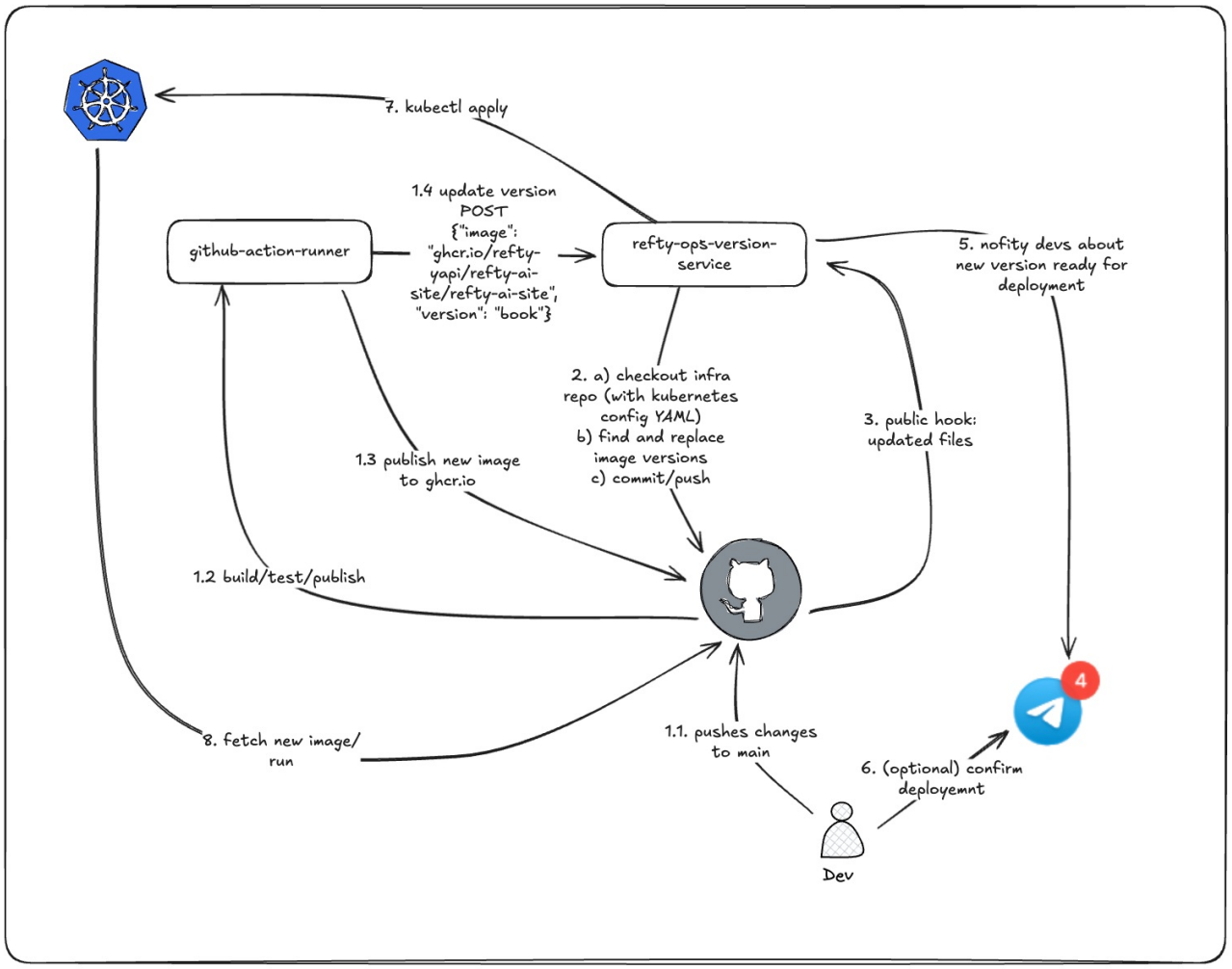
DevOps Infrastructure Service Implementation



Overview

This task involves implementing a microservice that automates the process of updating container image versions across Kubernetes deployment configurations. The service will integrate with GitHub to manage infrastructure as code and streamline the CI/CD pipeline.

Context

1. **Infrastructure**: We run our infrastructure within Kubernetes using Linux containers.
2. **Version Control**: We rely on GitHub as our code repository and container registry.
3. **CI/CD Goals**: We want to improve our CI/CD process by automating image version updates.
4. **Architecture**: The above diagram illustrates the target architecture we want to achieve.
5. **Current State**: We already have a github-runner pod running in our Kubernetes cluster.
6. **Next Phase**: The next step is to implement the refty-node-versions-service.

Technical Requirements

Service Implementation

1. **Technology Stack**: Implement in TypeScript or Python (choose based on your expertise).
2. **API Design**: Create a RESTful service with a POST endpoint (determine the appropriate path).
3. **Request Format**: Accept JSON payloads with the following structure:

```
{
  "image": "ghcr.io/refty-yapi/refty-node/refty-node",
  "version": "05-06-42a252"
}
```

Core Functionality

1. **Repository Integration**: Use https://github.com/alun/refty-infra-test as the reference infrastructure repository.
2. **Image Updates**: The service should update all YAML files containing the specified image across the repository.
3. **Version Management**: Replace image versions in format image: ghcr.io/refty-yapi/refty-node/refty-node:05-06-42a252.
4. **Git Operations**:
    - Create a single commit with all changes
    - Push to your fork of the refty-infra-test repository
    - Handle multiple YAML files that may reference the same image

GitHub Integration Requirements

1. **Authentication**: Configure GitHub access with appropriate tokens and permissions.
2. **Repository Management**: Fork the refty-infra-test repository for development.
3. **API Access**: Ensure the service can read from and write to the target repository.

Implementation Guidelines

Development Setup

1. **Repository Fork**: Create your own fork of the refty-infra-test repository.
2. **Configuration**: Set up GitHub authentication (personal access token or GitHub App).
3. **Environment**: Configure repository name and access credentials.

Code Quality Standards

1. **Error Handling**: Implement proper error handling for API calls and Git operations.
2. **Logging**: Add comprehensive logging for debugging and monitoring.
3. **Validation**: Validate input parameters and handle edge cases.
4. **Documentation**: Include clear API documentation and setup instructions.

Deployment (Optional, if you want to achive extra points)

1. **Containerization**: Package the service as a Docker container.
2. **Kubernetes Deployment**: Create deployment manifests if deploying to K8s.
3. **Environment Variables**: Use environment variables for configuration.

Deliverables

1. **Source Code**: Complete implementation pushed to your GitHub repository.
2. **Documentation**: README with setup and usage instructions.
3. **API Documentation**: Clear endpoint specification and examples.
4. **Repository Link**: Share the link to your implementation repository.

Timeline

- **Estimated Duration**: 1-3 hours (depends on your experience)
- **Priority**: Focus on core functionality first, then optional deployment features.

Success Criteria

- Service accepts POST requests with image and version parameters
- Updates all YAML files containing the specified image
- Creates and pushes a single commit to the repository
- Handles multiple files and edge cases gracefully
- Includes proper error handling and logging
- Code is well-documented and maintainable