JustInTimeout: Enacting Dynamic Network Timeouts via Distributed Feedback

Presenters:
Dylan Wilson
Nisarg Chokshi
Yash Trivedi

Introduction

Timeouts are defined as:

A **specified period of time** that will be allowed to elapse in a system before a specified event is to take place, unless another specified event occurs first; in either case, the period is terminated when either event takes place.

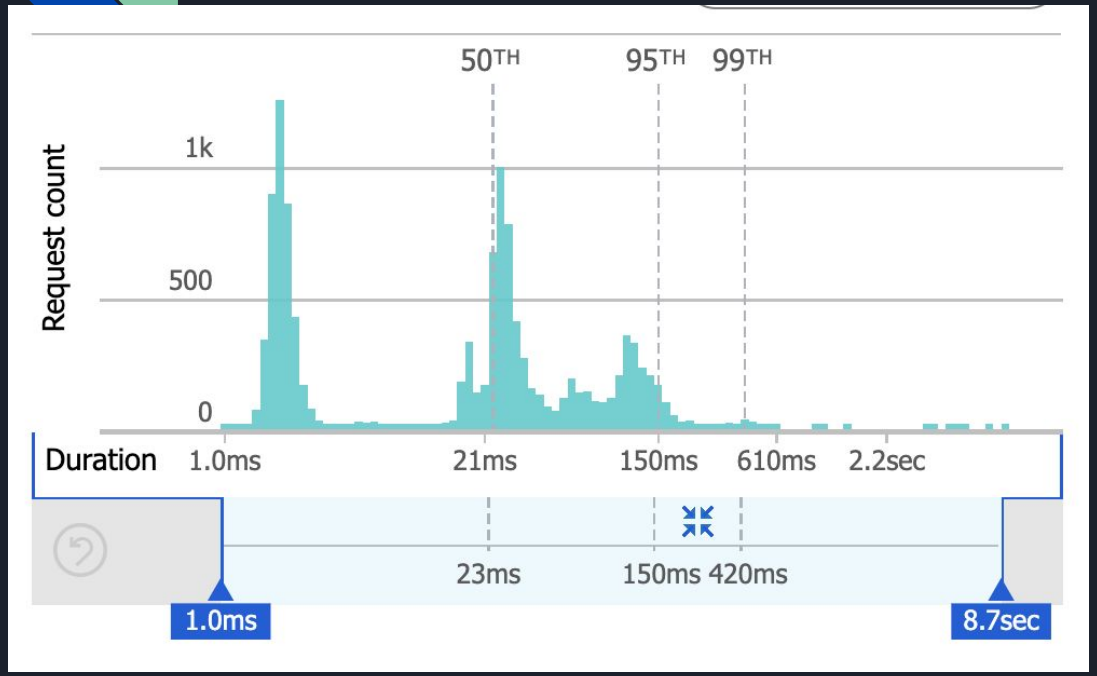
This means that timeouts are a way of the system deciding that the action that was supposed to happen, will not happen due to some erroneous condition.

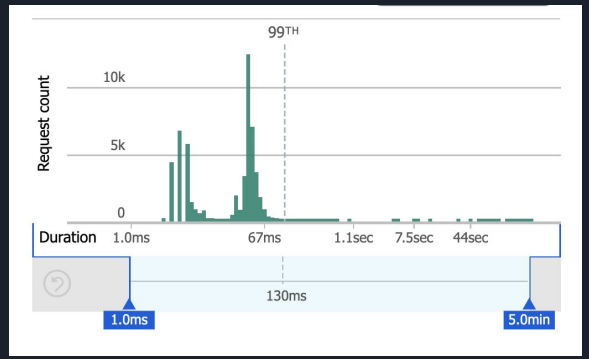
JustInTimeout is our way of combating the effervescent problem of bad timeouts by making the selection of this parameter non random.

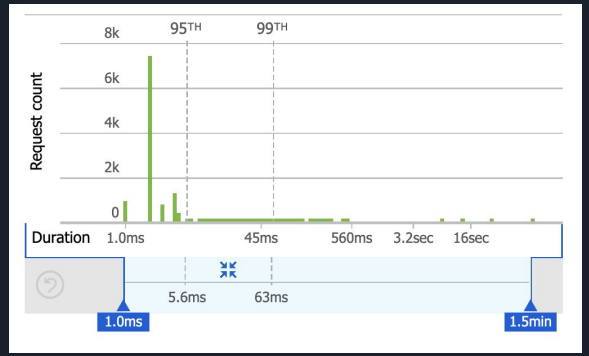
500 ms

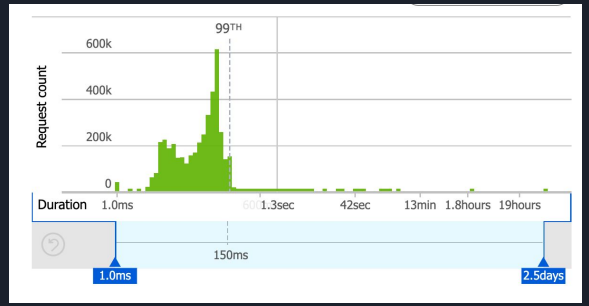
Bug or No Bug?

A real example









Goldilocks and the three timeouts

Too Short - Keep hitting the endpoint or never get service

Too Long - Hang or slow throughput if the endpoint doesn't respond

Just Right - Wait just a bit longer than you expect the call to take

Selecting a Timeout value is essentially about gathering expected service times for each endpoint and *adapting as they change*.

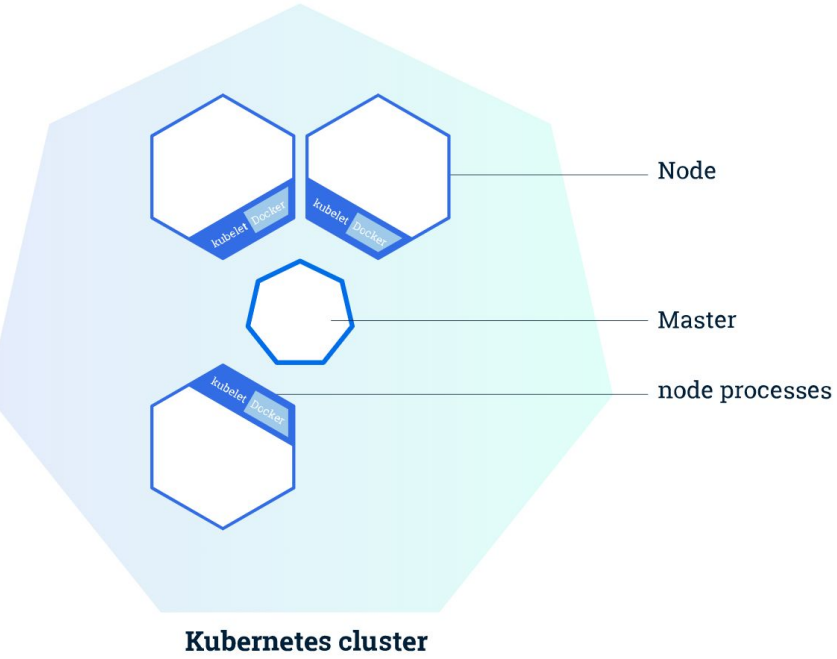
An aside: Build for robustness

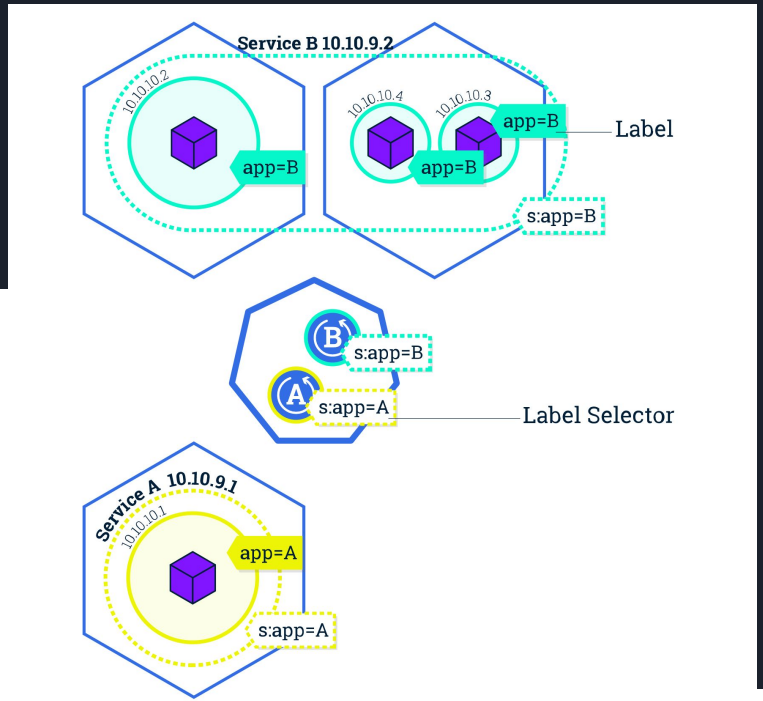
Avoid timeout fragility - build retry, circuit breaker into your work

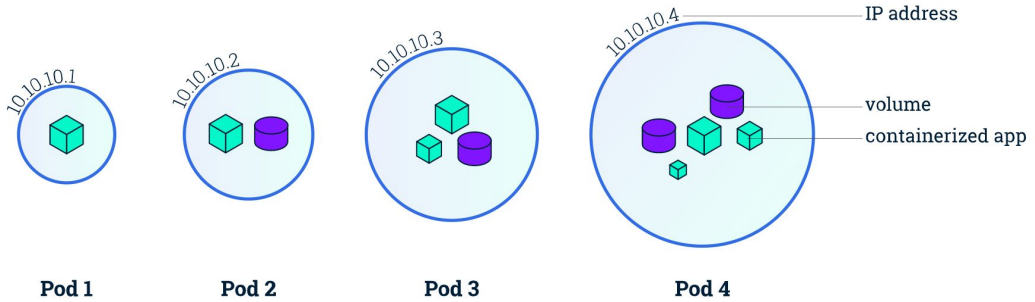
Adopt a library:

- Polly
- Jolly
- Others

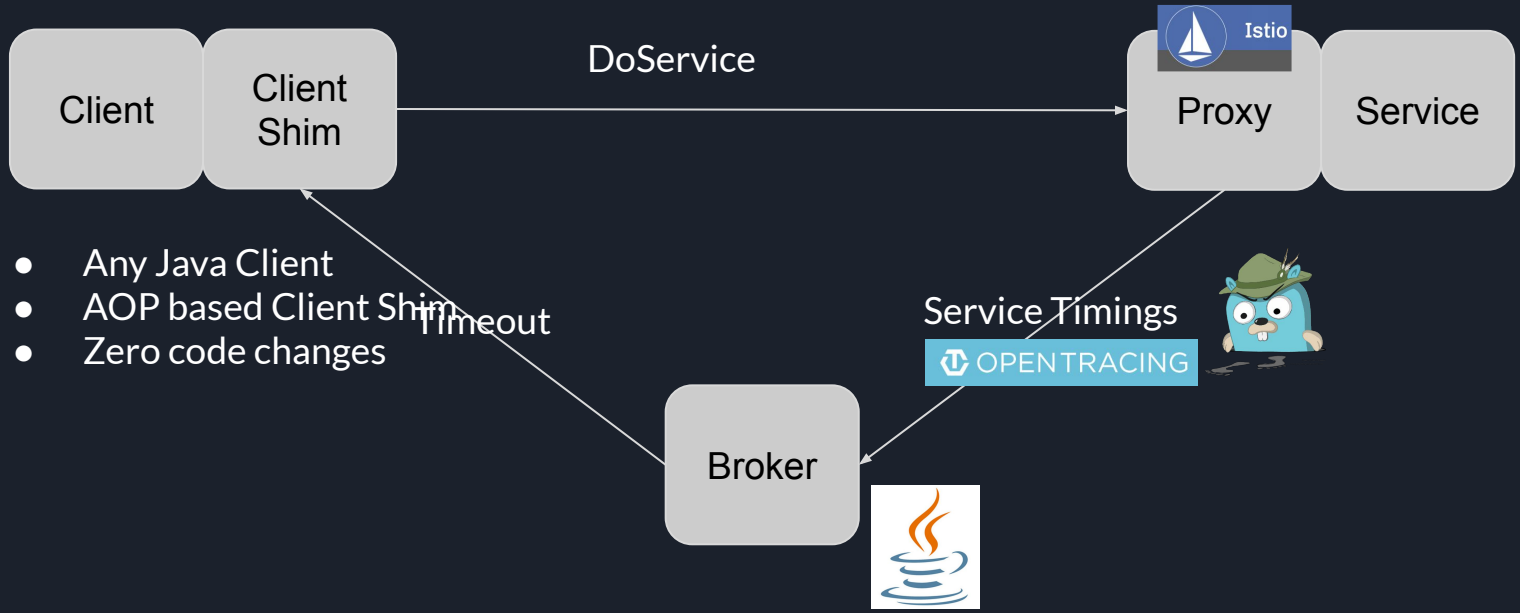
Kubernetes 101





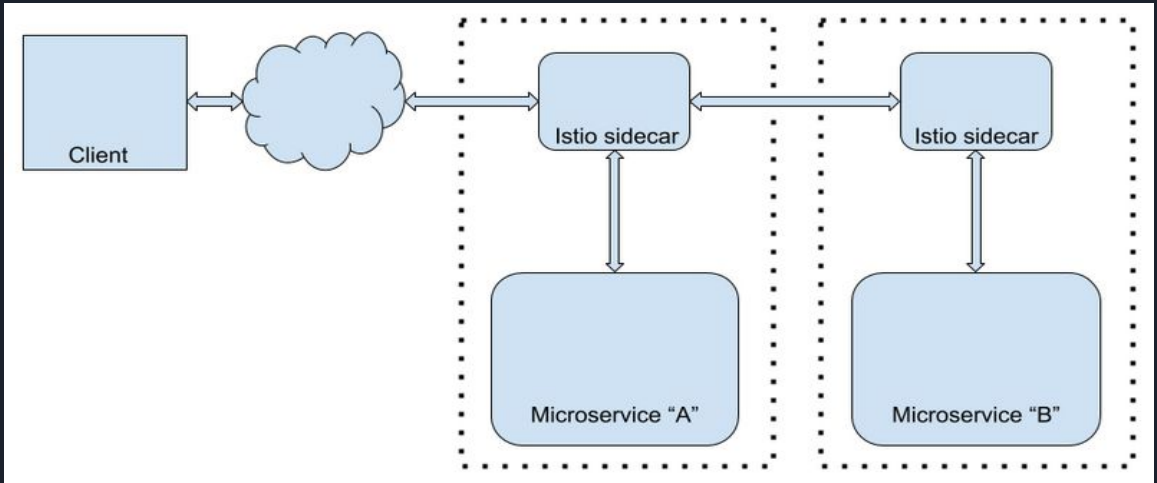


JustInTimeout: Logical View



- Any Java Client
- AOP based Client Shim
- Zero code changes

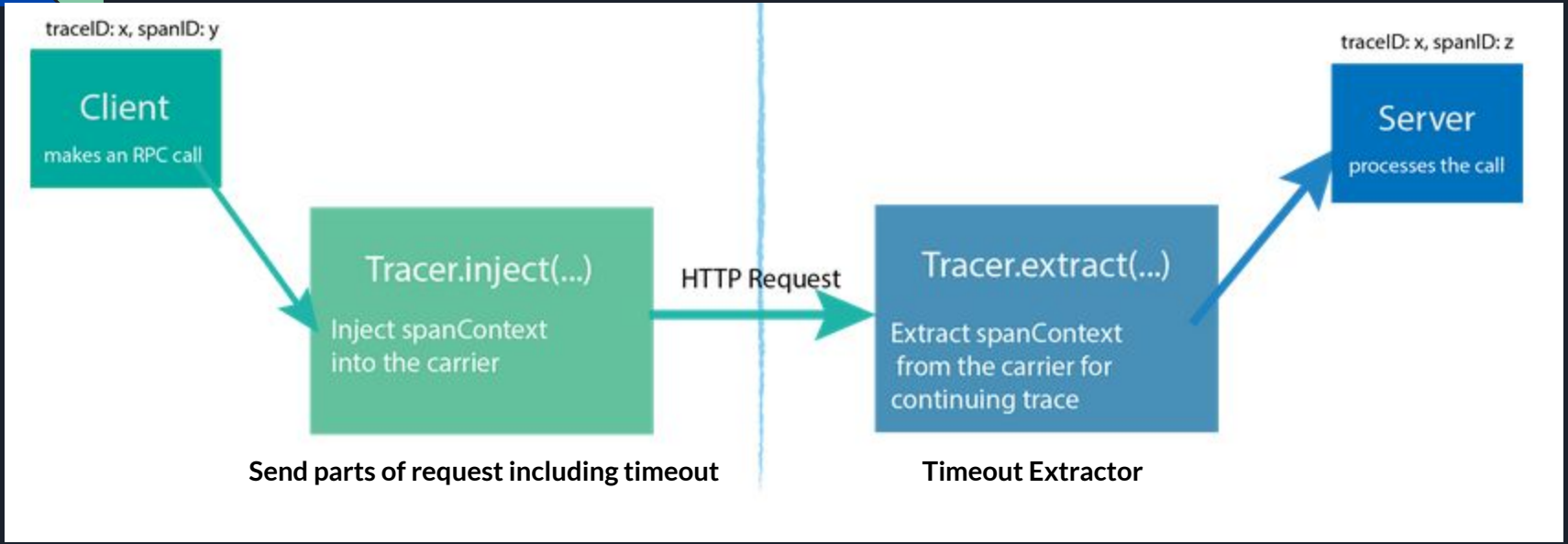
Application Proxy using Istio



Istio uses an extended version of the Envoy proxy. It is used as a high performance proxy to mediate all inbound and outbound traffic for all services in the service mesh.

This sidecar deployment allows Istio to extract a wealth of signals about traffic behavior as attributes. Istio can, in turn, use or send these attributes to any listener configured (In our case OpenTracing/Jaeger).

Timeout Extractor



Jaeger is a distributed tracing system based on the OpenTracing specification. It is used for monitoring and troubleshooting microservices-based distributed systems, including Distributed context propagation, Root cause analysis, Service dependency analysis and Performance / latency optimization.

Timeout Predictor (Broker)

This block would be responsible for collecting the timeout values received from the Timeout Extractor.

The data collected would then be run by a statistical model to get a prediction for an appropriate new value of timeout.

This value will be passed on to the Client Shim which would then be responsible for injecting that into the client's future request.

The main design goal for the Broker would be that it is supposed to make the predictions fast. A tradeoff can be made in terms of accuracy here.

Previous Work: TScope / TFix

TScope: System level tracing + Machine Learning = Find Timeout Bugs

TFix: TScope + Root Cause via Source Code Analysis + Recommendation from Normal Runs = Fixed Bugs

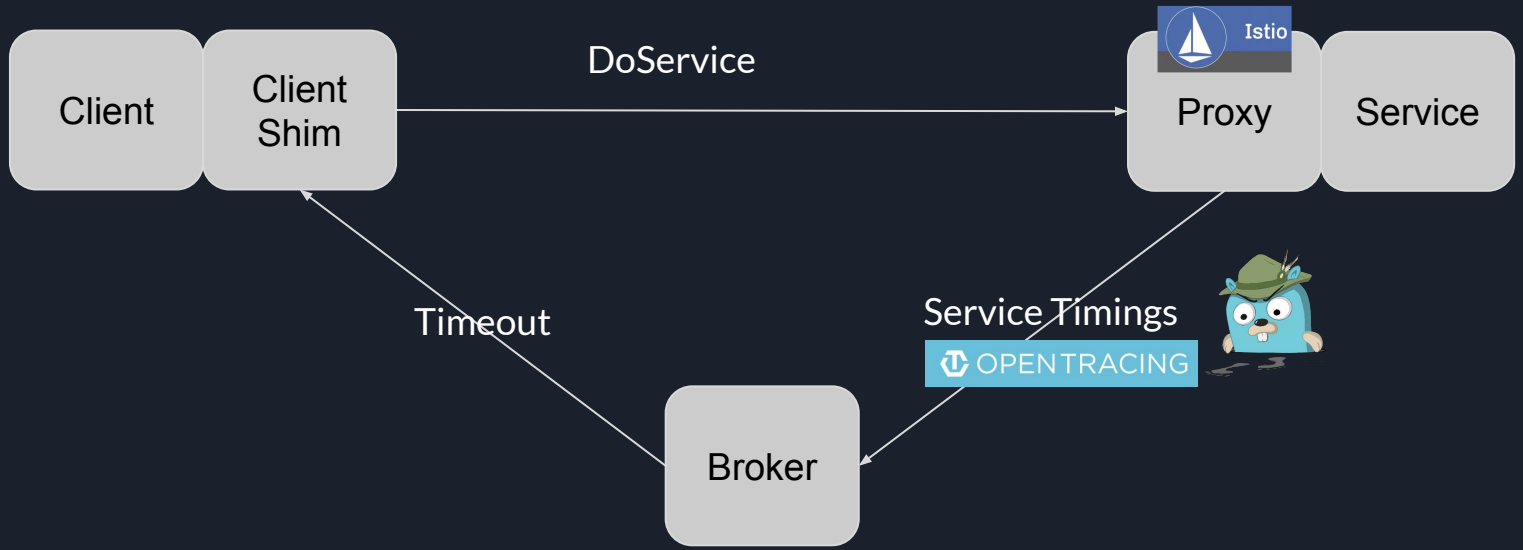
J. He, T. Dai and X. Gu, "TScope: Automatic Timeout Bug Identification for Server Systems," *2018 IEEE International Conference on Autonomic Computing (ICAC)*, Trento, 2018, pp. 1-10.

J. He, T. Dai and X. Gu, "TFix: Automatic Timeout Bug Fixing in Production Server Systems," *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, Dallas, TX, USA, 2019, pp. 612-623

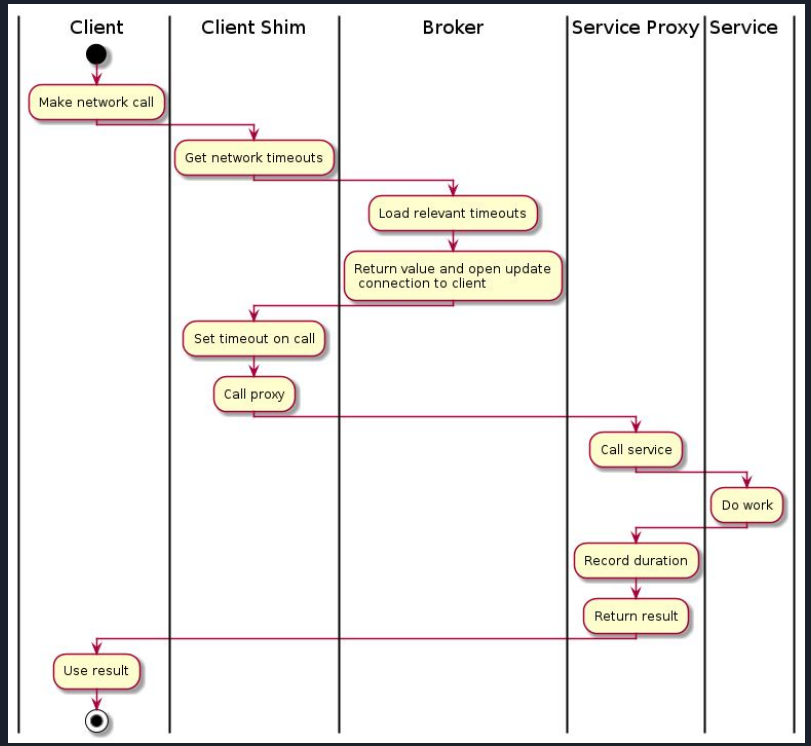
Related Works

- Adaptare: Supporting Automatic and Dependable Adaptation in Dynamic Environments. (Complex statistical model and pluggable framework)
- TScope: Automatic Timeout Bug Identification for Server Systems (Different timeout derivation mechanism and a different use for timeouts)
- Adaptive Timeout™ by Index Exchange (More fine grained approach)
- An Aspect-Oriented Approach to Assessing Fault Tolerance (Using AspectJ to get data from system calls)

JustInTimeout: Logical View

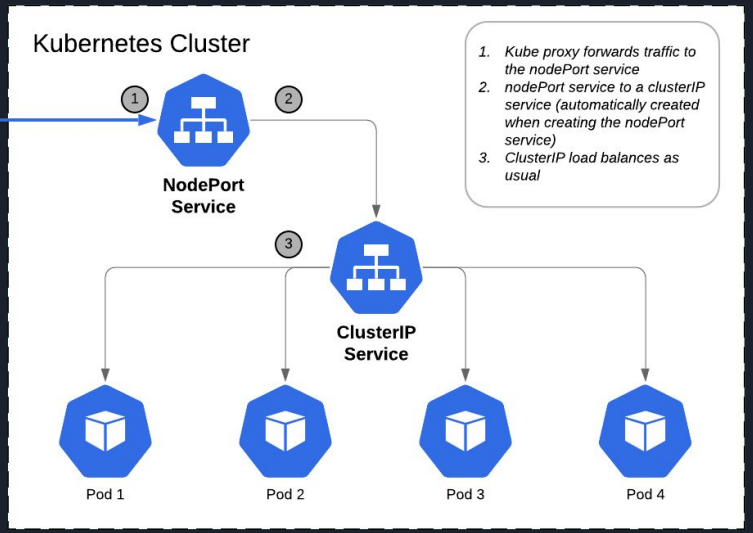


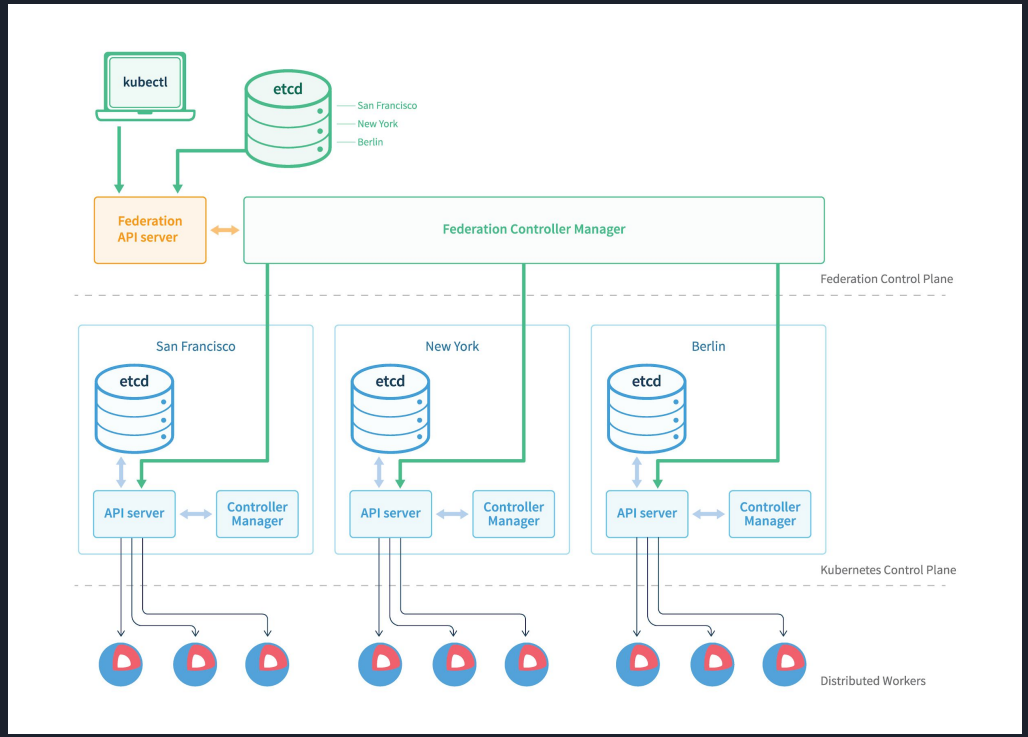
Proposed System Design



Kubernetes Networking

Pods, Services, Flat Network





Conclusion / Expected Results

<What we plan to achieve>

<Impact of the project on real world systems>