to manage? :**(

# Extending OPA - The Reality

It’s pretty easy...

- Minimal code beyond custom plugin
- In practice most people use custom images anyway.
- Super easy to build with multi-stage docker image

# From Humble Beginnings...

```
$ git init && go mod init github.com/my-org/my-repo
```

**main.go --->**

```go
package main

import (
    "fmt"
    "os"

    "github.com/open-policy-agent/opa/cmd"
)

func main() {
    if err := cmd.RootCommand.Execute(); err != nil {
        fmt.Println(err)
        os.Exit( code: 1)
    }
}
```

```
$ go run main.go
```

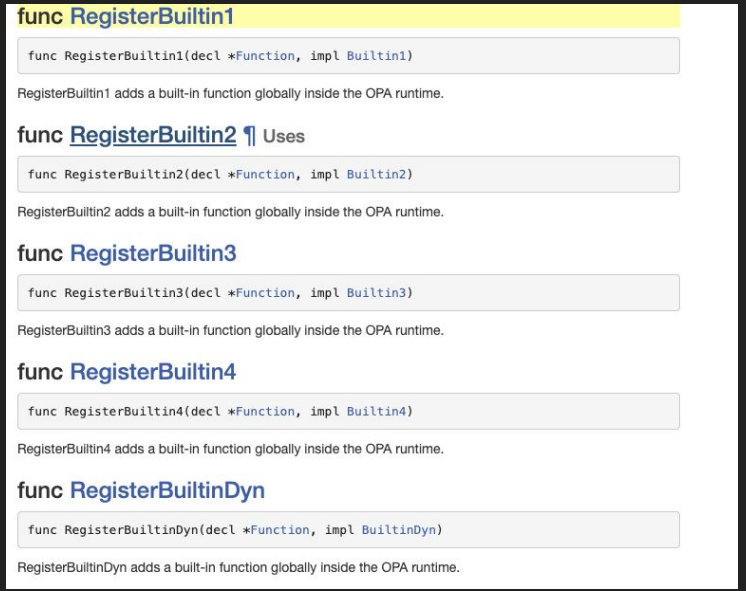
# Code Time


https://bit.ly/2TcpX1o

https://github.com/patrick-east/kubecon-na-2020/tree/master/custom-opa

# Custom Built-in Registration

- Compile-time → Signature
- Runtime → Implementation
- Register before “starting” OPA



func RegisterBuiltin1

```
func RegisterBuiltin1(decl *Function, impl Builtin1)
```

RegisterBuiltin1 adds a built-in function globally inside the OPA runtime.

func RegisterBuiltin2 ¶ Uses

```
func RegisterBuiltin2(decl *Function, impl Builtin2)
```

RegisterBuiltin2 adds a built-in function globally inside the OPA runtime.

func RegisterBuiltin3

```
func RegisterBuiltin3(decl *Function, impl Builtin3)
```

RegisterBuiltin3 adds a built-in function globally inside the OPA runtime.

func RegisterBuiltin4

```
func RegisterBuiltin4(decl *Function, impl Builtin4)
```

RegisterBuiltin4 adds a built-in function globally inside the OPA runtime.

func RegisterBuiltinDyn

```
func RegisterBuiltinDyn(decl *Function, impl BuiltinDyn)
```

RegisterBuiltinDyn adds a built-in function globally inside the OPA runtime.

# Code Time

https://bit.ly/2TcpX1o

https://github.com/patrick-east/kubecon-na-2020/tree/master/custom-opa

# New Requirement: Custom Decision Logger

- Don’t want to implement a decision log server??

# New Requirement: Custom Decision Logger

- Don’t want to implement a decision log server??
- Already got some cool infra?

# New Requirement: Custom Decision Logger

- Don’t want to implement a decision log server??
- Already got some cool infra? Use it!

# OPA Plugins 101

# OPA Plugins 101

- Define a struct and plugin name

# OPA Plugins 101

- Define a struct and plugin name

- Implement the Plugin interface →

```
type Plugin interface {
    Start(ctx context.Context) error
    Stop(ctx context.Context)
    Reconfigure(ctx context.Context, config interface{})
}
```

# OPA Plugins 101

- Define a struct and plugin name
- Implement the Plugin interface →

```
type Plugin interface {
    Start(ctx context.Context) error
    Stop(ctx context.Context)
    Reconfigure(ctx context.Context, config interface{})
}
```

- Define Factory →

```
type Factory interface {
    Validate(manager *Manager, config []byte) (interface{}, error)
    New(manager *Manager, config interface{}) Plugin
}
```

# OPA Plugins 101

- Define a struct and plugin name
- Implement the Plugin interface →

```
type Plugin interface {
    Start(ctx context.Context) error
    Stop(ctx context.Context)
    Reconfigure(ctx context.Context, config interface{})
}
```

- Define Factory →

```
type Factory interface {
    Validate(manager *Manager, config []byte) (interface{}, error)
    New(manager *Manager, config interface{}) Plugin
}
```

- Register →

```
func RegisterPlugin(name string, factory plugins.Factory)
```

# Code Time

https://bit.ly/2TcpX1o

https://github.com/patrick-east/kubecon-na-2020/tree/master/custom-opa

# Custom Decision Logger Plugin

## type Logger

```
type Logger interface {
    plugins.Plugin

    Log(context.Context, EventV1) error
}
```

Logger defines the interface for decision logging plugins.

# Custom Decision Logger Plugin

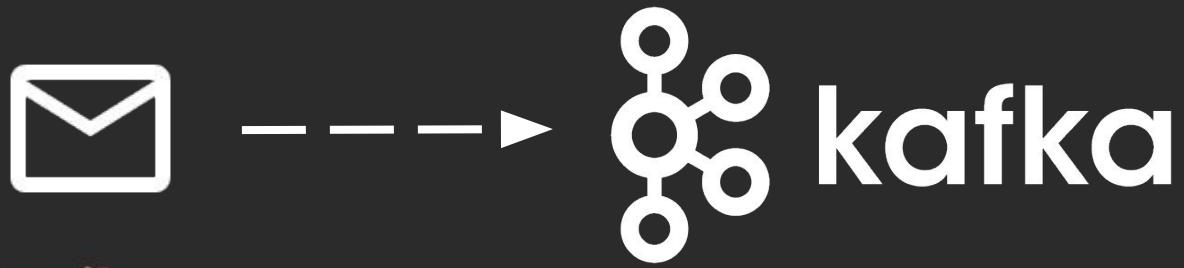
```
func (l *KafkaLogger) Log(ctx context.Context, event logs.EventV1) error {




    return nil
}
```

# Custom Decision Logger Plugin

```
func (l *KafkaLogger) Log(ctx context.Context, event logs.EventV1) error {
```




```
    return nil
}
```

# Code Time

https://bit.ly/2TcpX1o

https://github.com/patrick-east/kubecon-na-2020/tree/master/custom-opa

# New Requirement: Custom API Frontend

- All this HTTP and JSON overhead? Cmon…

# New Requirement: Custom API Frontend

- All this HTTP and JSON overhead? Cmon…
- Why not gRPC?????

# New Requirement: Custom API Frontend

- All this HTTP and JSON overhead? Cmon…
- Why not gRPC?????

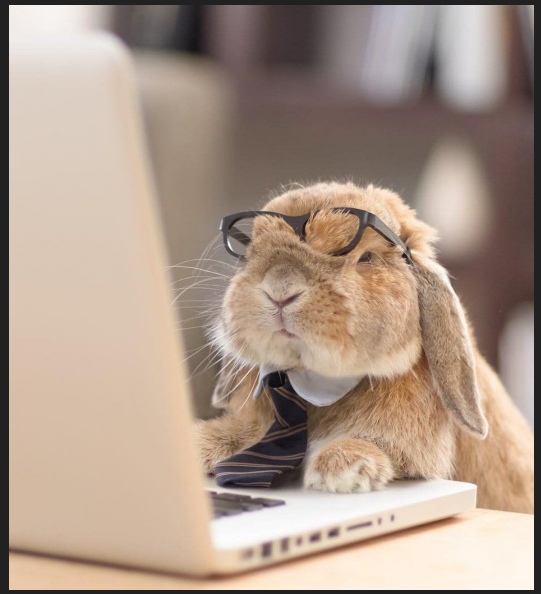
# Custom API Frontend - Solution

# Custom API Frontend - Solution

# Code Time

https://bit.ly/2TcpX1o

https://github.com/patrick-east/kubecon-na-2020/tree/master/custom-opa



# Custom API Frontend - Problem

```
type Manager struct {
    Store  storage.Store
    Config *config.Config
    Info   *ast.Term
    ID     string
    // contains filtered or unexported fields
}
```

Manager implements lifecycle management of plugins and gives plugins access to engine-wide components like storage.

Pro Tip: Manager.Store == The Policies

# Code Time

https://bit.ly/2TcpX1o

https://github.com/patrick-east/kubecon-na-2020/tree/master/custom-opa

# Thank You!