

Love, Hate, and Policy Languages

Gabriel L. Manor

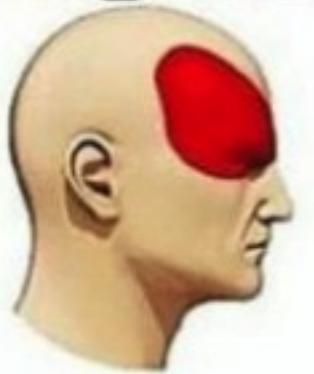
Engineering Director, DevRel @ Permit.io

The day is short, and the task is
great, and the workmen are
sluggish, and the reward is much,
and the Master of the house is
urgent.

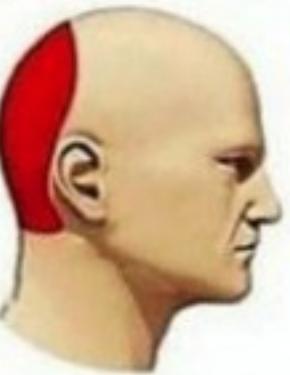
— Ethics of the Fathers, 2:15

Types of Headaches

Migraine



Hypertension



Stress



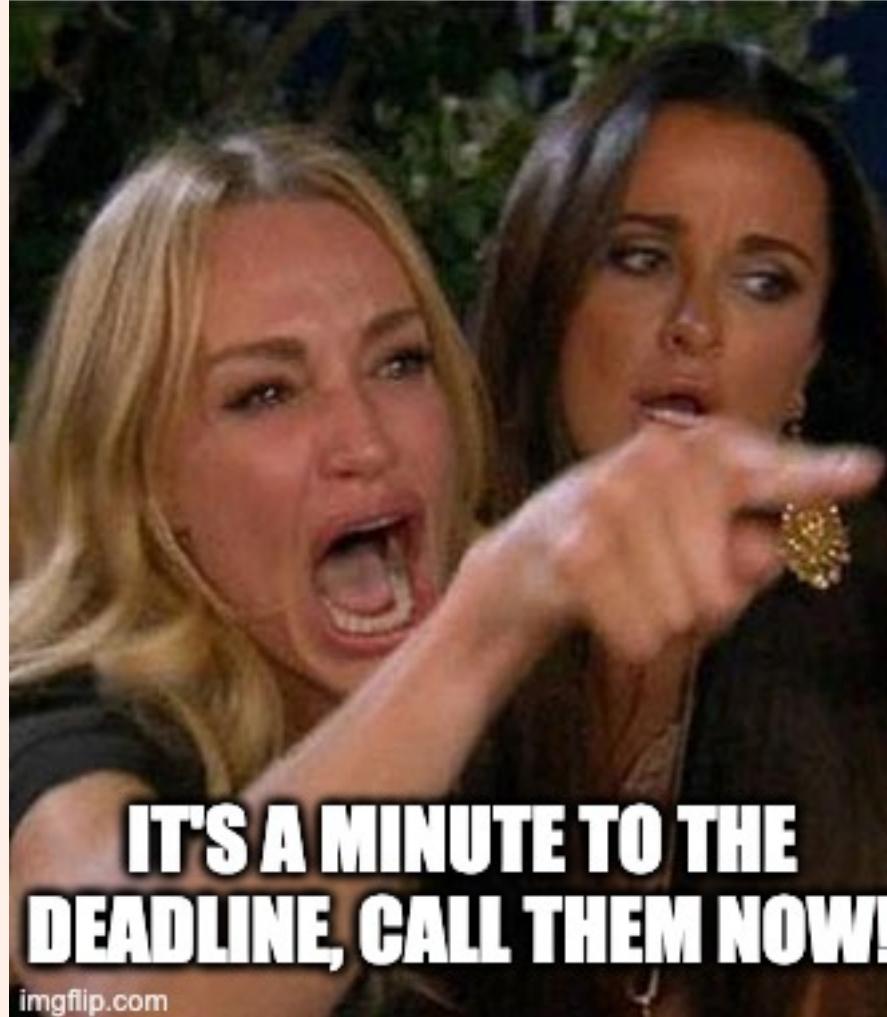
Start a
discussion with
someone I don't know



imgflip.com



My inner me

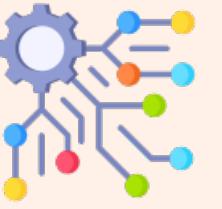


Also me





Chaos



Complexity



Velocity



Confidence

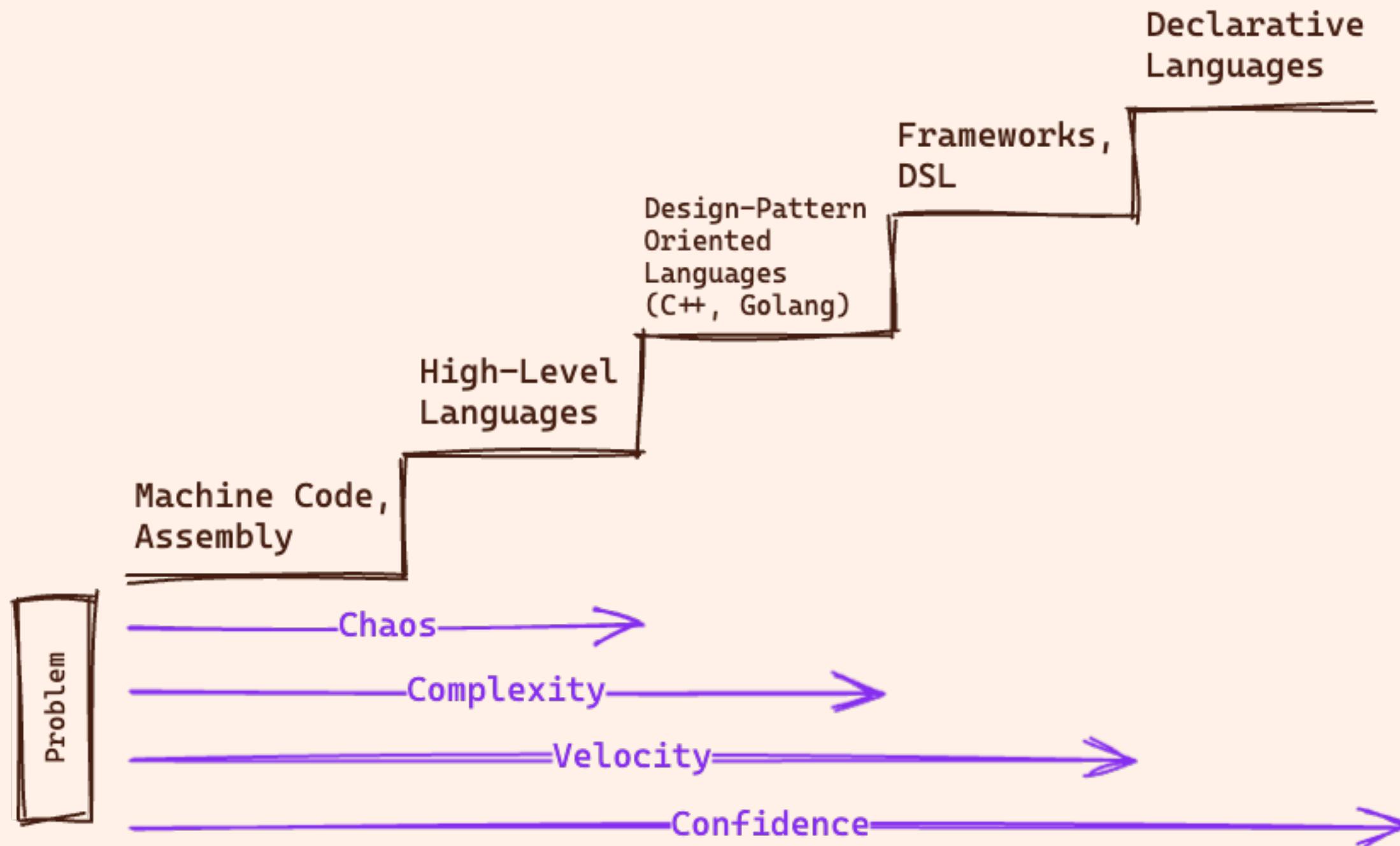


Gabriel L. Manor

Engineering Director @ Permit.io

🏆 Implementing Software Solution in His Personal Life

Not an ethical hacker, zero awards winner, dark mode hater.





HTML

```
<div>This is a div in HTML.</div>
```

C++

```
#include <windows.h>

// Function declarations
LRESULT CALLBACK WindowProcedure(HWND, UINT, WPARAM, LPARAM);
void DrawRectangle(HWND hwnd);

// Entry point
int WINAPI WinMain() {
    WNDCLASSW wc = {0};

    wc.hbrBackground = (HBRUSH)COLOR_WINDOW;
    wc.hCursor = LoadCursor(NULL, IDC_ARROW);
    wc.hInstance = hInst;
    wc.lpszClassName = L"myWindowClass";
    wc.lpfnWndProc = WindowProcedure;

    if (!RegisterClass(&wc)) return -1;

    CreateWindow(
        L"myWindowClass",
        L"Simple Div in C++",
        WS_OVERLAPPEDWINDOW | WS_VISIBLE,
        100,
        100,
        500,
        500,
        NULL,
        NULL,
        NULL,
        NULL,
        NULL
    );

    MSG msg = {0};
    while (GetMessage(&msg, NULL, 0, 0)) {
        TranslateMessage(&msg);
        DispatchMessage(&msg);
    }

    return 0;
}

// Window procedure
LRESULT CALLBACK WindowProcedure(HWND hwnd, UINT msg, WPARAM wp, LPARAM lp) {
    switch (msg) {
        case WM_DESTROY:
            PostQuitMessage(0);
            break;
        case WM_PAINT:
            DrawRectangle(hwnd);
            break;
        default:
            return DefWindowProc(hwnd, msg, wp, lp);
    }
    return 0;
}

// Function to draw a rectangle
void DrawRectangle(HWND hwnd) {
    PAINTSTRUCT ps;
    HDC hdc = BeginPaint(hwnd, &ps);
    // Set up the rectangle dimensions
    RECT rect = { 50, 50, 250, 150 };
    // Set up the brush color
    HBRUSH brush = CreateSolidBrush(RGB(0, 0, 255));
    FillRect(hdc, &rect, brush);
    EndPaint(hwnd, &ps);
}
```

Assembly

```
section .data
    className db 'WinClass',0
    windowName db 'Window',0

section .bss
    hInstance resb 6
    msg resb 28
    hwnd resb 4
    wndClass resb 48

section .text
    global _start

_start:
    ; Get instance handle
    call [GetModuleHandle]
    mov [hInstance], eax

    ; Define window class
    mov eax, [hInstance]
    mov [wndClass+36], eax
    mov dword [wndClass+40], WindowProc
    mov dword [wndClass+44], 0x0003
    mov [wndClass], 0x00000000
    lea eax, [className]
    mov [wndClass+5], eax
    call [RegisterClass]

    ; Create window
    push 0
    push 0
    push 0
    push 0
    push 0
    push 0
    push 0x360000
    push windowName
    push className
    push 0x00cf0000
    call [CreateWindowEx]
    mov [hwnd], eax

    ; Show and update window
    push 1
    push [hwnd]
    call [ShowWindow]
    push [hwnd]
    call [UpdateWindow]

    ; Message loop
.message_loop:
    push 0
    push 0
    push [hwnd]
    lea eax, [msg]
    push eax
    call [GetMessage]
    test eax, eax
    jz .exit
    lea eax, [msg]
    push eax
    call [TranslateMessage]
    lea eax, [msg]
    push eax
    call [DispatchMessage]
    jmp .message_loop

.exit:
    mov eax, 1
    xor ebx, ebx
    int 0x80

; Window procedure
WindowProc:
    push ebp
    mov ebp, esp
    cmp dword [ebp+12], 0x000F
    je .wm_paint
    cmp dword [ebp+12], 0x0002
    je .wm_destroy

    ; Default processing
    push dword [ebp+16]
    push dword [ebp+12]
    push dword [ebp+8]
    call [DefWindowProc]
    jmp .done

.wm_paint:
    ; Handle WM_PAINT message here to draw the rectangle
    ; This part involves setting up a device context, drawing commands, etc.

.done:
    jmp .done

.wm_destroy:
    push 0
    call [PostQuitMessage]
    xor eax, eax
    jmp .done

.done:
    mov esp, ebp
    pop ebp
    ret
```

Domain-Specific Declarative Languages



SQL



YAML



LilyPond



CEL



XML



LaTeX



Gherkin



HCL



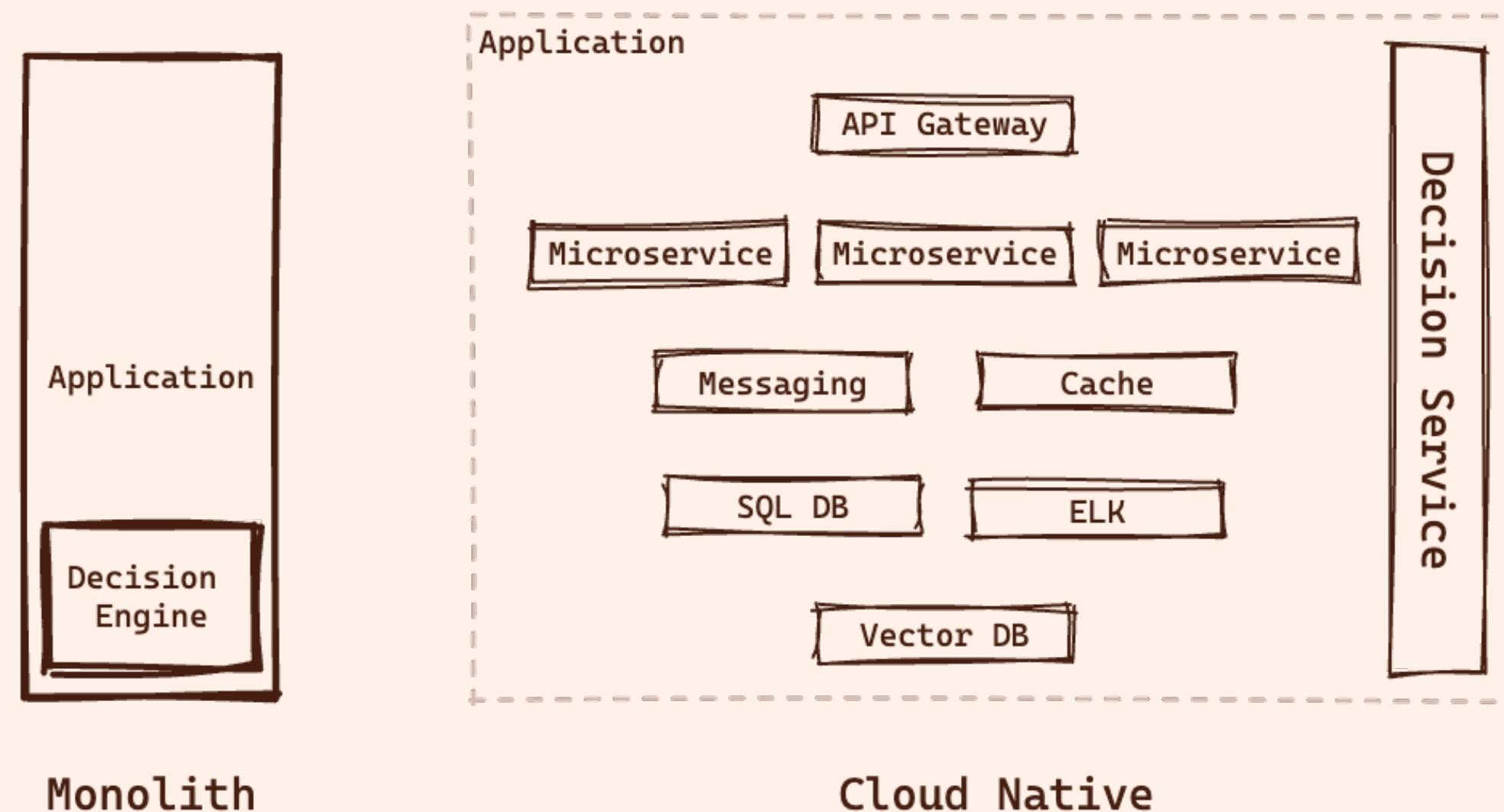
```
if (true === "true") {  
    return "false";  
}
```



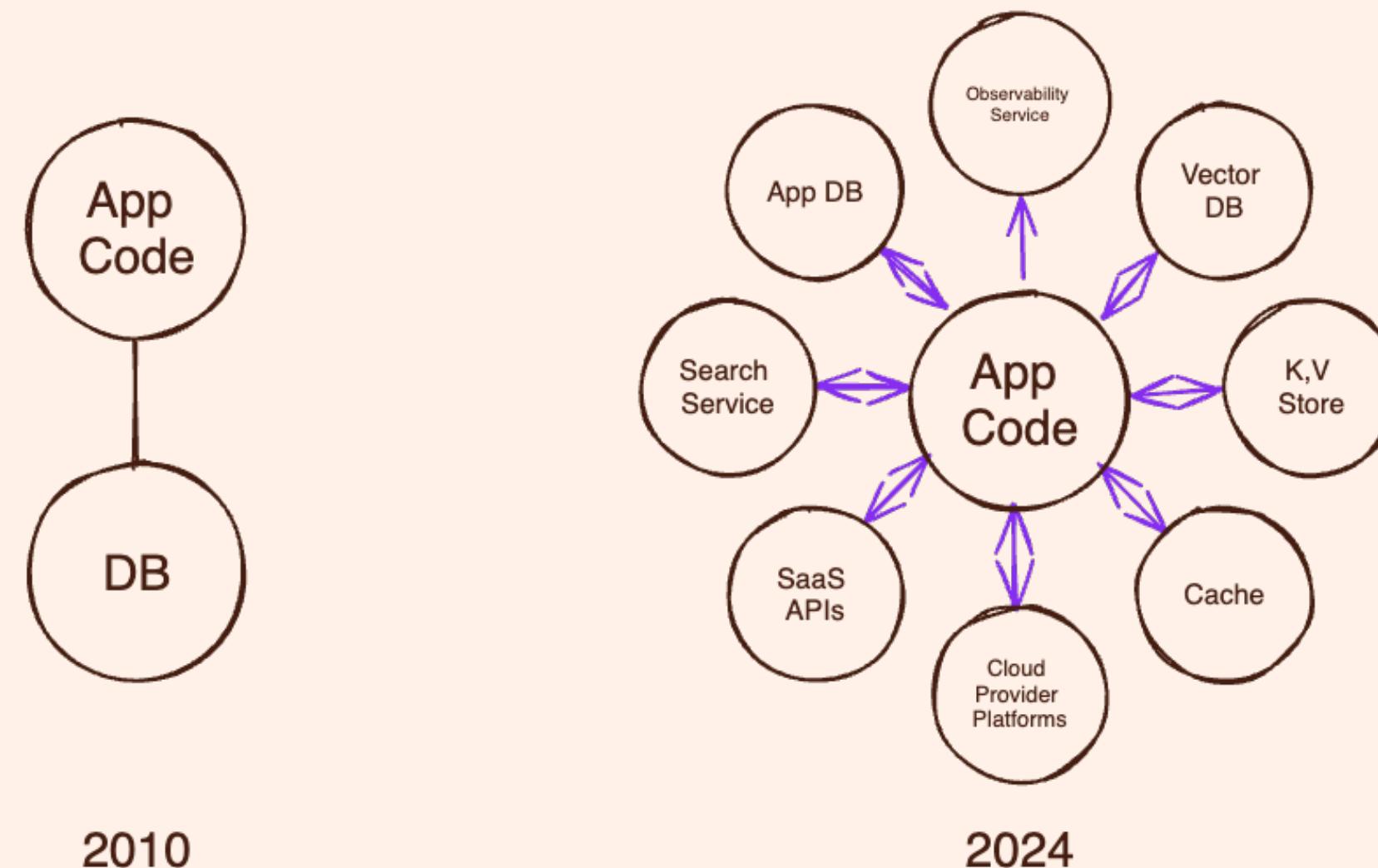
(Not) Fine-Grained Desicion

```
if (
  user.role == DOCTOR ||
  (user.specialization == "CARDIOLOGY" &&
   allowedRecordTypes.includes(patientRecord.type) &&
   (user.role == NURSE || user.role == MEDICAL_ASSISTANT))
) {
  // Hello future me. I'm so sorry.
  // I really hoped you never have to see this code again.
  // But here we are.
  // How's mom?
  return true;
}
```

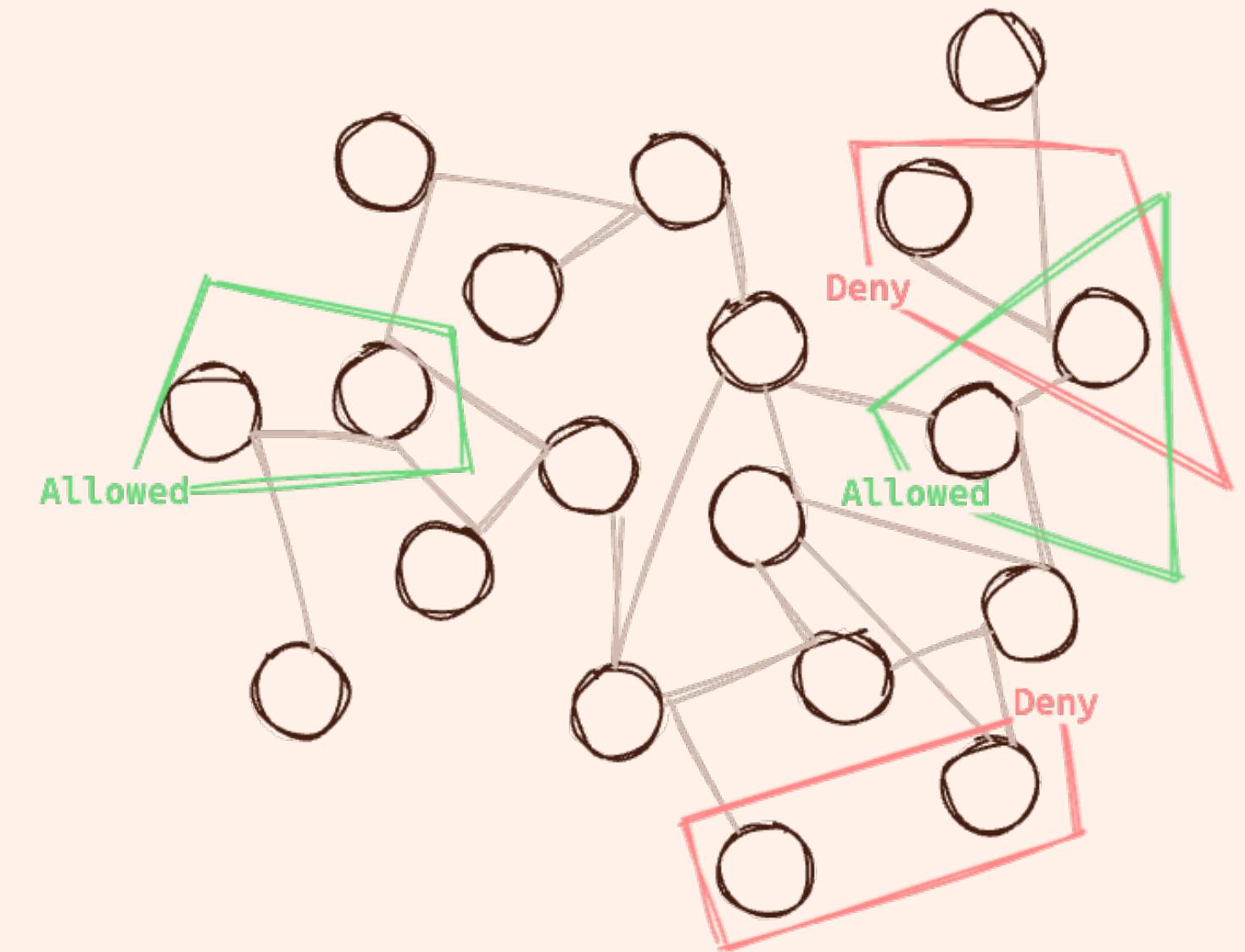
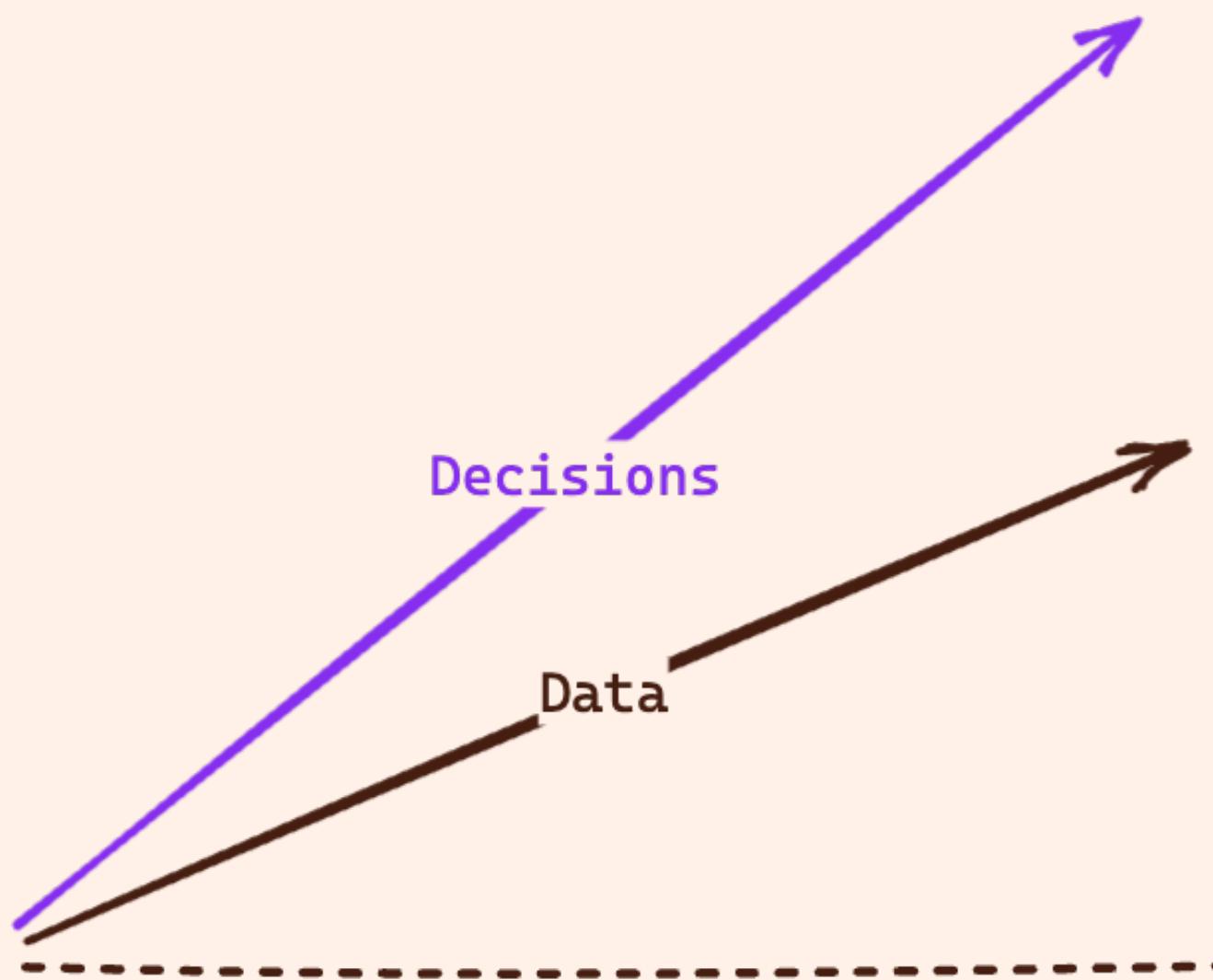
Chaos, Complexity, Velocity, Confidence



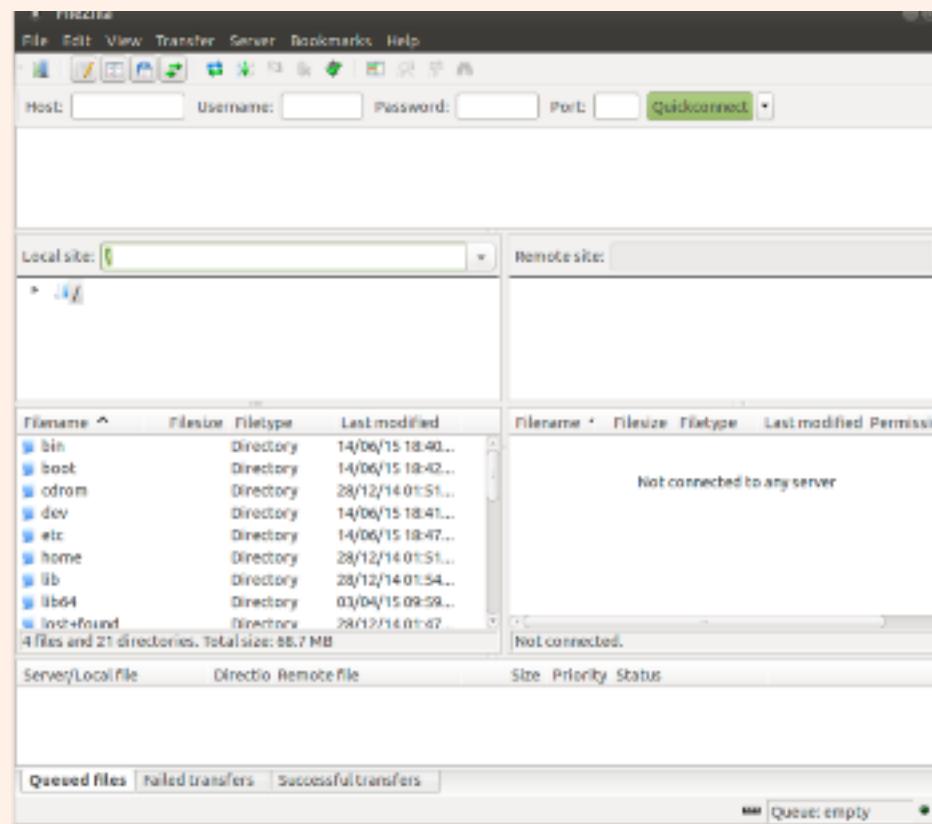
Chaos, *Complexity*, Velocity, Confidence



Decision Fatigue Syndrome



Chaos, Complexity, *Velocity*, Confidence



Monthly Night Work

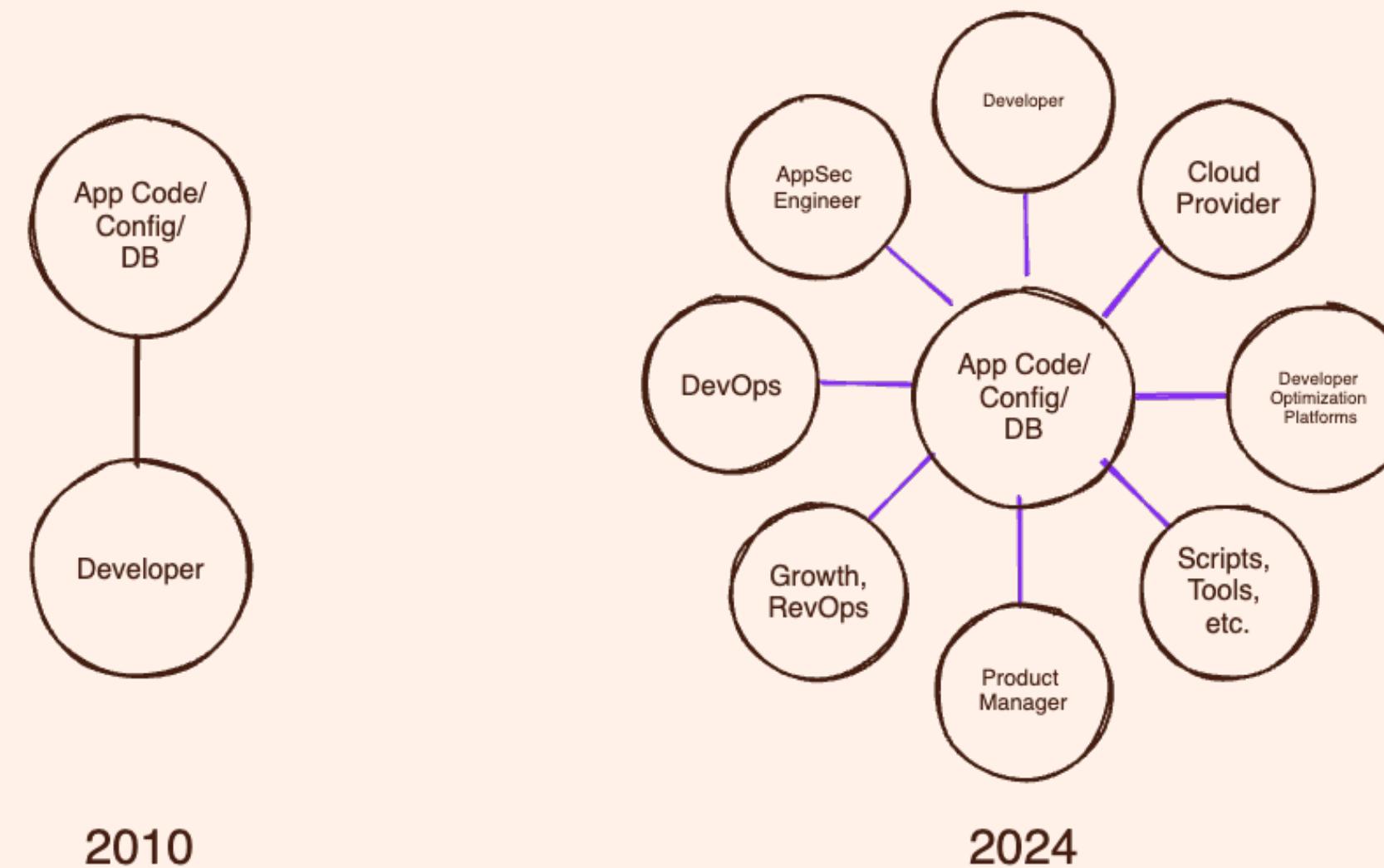
<input checked="" type="checkbox"/> Deploy to Production	Deploy to Production #282: Manually run by RazcoDev	 main	 2 days ago	 25s	...
<input checked="" type="checkbox"/> Deploy to Production	Deploy to Production #281: Manually run by obed	 main	 2 days ago	 25s	...
<input checked="" type="checkbox"/> Deploy to Production	Deploy to Production #280: Manually run by omer9664	 main	 3 days ago	 28s	...
<input checked="" type="checkbox"/> Deploy to Production	Deploy to Production #288: Manually run by omer9664	 main	 6 days ago	 25s	...
<input checked="" type="checkbox"/> Deploy to Production	Deploy to Production #288: Manually run by omer9664	 main	 last week	 25s	...
<input checked="" type="checkbox"/> Deploy to Production	Deploy to Production #282: Manually run by razekatz	 main	 last week	 28s	...
<input checked="" type="checkbox"/> Deploy to Production	Deploy to Production #286: Manually run by noamovich18	 main	 last week	 23s	...
<input checked="" type="checkbox"/> Deploy to Production	Deploy to Production #285: Manually run by omer9584	 main	 last week	 25s	...

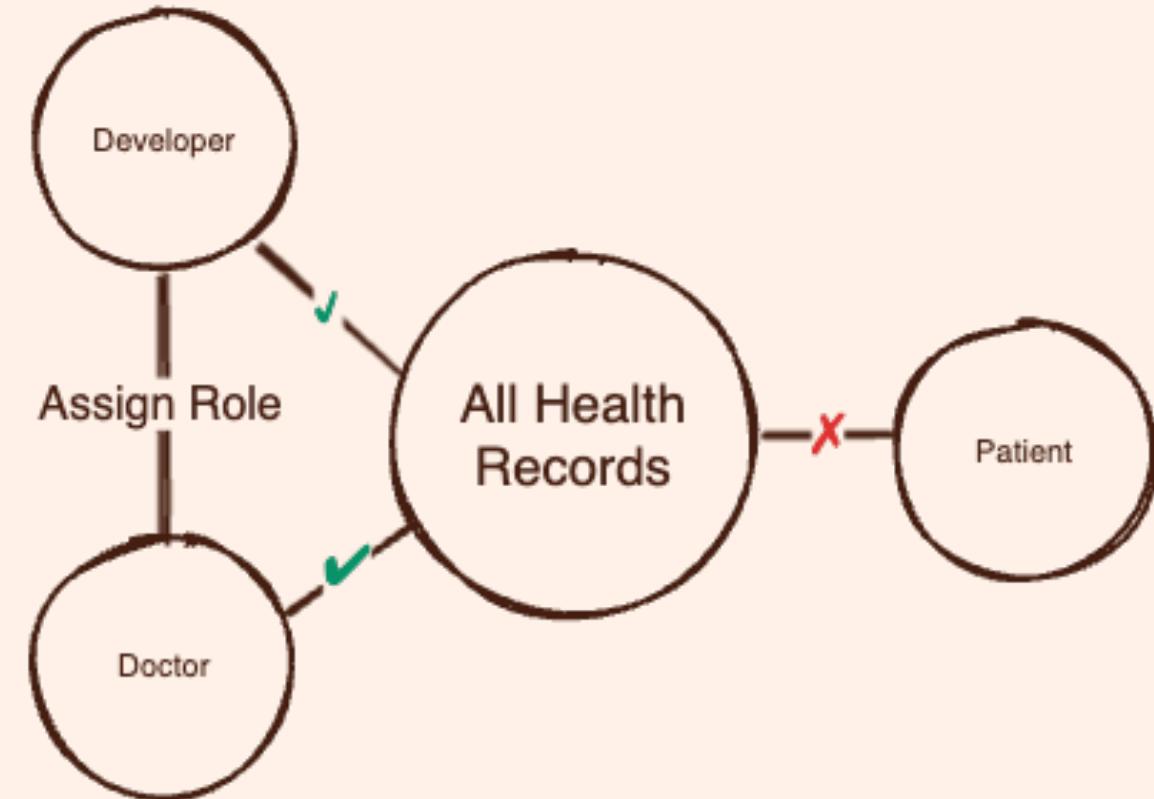
Multiple Daily Deployments

Chaos, Complexity, Velocity, Confidence

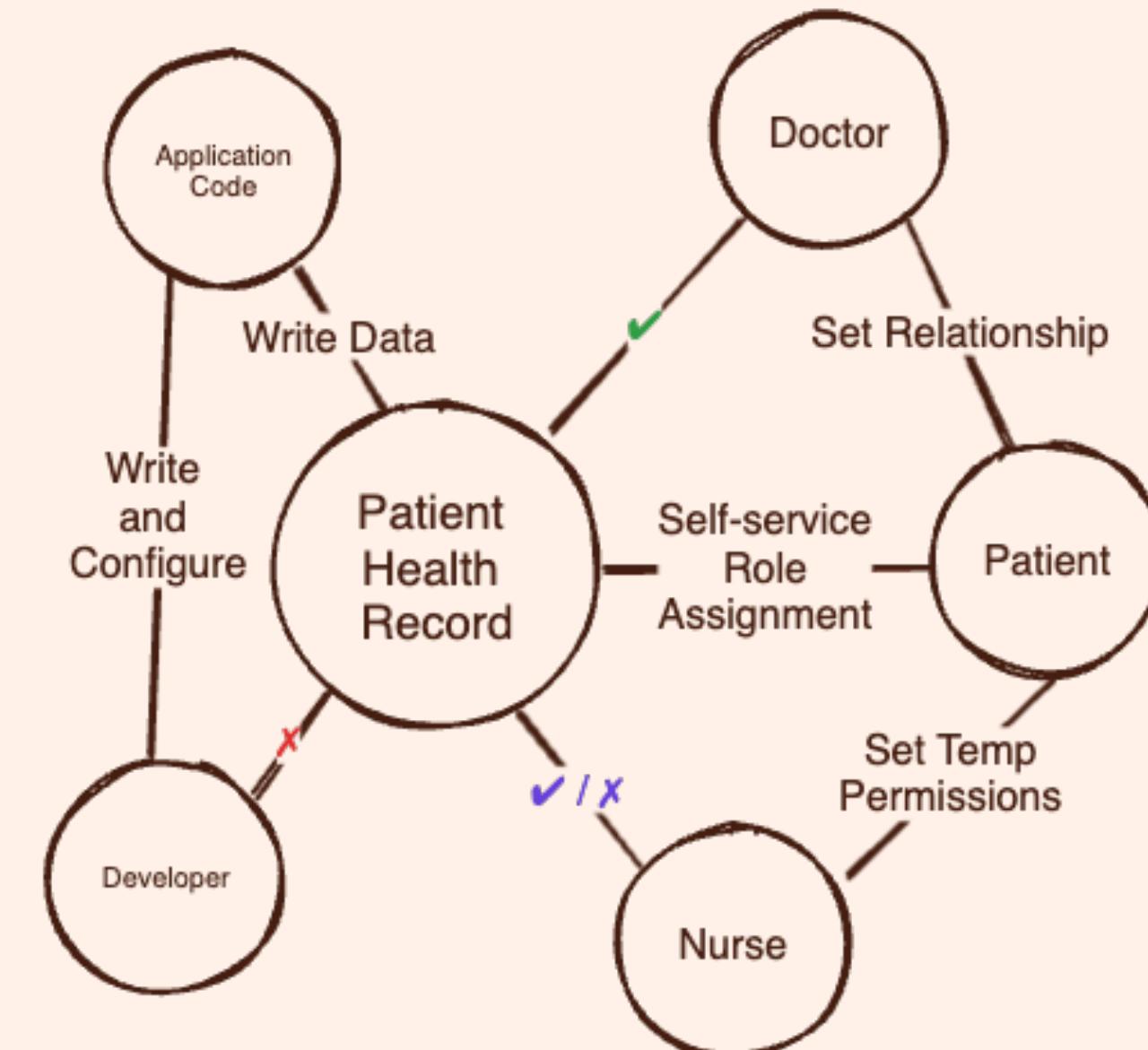


Application Code Stakeholders





2018

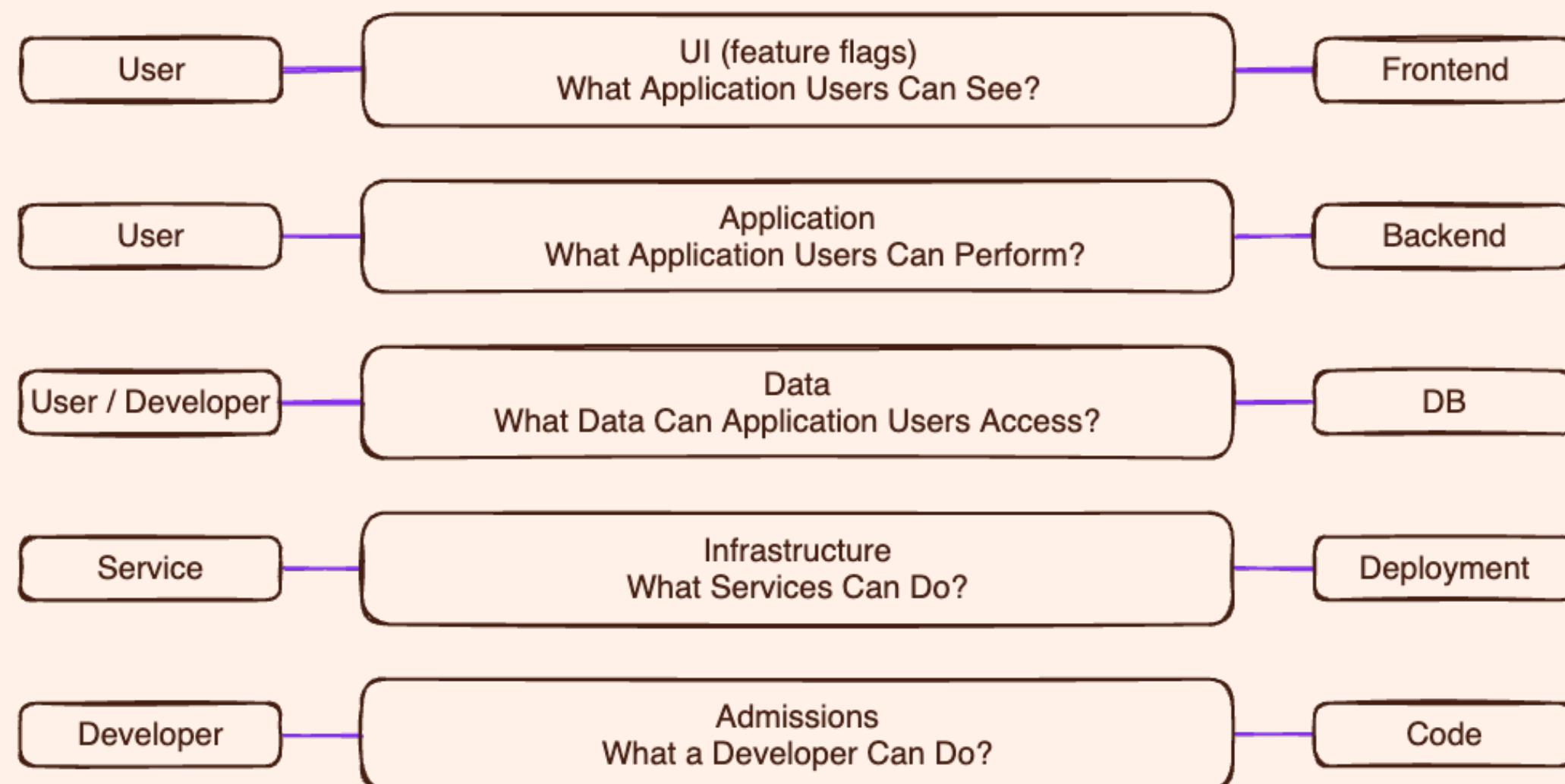


2024

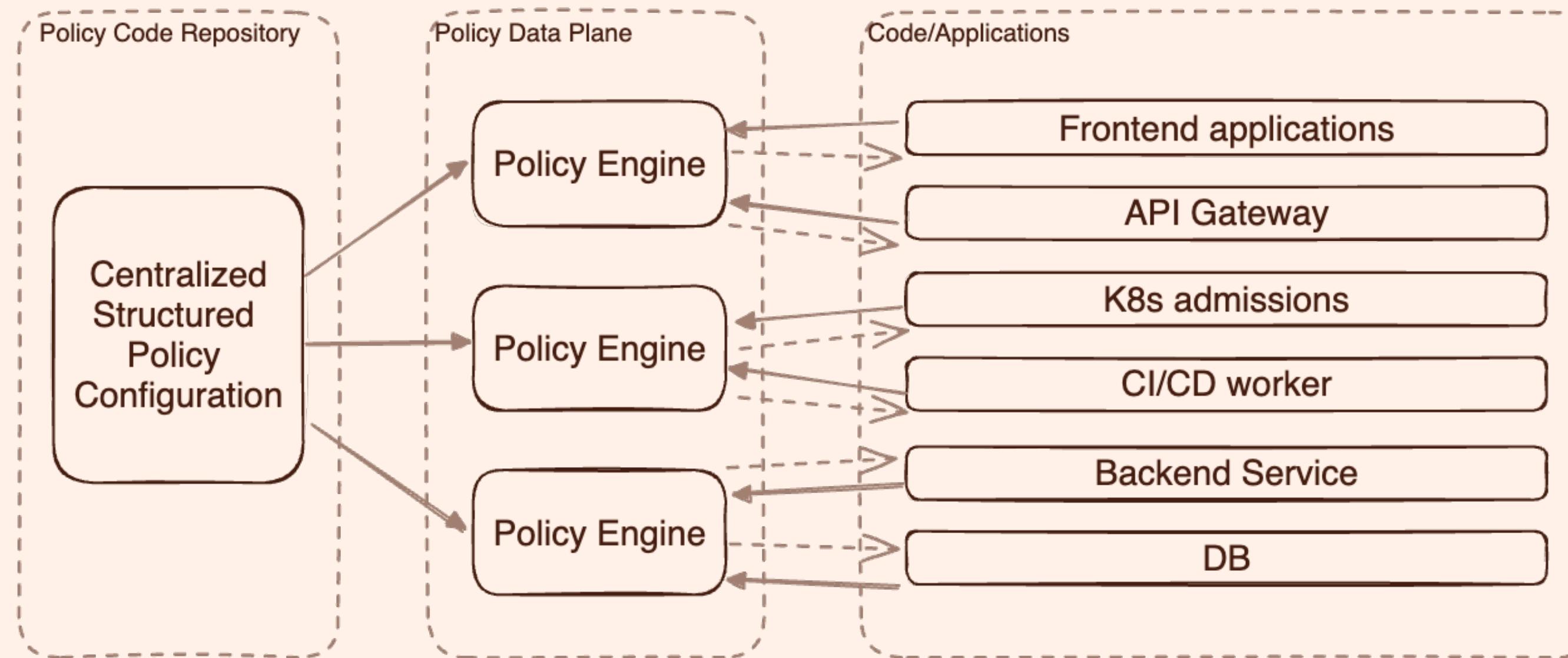


Chaos ➡ Structure

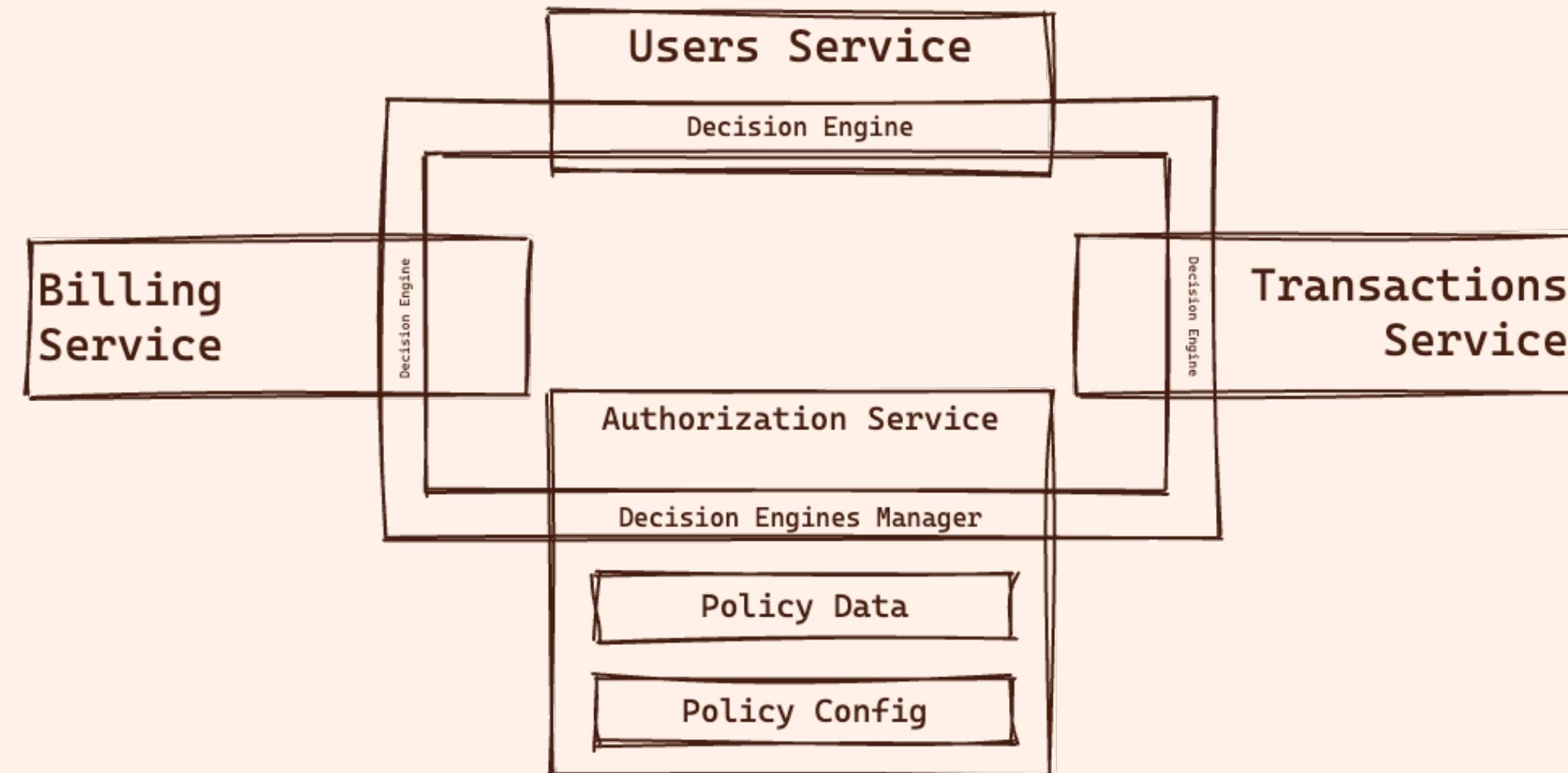
Authorization Layers



Abstract Authorization Service Model

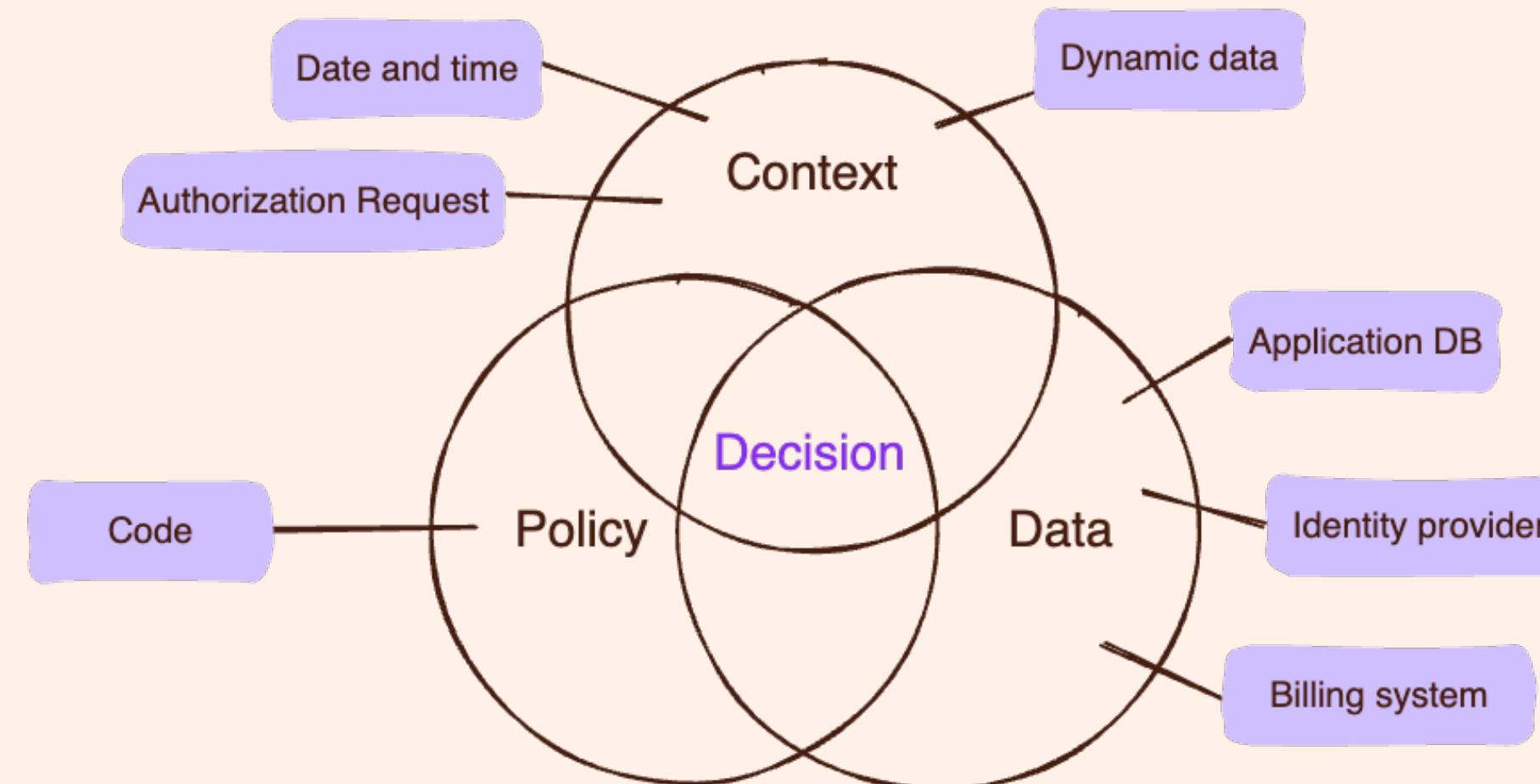


Application-Level Domain-Driven Model



Complexity ➡ Design Patterns

Abstract Decision Components



Authorization Decision Types

Allow / Deny

- Blue ceramic mug
- Hardcover novel
- USB flash drive
- Wooden cutting board
- Stainless steel watch
- Pair of running shoes
- Handheld flashlight
- Green yoga mat
- Set of colored pencils
- Glass water bottle

Binary Check Function

```
check({ user }, { action }, { resource }, { context });
```

Binary Check Function

```
check({ user }, { action }, { resource }, { context });
```

```
response = requests.post('http://host.docker.internal:8180/v1/is_authorized', json={  
    "principal": f"User::\"{user}\"",  
    "action": f"Action::\"{method.lower()}\"",  
    "resource": f"ResourceType::\"{original_url.split('/')[1]}\"",  
    "context": request.json  
})
```

```
const response = await fetch(  
  "http://host.docker.internal:8180/v1/is_authorized",  
  {  
    method: "POST",  
    body: JSON.stringify({  
      principal: `User::\"${user}\\"",  
      action: `Action::\"${method.toLowerCase()}\"`,  
      resource: `ResourceType::\"${originalUrl.split("/")[1]}\"`,  
      context: body,  
    }),  
  }  
);
```

Partial Evaluation Flow

```
permit (
    principal == PhotoApp::User::"stacey",
    action == PhotoApp::Action::"viewPhoto",
    resource
)
when { resource in PhotoApp::Account::"stacey" };
```



```
evaluate("pictures", "stacy", "viewPhoto");
```



```
SELECT *
FROM photos
WHERE owner = 'Stacey';
```

Velocity ➡ Frameworks

Permission Models



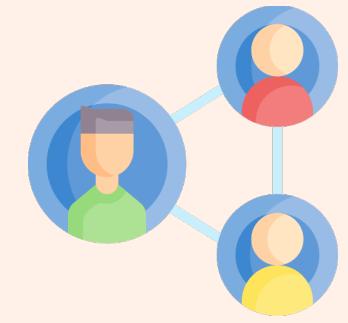
PBAC



RBAC



ABAC



ReBAC

Policy Based Access Control

✓ No External Data Policies

allow developers to deploy to production only on weekends

deny developers from creating S3 buckets with the name "prod"

allow developers to create S3 buckets only if their name starts with "dev-"

✗ External Data Policies

allow developers to deploy S3 buckets only if the next billing cycle is less than \$100

Role Based Access Control

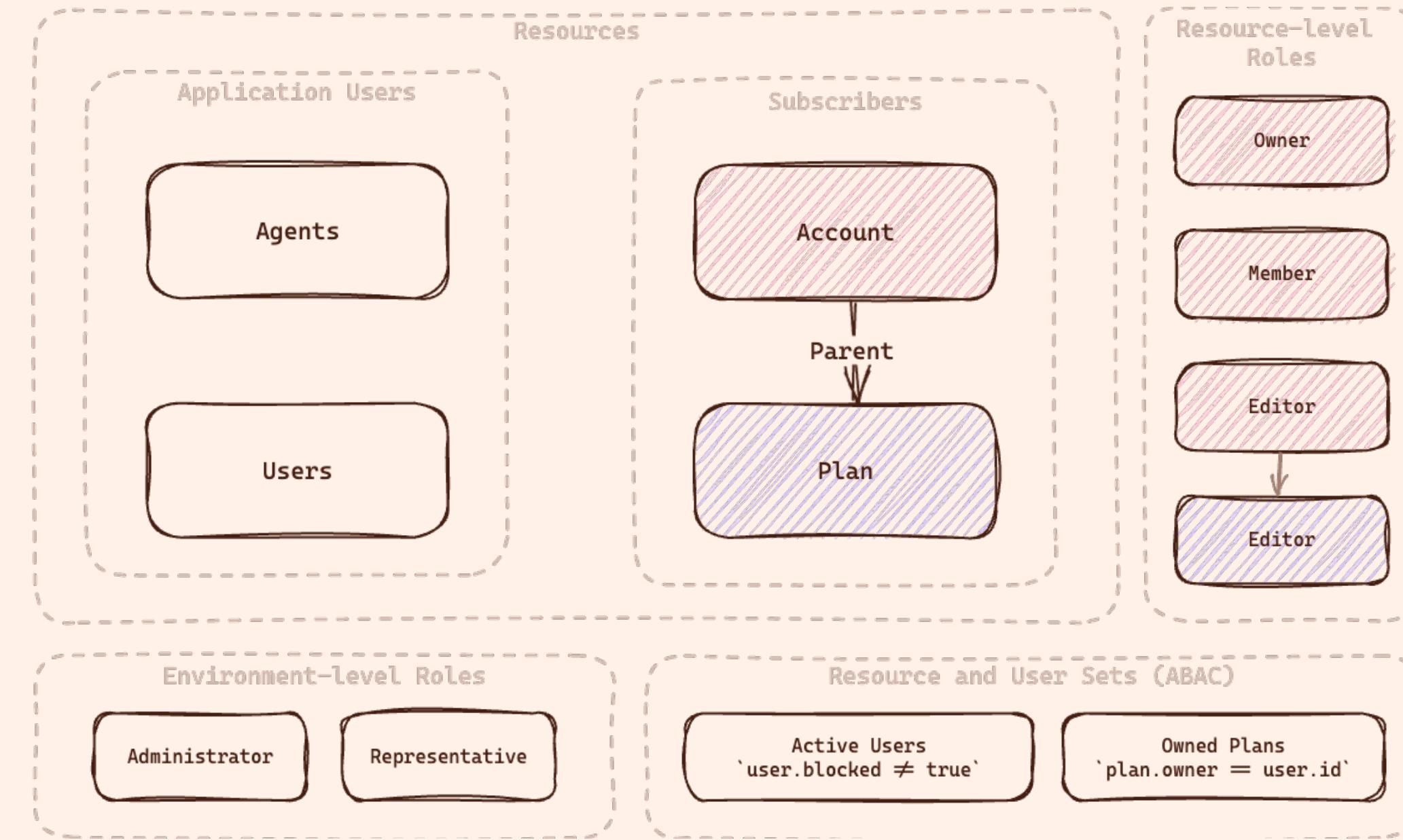
- Leverage user roles to determine permissions
- Commonly used as a term for authorization
- Very simple to implement and audit
- Flexible to use in different domains
- Hard to consider external data in policies
- Limited granularity in resource permissions

Relationship Based Access Control

- Assign instance-level roles to users
- Leverage entity relationships to derive role permissions
- Built to support modern data models in efficient way
- Require centralized graph to calculate decisions
- Limited granularity in user and resource inspection

Attribute Based Access Control

- Leverage entities' attributes to model permissions
- Use attribute conditions in the policies
- Most flexible structured policy model
- Complex to build and audit in imperative languages
- Usually used as extension to RBAC and ReBAC systems



RBAC, ReBAC, ABAC Modeling Example



io.permit.io/pink-mobile

Pink-Mobile

Manager Representative User

Luna Lovegood

Ron Weasley – ron@weasley.me Unassign

Add User Email Add

Sirius Black

Harry Potter – harry@potter.io Unassign

Add User Email Add

Pink-Mobile

Manager Representative User

Sirius Black ▼

Harry Potter ▼

User Plan: VIP

Change User's Plan

Pink-Mobile

Manager Representative User

Harry Potter ▼

Name: Harry Potter

Email: harry@potter.io

Plan: VIP

Change Plan

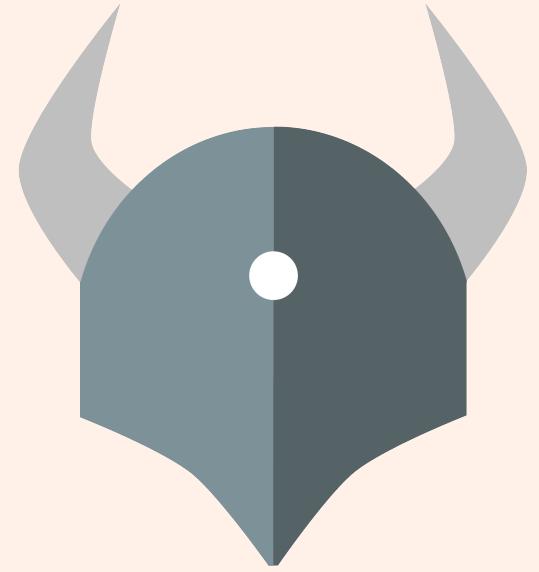
Confidence ➡ Domain-Specific Declarative Code

```
// Users can edit their own info, admins can edit anyone's info
permit (
    principal,
    action,
    resource in HealthCareApp::InfoType::"accountinfo"
)
when {
    resource.subject == principal ||
    principal in HealthCareApp::Role::"admin"
};

//A patient may create an appointment for themselves, or an administrator can do it
permit (
    principal,
    action == HealthCareApp::Action::"createAppointment",
    resource
)
when {
    (context.referrer in HealthCareApp::Role::"doctor" && resource.patient == principal) ||
    principal in HealthCareApp::Role::"admin"
};
```

Domain-Specific Declarative Code Advantages

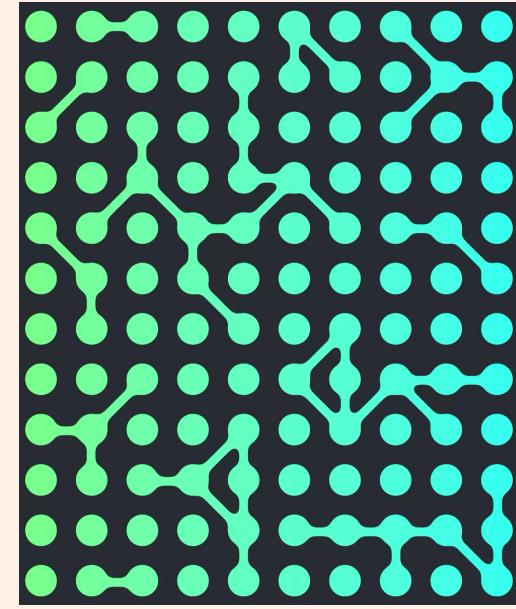
- Performance
- Reusability
- Versioning
- Testing
- Maintainance
- Readability
- Validation



Open Policy
Agent



AWS
Cedar



OpenFGA



```
allow {
    input.user.role == "viewer"
    validate_department(input.user, input.document)
    validate_classification(input.user.role, input.document.classification)
    validate_dynamic_rules(input.user, input.document)
}

validate_department(user, document) {
    user.department == document.department
}

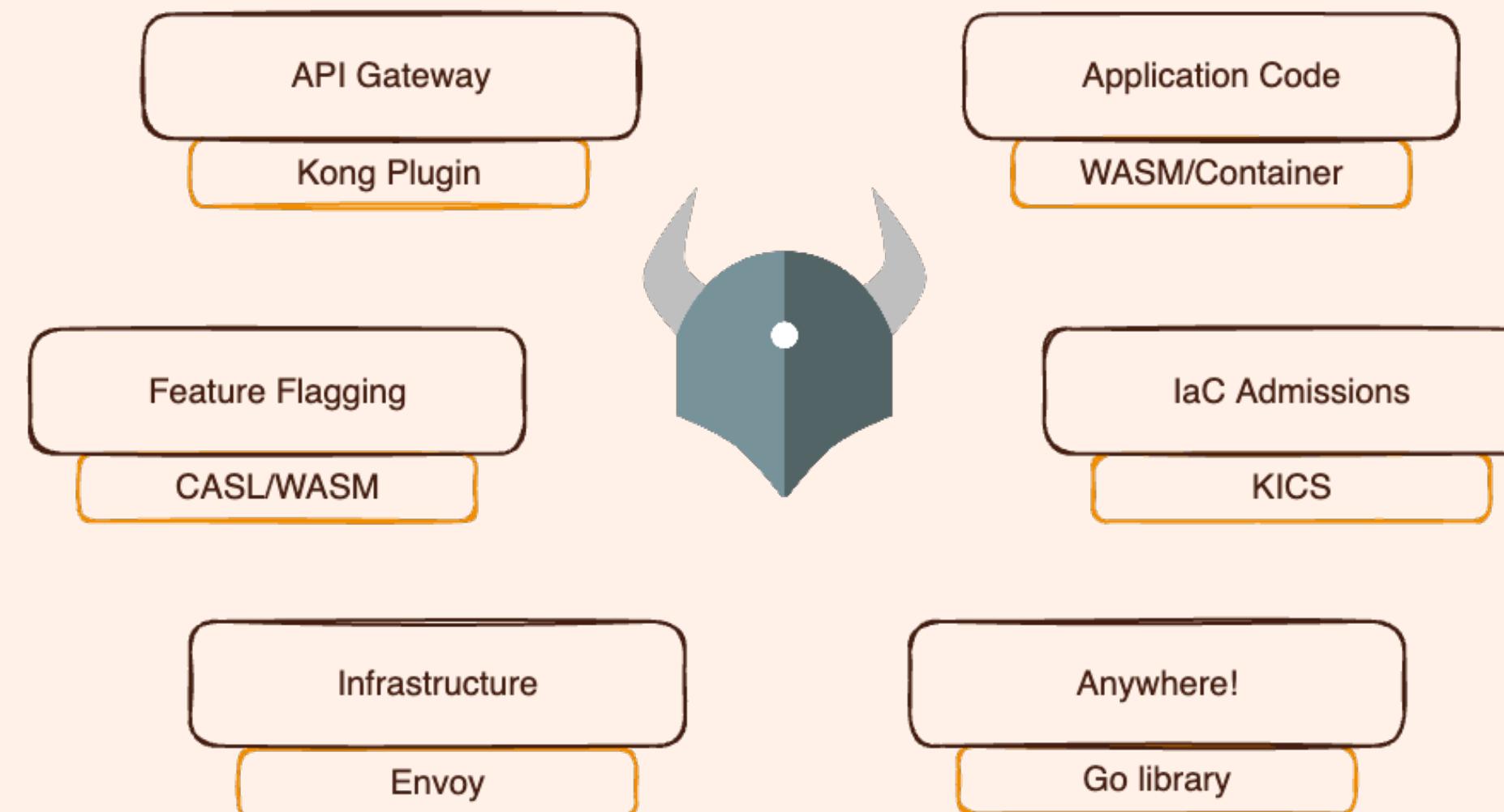
validate_classification(user_role, doc_classification) {
    role_permissions[user_role][_] == doc_classification
}

validate_dynamic_rules(user, document) {
    dynamic_rules[_](user, document)
}
```

Open Policy Agent

- Started as a multi-purpose policy engine
- Using Rego as a policy language
- As part of their GTM, they are focusing on Kubernetes admission
- Works great for application-level authorization (<10 ms for decision)
- Hard to deal with large (>2gb) sets of data
- Need some tweak for efficient ReBAC support

OPA Multi Domain Support



```
permit (
    principal == PhotoApp::User::"stacey",
    action == PhotoApp::Action::"viewPhoto",
    resource
)
when { resource in PhotoApp::Account::"stacey" };
```

AWS Cedar

- Started as a language for application-level authorization
- Not like AWS IAM, it's a language that can be used in any application
- AWS released it as open source, but also offer it as a service (AVP)
- Use Dafny language to provide scientific proof for correctness and performance
- (Still) hard to deal with unstructured data – no real ReBAC support
- Faster option for ABAC decisions
- Support audit, static analysis, and partial evaluation out of box

```
model
  schema 1.1

type organization
  relations
    define member: [user] or owner
    define owner: [user]
    define repo_admin: [user, organization#member]
    define repo_reader: [user, organization#member]
    define repo_writer: [user, organization#member]

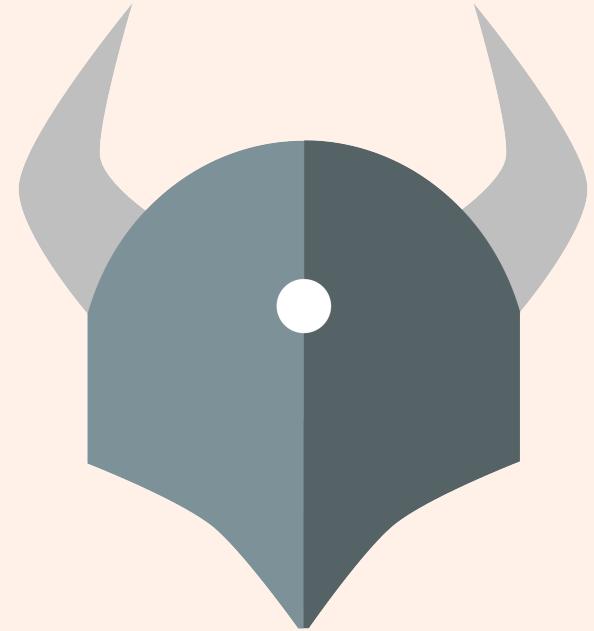
type repo
  relations
    define admin: [user, team#member] or repo_admin from owner
    define maintainer: [user, team#member] or admin
    define owner: [organization]
    define reader: [user, team#member] or triager or repo_reader from owner
    define triager: [user, team#member] or writer
    define writer: [user, team#member] or maintainer or repo_writer from owner

type team
  relations
    define member: [user, team#member]

type user
```

OpenFGA

- Open source implementation of Google Zanzibar that fit any scale
- Configured with JSON/YAML/DSL
- Need to deploy one centralized platform for decision making
- Backed and maintained by Auth0 and use by them for authorization
- Contain many application-specific features



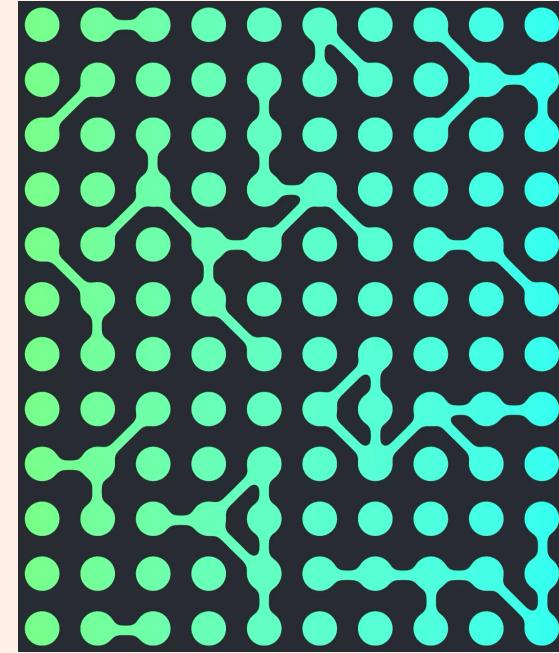
Open Policy Agent

- Multi-domain support
- Complex language
- Great for large-scale enterprise



AWS Cedar

- Simplest language
- Application oriented
- Not support ReBAC
- Best to start with

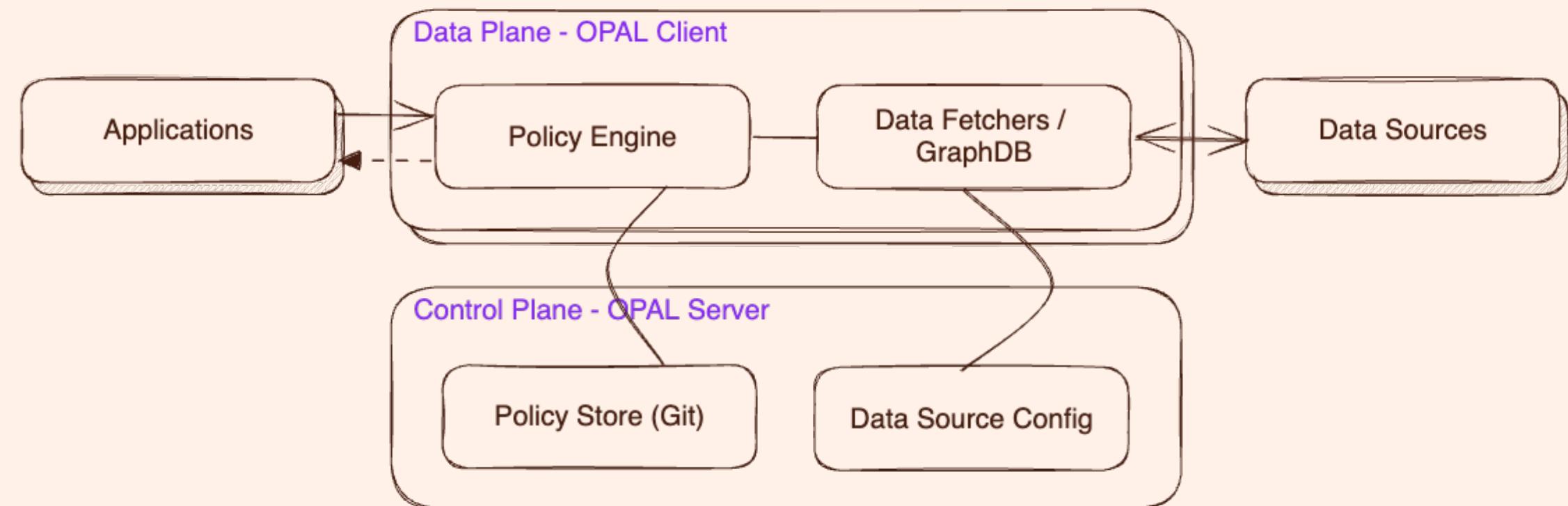


OpenFGA

- Best for ReBAC
- No new language
- Hard to desentralize
- No ABAC support

Company	Language	Models
 Netflix	Rego	RBAC, ABAC, ReBAC
 AirBnB	Himeji (Zanzibar)	ReBAC
 Uber	CEL	ABAC
 Google	Zanzibar	ReBAC
 Reddit	Rego	RBAC, ABAC
 AWS	Cedar	RBAC, ABAC

OPAL - Open Policy Administration Layer



An End-to-End *Batteries Included* Authorization Service

OPAL Specifications

- End-to-End Authorization
- Policy Engine Plugins
- GitOps Policy Flow
- Data Sharding ReBAC
- Centralized Config
- Decentralized Decision
- Pub/Sub Policy & Data Updates
- Extensible Data Fetchers
- Plugable Data Sources

Structure



Design Pattern



Framework



Declarative



Thank You 🙏

Share Your Thoughts
and Get the Slides ➡️

Join OPAL Slack
Community [io.permit.io/
opal-slack](https://io.permit.io/opal-slack)

