
Backend
Developoment
Typescript Express Chapter 4

Agenda

- What a good project look like
- How can we work together
- Git Strategy
- Project Structure
- Formatter
- API Testing
- Logging Monitoring

What a good project look like

Clear Goal

- Know what to do
- Know why to do it
- Know how to do it

Clear Document

- Where the project locate
- Where the project document locate
- Where the project meeting & project update
- Know the team process
- Know the team formatter

Clear Task

- Atomic Task
- Dynamic timeline
- Clear milestone
- Reasonable task
- Reasonable workload

How can we work together

Project Sharing Service

- GitHub
- GitLab
- BitBucket
- Private Git Service

Project Documenting Service

- Confluent
- Notion
- Lotus Note
- Google Docs
- Microsoft Word

How can we work together

Project Commute/planning

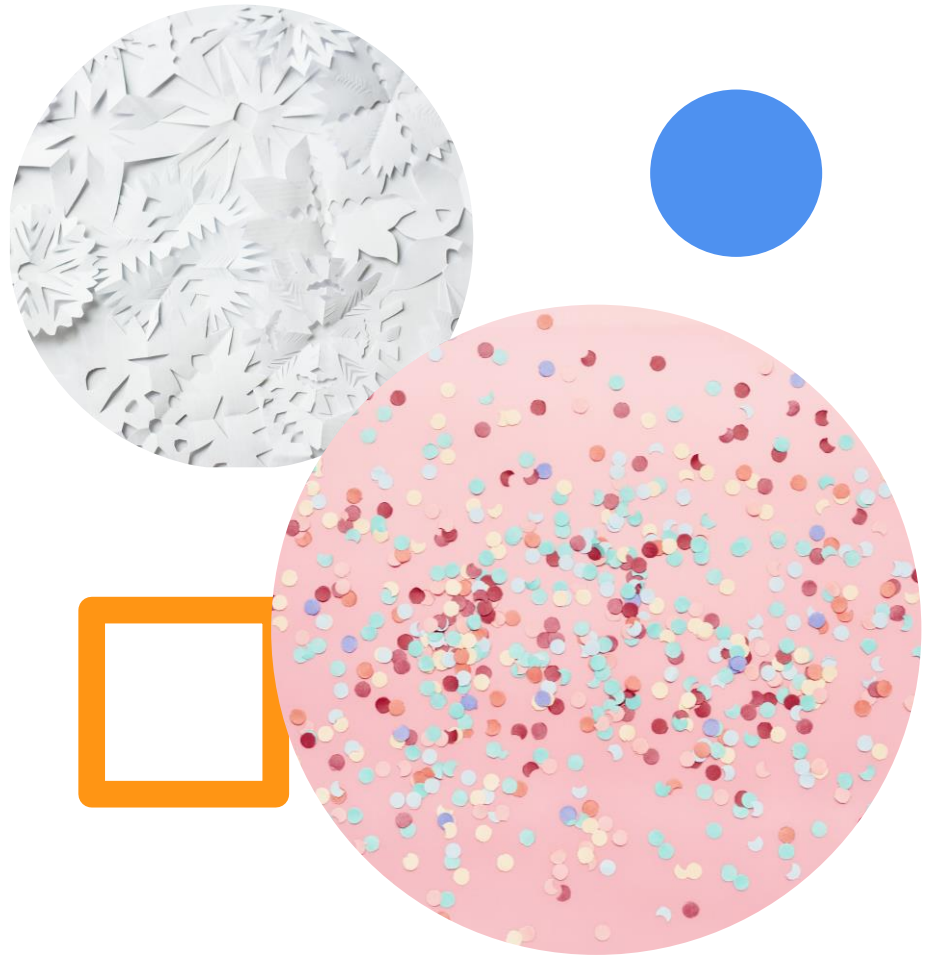
- Microsoft Team / Microsoft Todo
- Slack / JIRA

Project Formatter Service

- Private code style (depend on team)
- ES lint
- Pretteir

Example

- Project: Movie Management
- Git Repository Service: GitHub
- Documentation : Notion
- Project Planning: GitHub Project
- Project Communicate: Discord
- Project Process: Agile, Scrum framework



Git Strategy

- One of the processes in the project determines what each team member should do when the project starts.
- Everyone in the team see the same picture
- Easier to Plan and Deliver the product

Git Strategy Example (1)

- The project has 4 branches:
  - **Main** : Maintains each version of the product.
  - **Development** : Serves as the workspace for the project.
    - **Task-<id>-<summarized-taskname>:** Created from the development branch.
  - **Dev-release**: a development server release version.
  - **Prod-release**: a production server release version

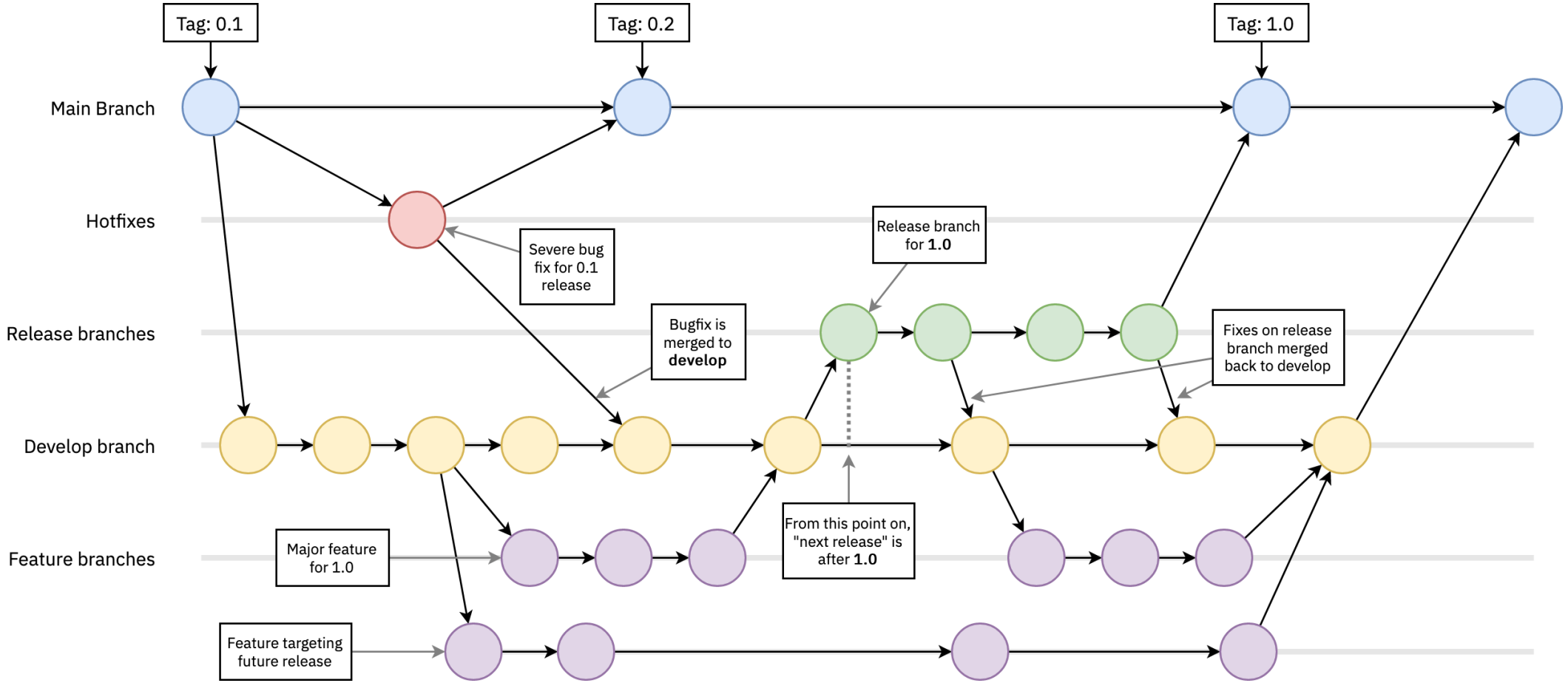
Git Strategy Example (2)

- The project has 6 developer:
  - **1 Lead** : lead developer
    - Managing team
    - Keep project on timeline
    - Create team culture
    - Communicate with PM, and client
  - **2 Senior** : senior developer
    - Review PR (keep code in place)
    - Keep the culture among the team
    - Create a based project (create database by design, create project structure that make everyone follow)
    - Create PR template
  - **3 Junior**: just dev

Git Strategy Example (3)

- The process
  - 1. When a task is assigned, the person in charge needs to create a branch from the development branch following this format: <task-id>-<task-name> (e.g., momag15-create-movie).
  - 2. When the task is finished, open a pull request (PR) using the PR template and assign one of the senior members as a reviewer.
  - 3. After the review is completed and approved, the reviewer merges the branch into the development branch.
  - 4. If the product needs to be deployed to production, assign the lead developer to review and merge the branch into the production branch.

Git Strategy



Project Structure

- A structure of the project that make everyone see the same project and the same picture

Example

```
src/
├── modules/                          # Core domain modules
│   ├── [ModuleName]/                 # Example: 'User' or 'Order'
│   │   ├── domain/                   # Contains domain logic
│   │   │   ├── entities/             # TypeORM entities (domain objects)
│   │   │   ├── value-objects/        # Value objects used in the domain
│   │   │   └── services/             # Domain services for complex logic
│   │   ├── application/              # Application services (use cases)
│   │   │   └── dto/                  # Data Transfer Objects for use cases
│   │   ├── infrastructure/           # Infrastructure implementations
│   │   │   ├── repositories/         # TypeORM repository implementations
│   │   │   └── orm/                  # Database-specific TypeORM configurations
│   │   ├── presentation/             # REST controllers and routes
│   │   └── tests/                    # Unit and integration tests
│   └── shared/                       # Shared domain logic between modules
│       ├── entities/                 # Shared entities or base models
│       ├── value-objects/            # Common value objects used across modules
│       └── services/                 # Shared services (e.g., domain events)
├── config/                           # Configuration files (e.g., database, env)
├── infrastructure/                   # Cross-cutting infrastructure concerns
│   ├── database/                     # TypeORM connection setup and migrations
│   ├── http/                         # Express app and middlewares
│   └── logger/                       # Logging utilities
├── utils/                            # Utility functions
└── app.ts                            # Main app entry point
```

Formatter

- A formatter that use along the project
  - Document formatter
    - Git Message Formatter: Husky
    - Git PR Template
    - Planning Template
  - Development formatter
    - ES lint
    - Pretteir

Example

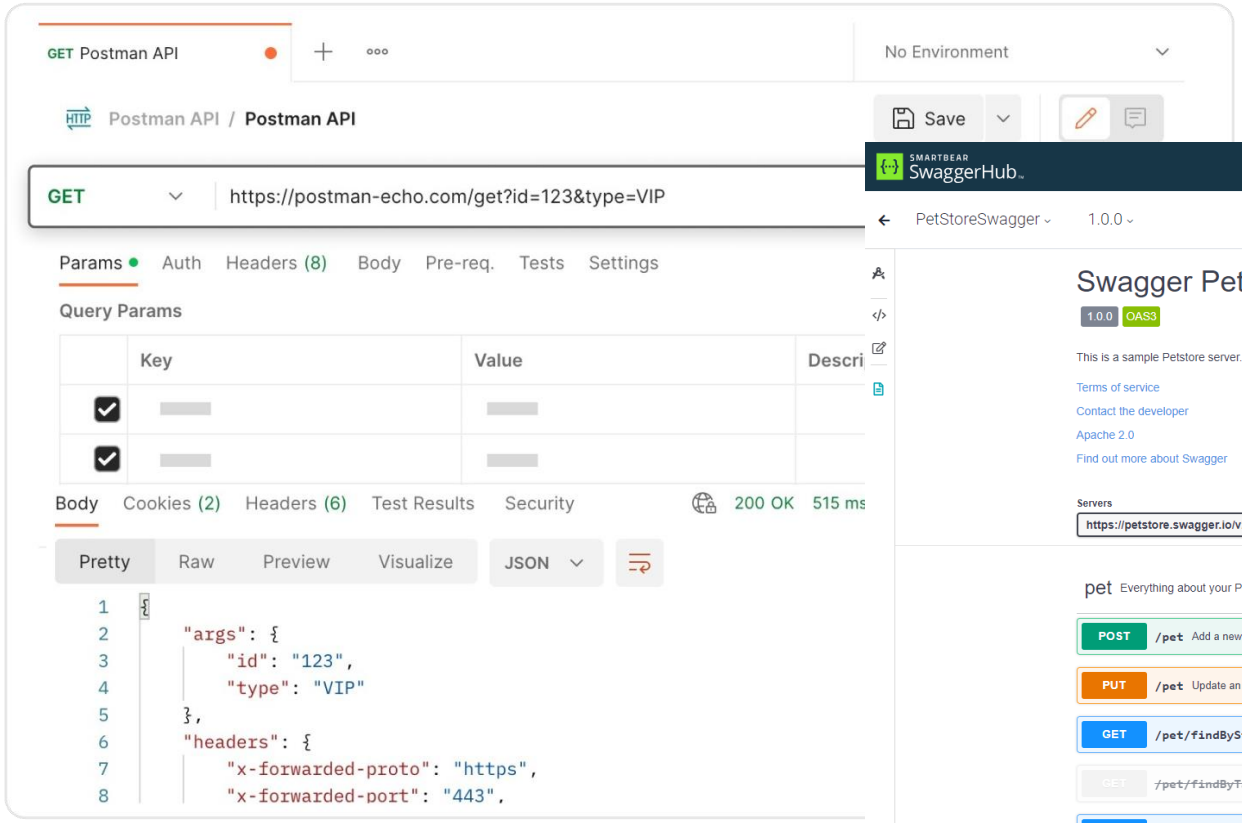
```
import globals from 'globals';
import pluginJs from '@eslint/js';
import tseslint from 'typescript-eslint';

export default [
    { files: ['**/*.{js,mjs,cjs,ts}'] },
    { languageOptions: { globals: globals.node } },
    pluginJs.configs.recommended,
    ...tseslint.configs.recommended,
    {
        rules: {
            'no-cond-assign': 'warn',
        },
    },
];
```

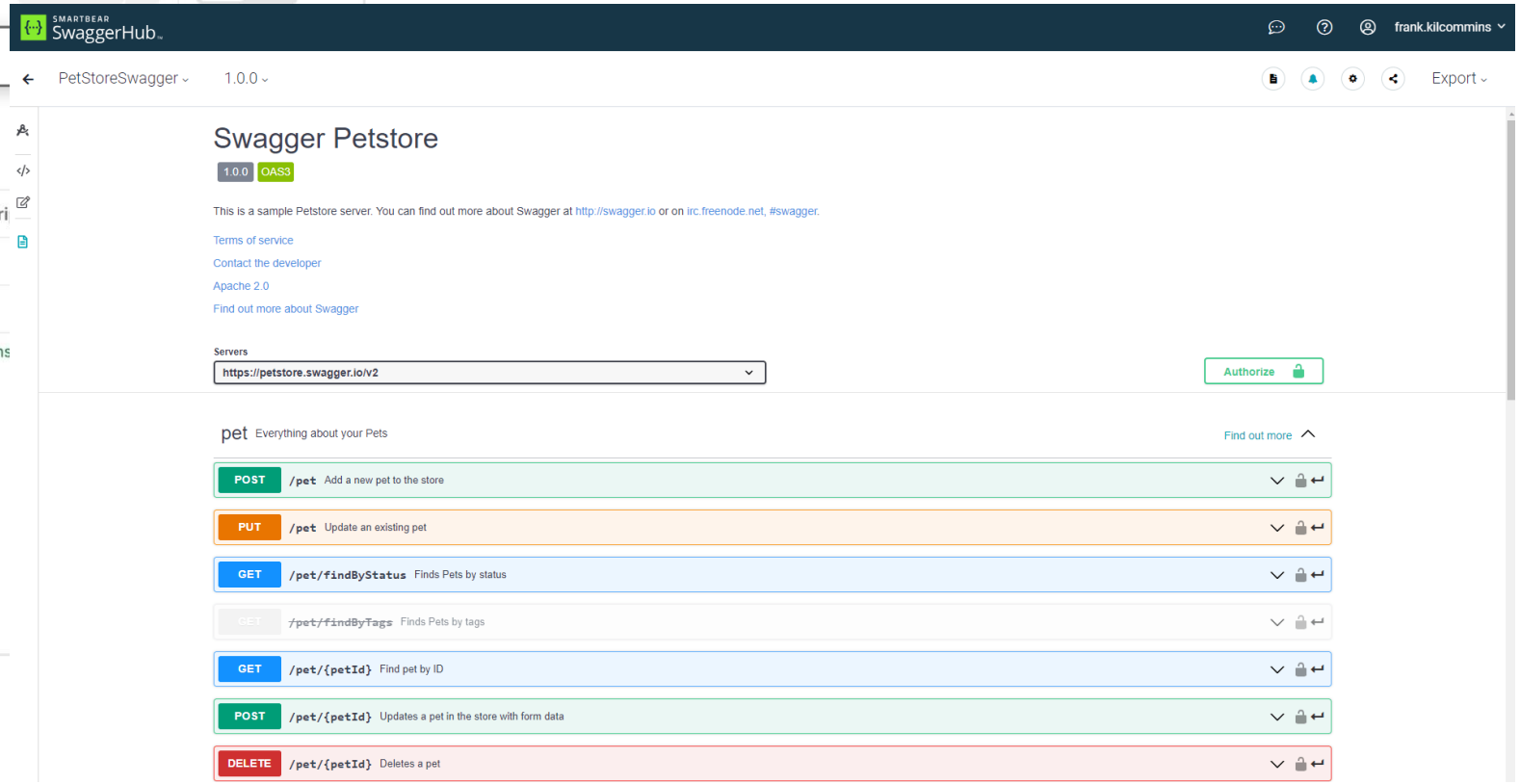
```
{
    "tabWidth": 4,
    "useTabs": false,
    "semi": true,
    "singleQuote": true,
    "printWidth": 9999,
    "proseWrap": "preserve"
}
```

API Testing

- A Place for Testing API that team develop
  - Postman
  - Insomnia
  - API Dog
  - Open API
  - Swagger

Example

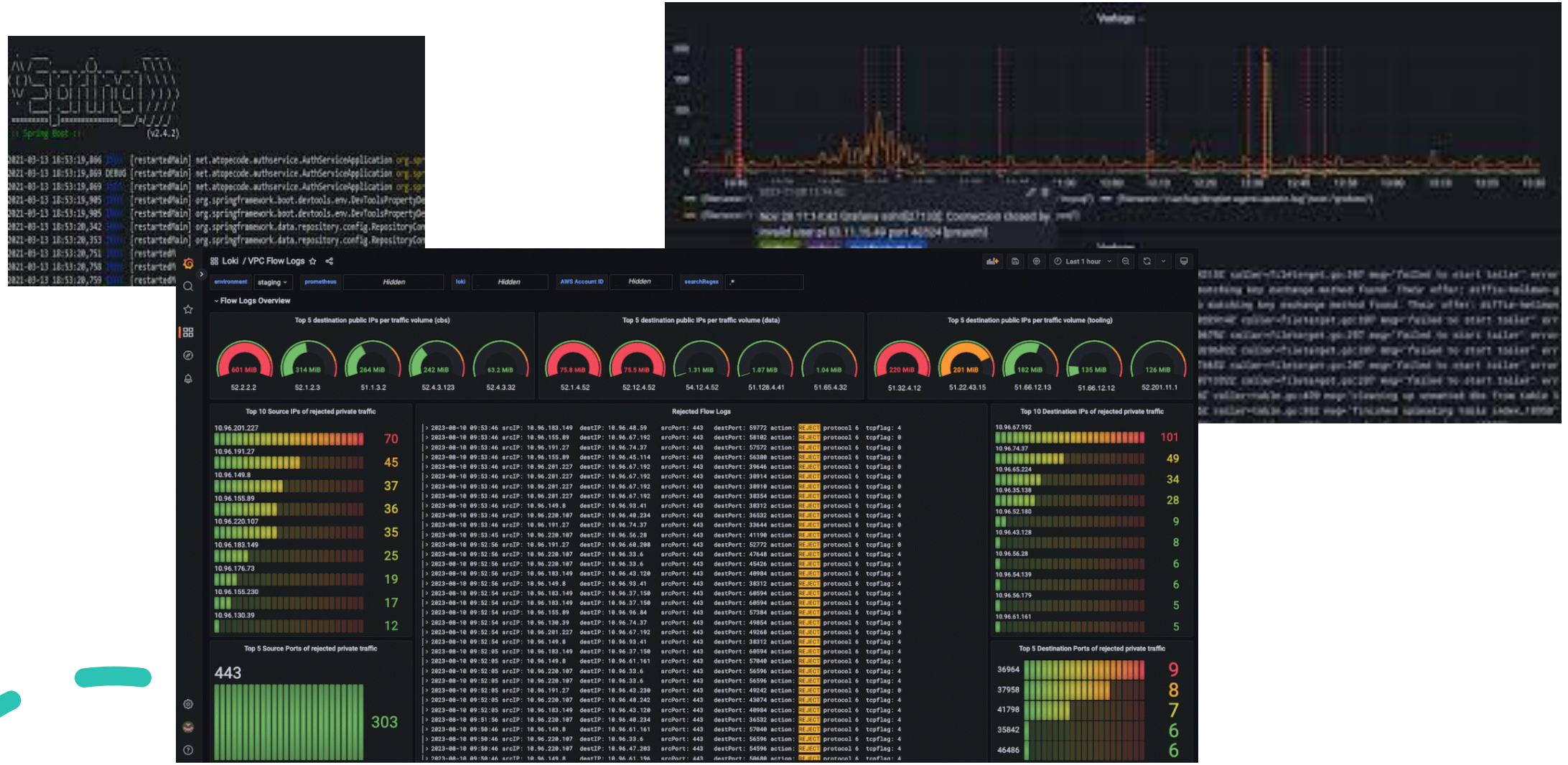

GET Postman API
No Environment
Postman API / Postman API
Save
GET
https://postman-echo.com/get?id=123&type=VIP
Params
Auth
Headers (8)
Body
Pre-req.
Tests
Settings
Query Params
Key
Value
Body
Cookies (2)
Headers (6)
Test Results
Security
200 OK
515 ms
Pretty
Raw
Preview
Visualize
JSON
{
"args": {
"id": "123",
"type": "VIP"
},
"headers": {
"x-forwarded-proto": "https",
"x-forwarded-port": "443",


SmartBear SwaggerHub
frank.kilcommins
PetStoreSwagger
1.0.0
Export
Swagger Petstore
1.0.0
OAS3
This is a sample Petstore server. You can find out more about Swagger at http://swagger.io or on irc.freenode.net, #swagger.
Terms of service
Contact the developer
Apache 2.0
Find out more about Swagger
Servers
https://petstore.swagger.io/v2
Authorize
pet
Everything about your Pets
Find out more
POST /pet Add a new pet to the store
PUT /pet Update an existing pet
GET /pet/findByStatus Finds Pets by status
GET /pet/findByTags Finds Pets by tags
GET /pet/{petId} Find pet by ID
POST /pet/{petId} Updates a pet in the store with form data
DELETE /pet/{petId} Deletes a pet

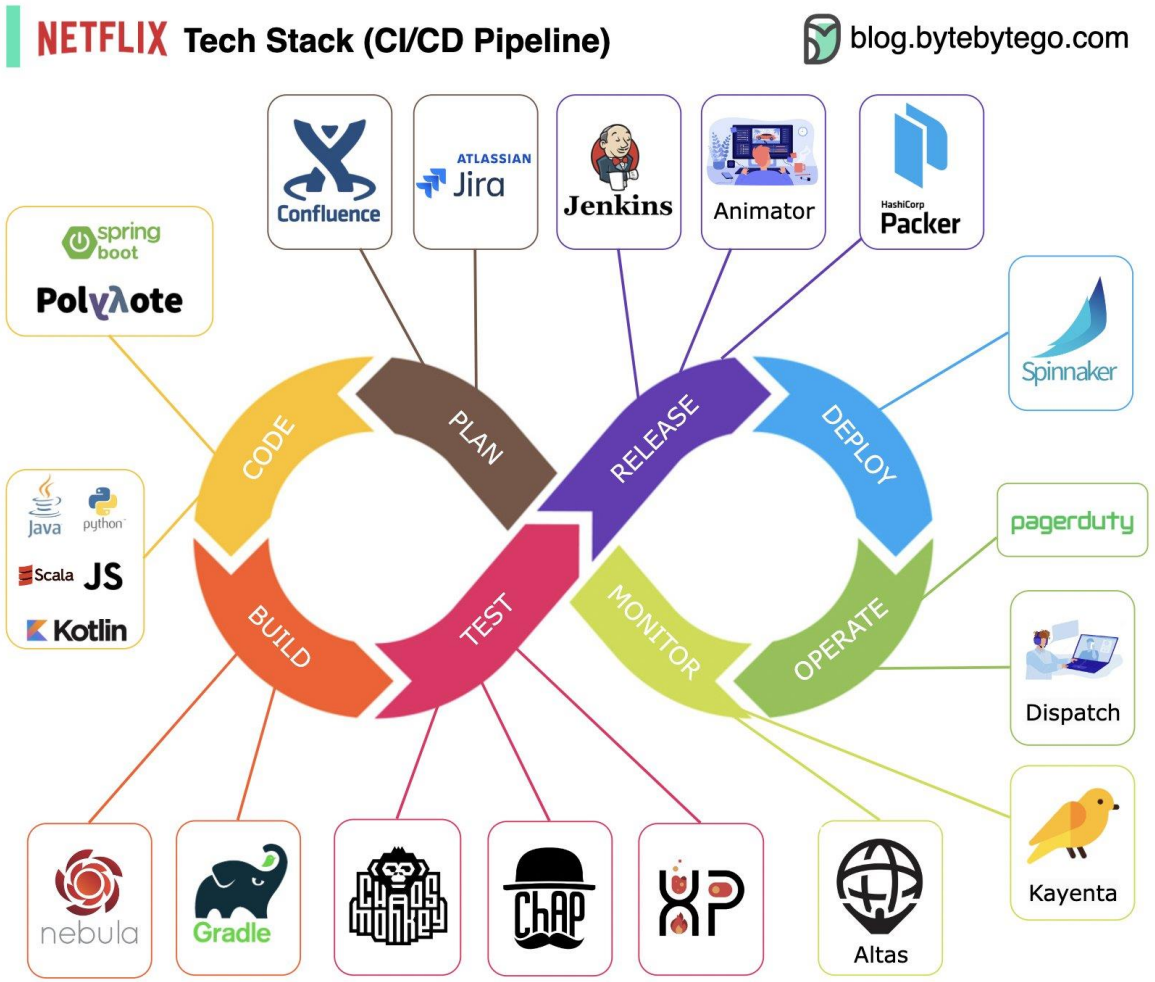
Logging Monitoring

- A Place to debugging and observed the behavior of the product
  - Grafana
  - Loki
  - Local Console logging
  - ELK stack
    - Elastic search
    - Log stash
    - Kibana

Example


Loki / VPC Flow Logs
Flow Logs Overview
Top 5 destination public IPs per traffic volume (cbs)
Top 5 destination public IPs per traffic volume (data)
Top 5 destination public IPs per traffic volume (tooling)
Top 10 Source IPs of rejected private traffic
Rejected Flow Logs
Top 10 Destination IPs of rejected private traffic
Top 5 Source Ports of rejected private traffic
Top 5 Destination Ports of rejected private traffic

An Ideal Setup



Now Let see the a proper setup

oat431/fourth_ts_class

Work Shop 4

- From previous Workshop
  - Create the same project but with a proper backend setup

  - Submit: oat431@gmail.com
  - Title : [your nickname] – 4th workshop
  - Attach: your workshop repository

Next Chapter

Relational Database