TAIWAN, TAIPEI FISSION WORKSHOP

Ta-Ching Chen, Fission Contributor

HOLA!

- Ta-Ching Chen (陳大慶)
- Software Engineer/Kubernetes Education Consultant
- Blog: https://tachingchen.com/tw/
- Email: hello@tachingchen.com

Intro / Hands-on / Q&A

40m / 2h30m / 20m

What is Serverless?

Serverless computing allows you to build and run applications and services without thinking about servers.

- AWS Lambda

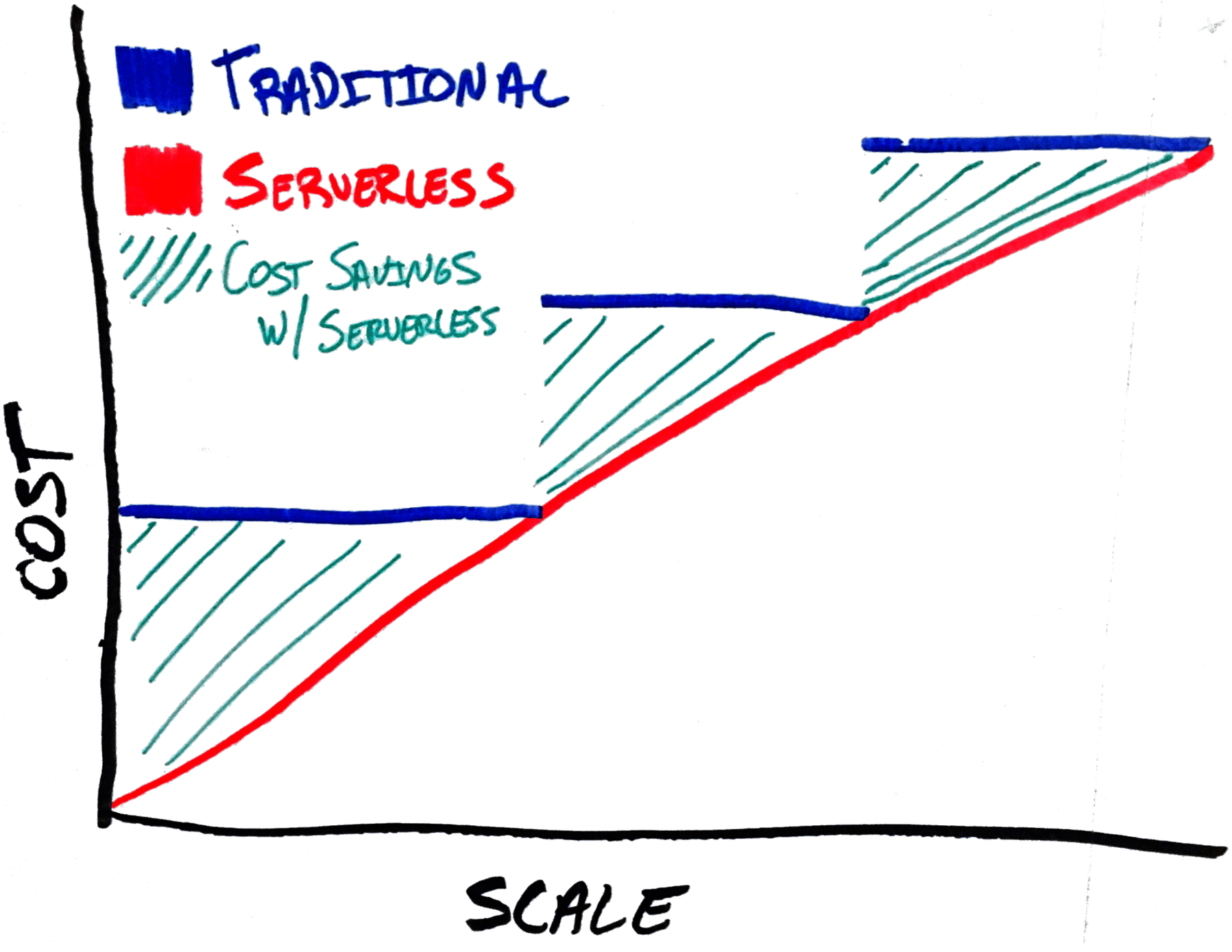
Why Serverless?

Focus on core business logic development (Agility)

Automated infrastructure management

Pay for what you use (idle = free)

Scalability



https://www.trek10.com/blog/serverless-framework-for-processes-projects-and-scale/

FaaS on Cloud Provider

Vendor lock-in

FaaS environment simulation

Data privacy/protection

Execution limits & restrictions

Lack of on-premise supportability

Expensive for heavy usage (!?)

FaaS on Kubernetes

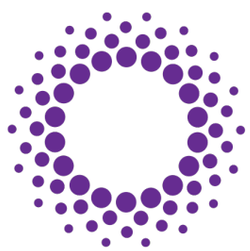
Portability

Flexible cost model (Optimization)

DevOps pipelines integration

Fission - Serverless Framework for Kubernetes

- Open source Kubernetes-native FaaS framework
  - Backed by Platform9
- On-premises/Cloud support
  - Wherever you can run Kubernetes, you can run Fission
- Tunable cost/performance tradeoffs
- Designed to be easy use, productive and fast
  - Focus on core business logic development



fission

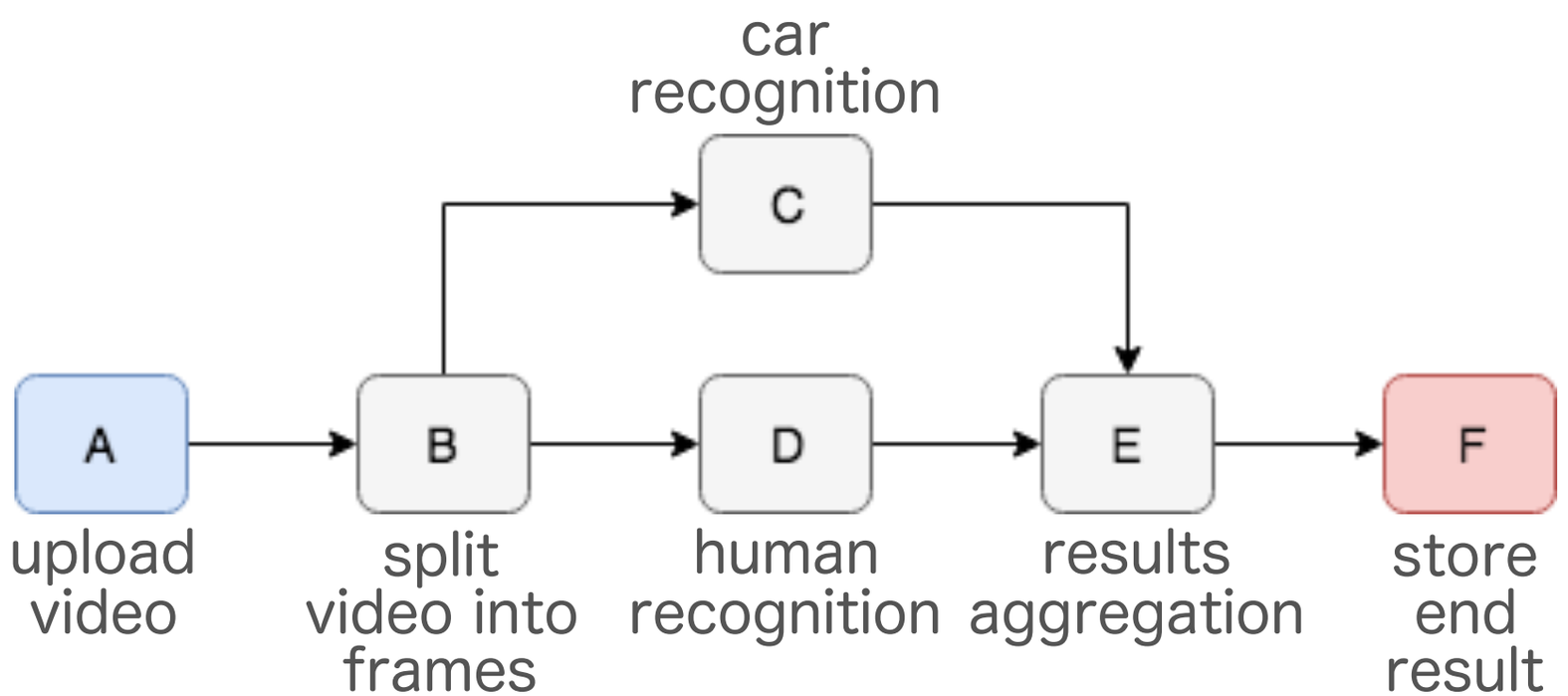
kubernetes

Fission

- Declarative Config
  - Fission resources are K8S CRDs, can be stored in YAML/JSON files
  - Spec files allow us to recreate application in different fission clusters
- Support Rich Event Sources
  - HTTP, Time, Kubewatch trigger
  - Message queue trigger: NATS, Kafka
- Rich Environment Supportability
  - NodeJs, Go, Python, Java (JVM) ..etc
  - BYOE (Build Your Own Environment)
- Build Automation
  - Builder manager creates a deployable package from source code

Fission

- More Cool Features
  - Function Log Collection
  - Canary Deployment (Beta)
  - Record & Replay (Beta)
- Fission Workflow
  - Function composition for serverless functions

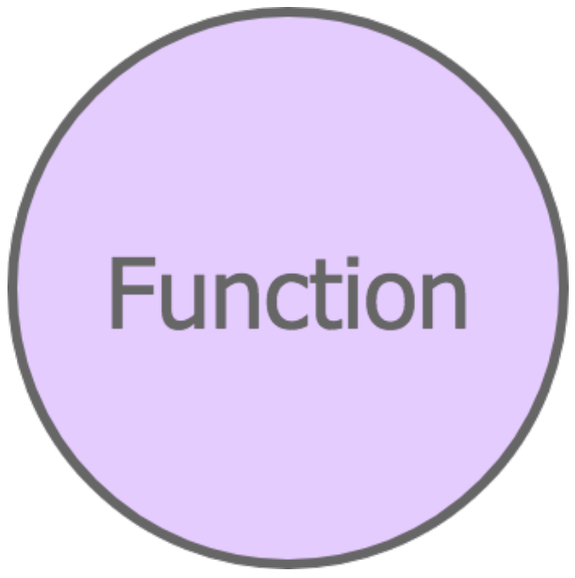


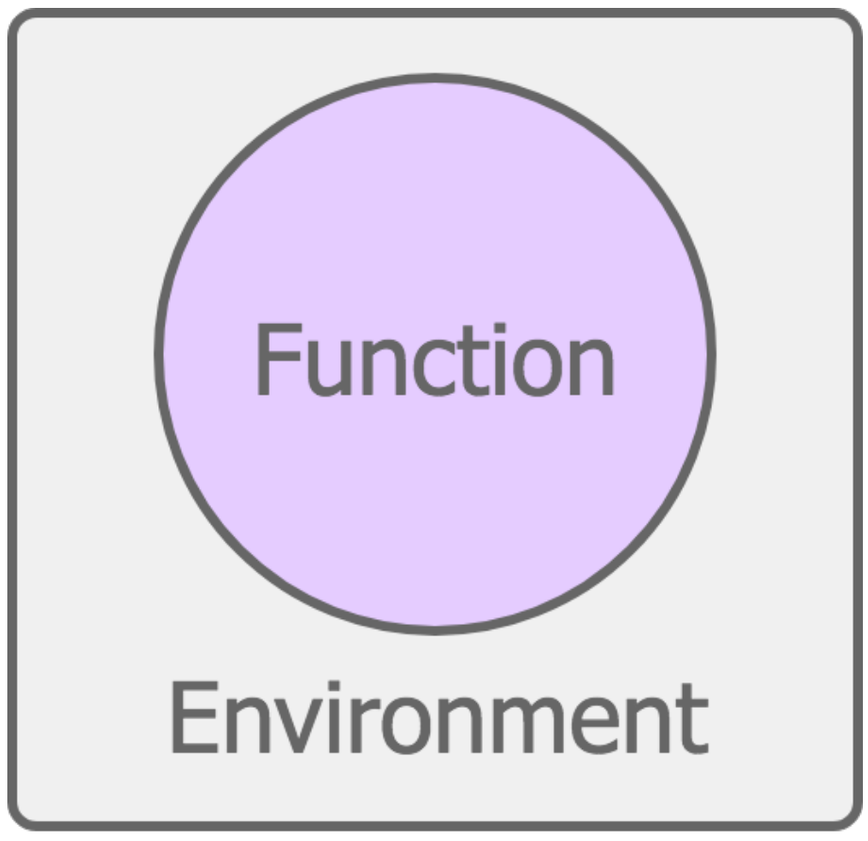
FISSION CONCEPTS

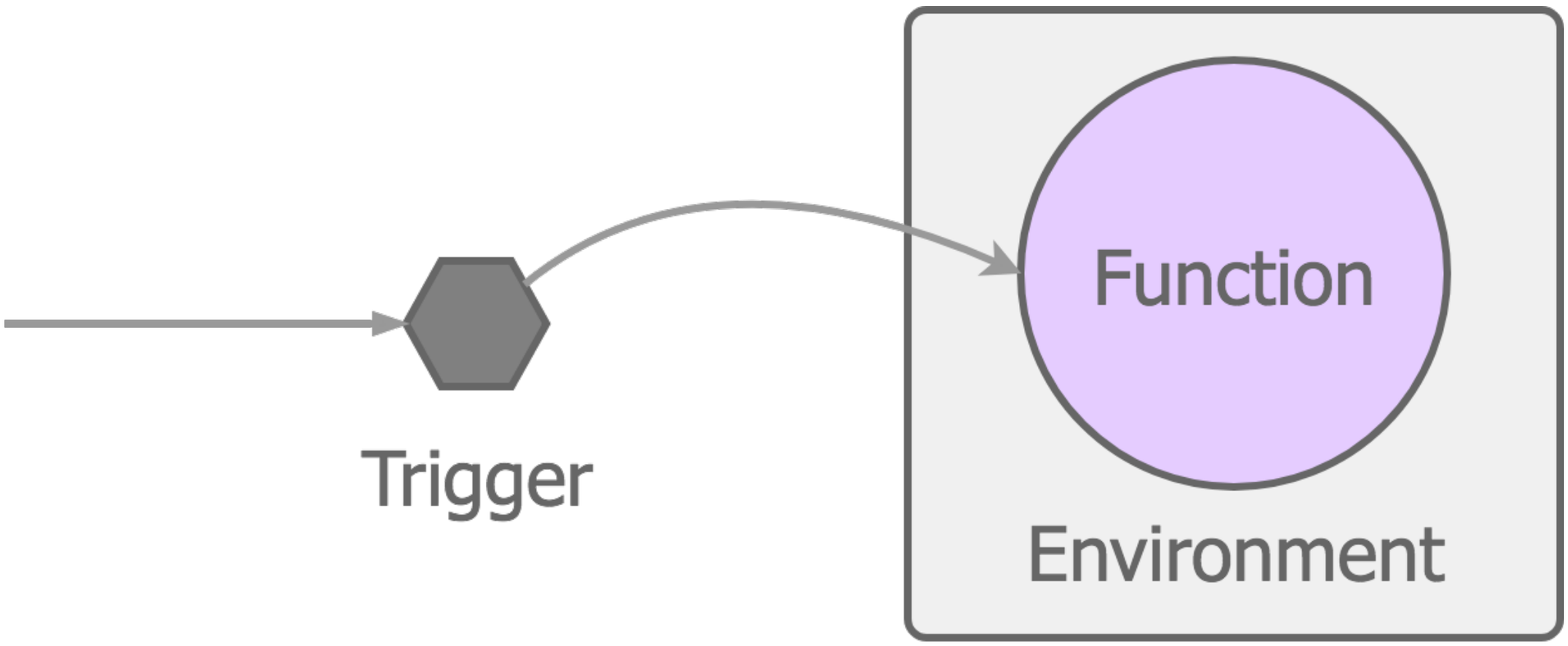
BASIC FISSION OBJECTS & RELATION

Three Core Objects

- Functions
  - Source code (3rd-party dependencies)
- Environments
  - Language runtime container
- Triggers
  - HTTP, Timer, KubeWatch trigger
  - Message queue trigger: NATS, Kafka


Function

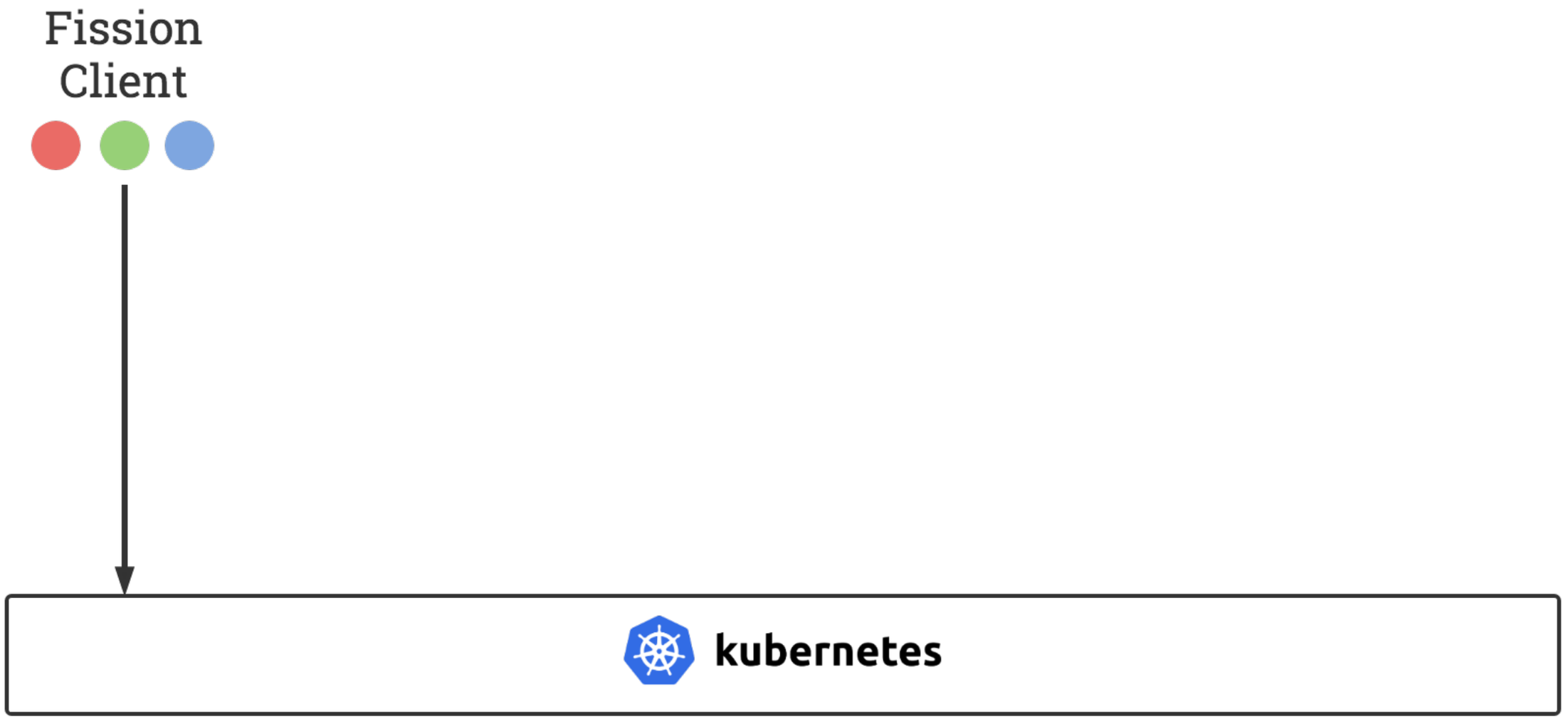

Function
Environment


Function
Environment
Trigger

FISSION CONCEPTS

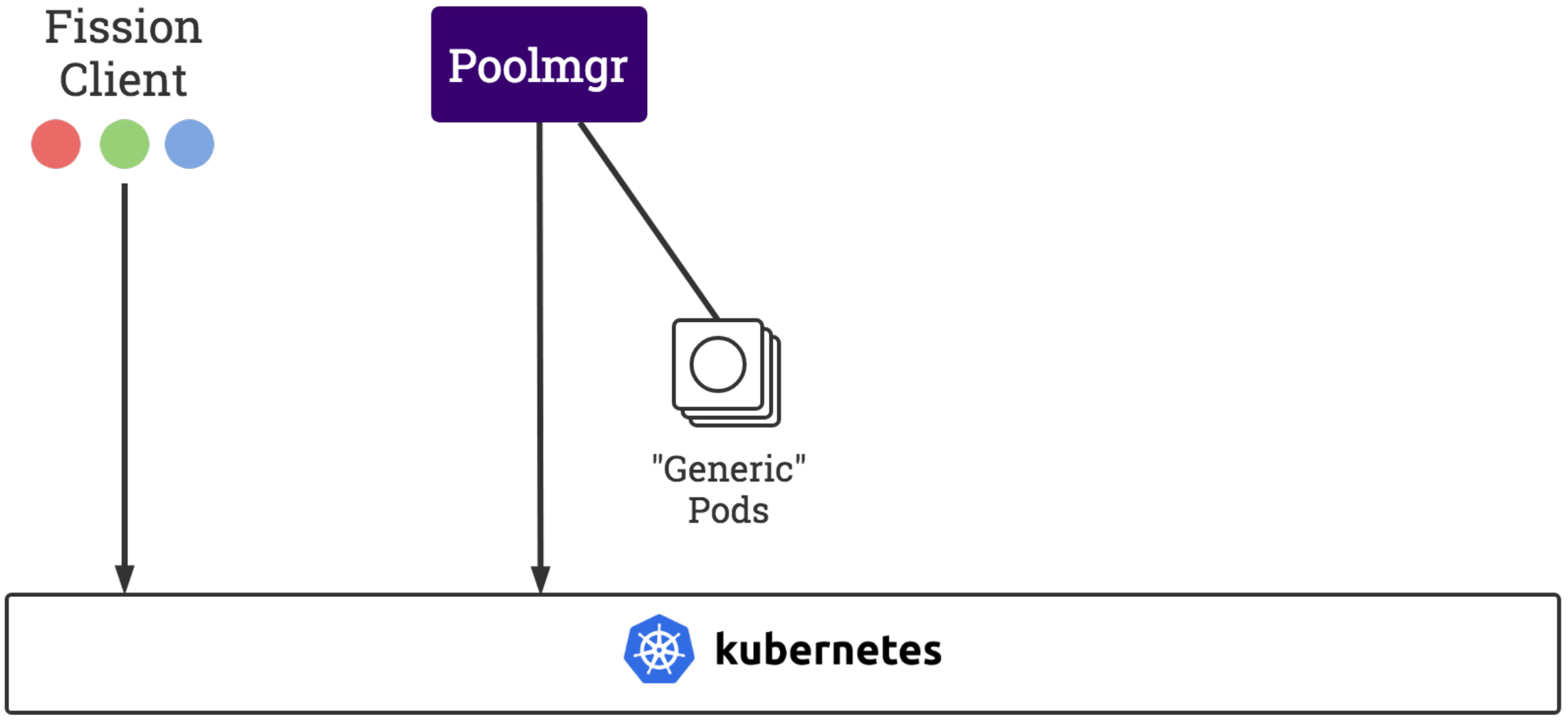
HOW FISSION WORKS WITH KUBERNETES?

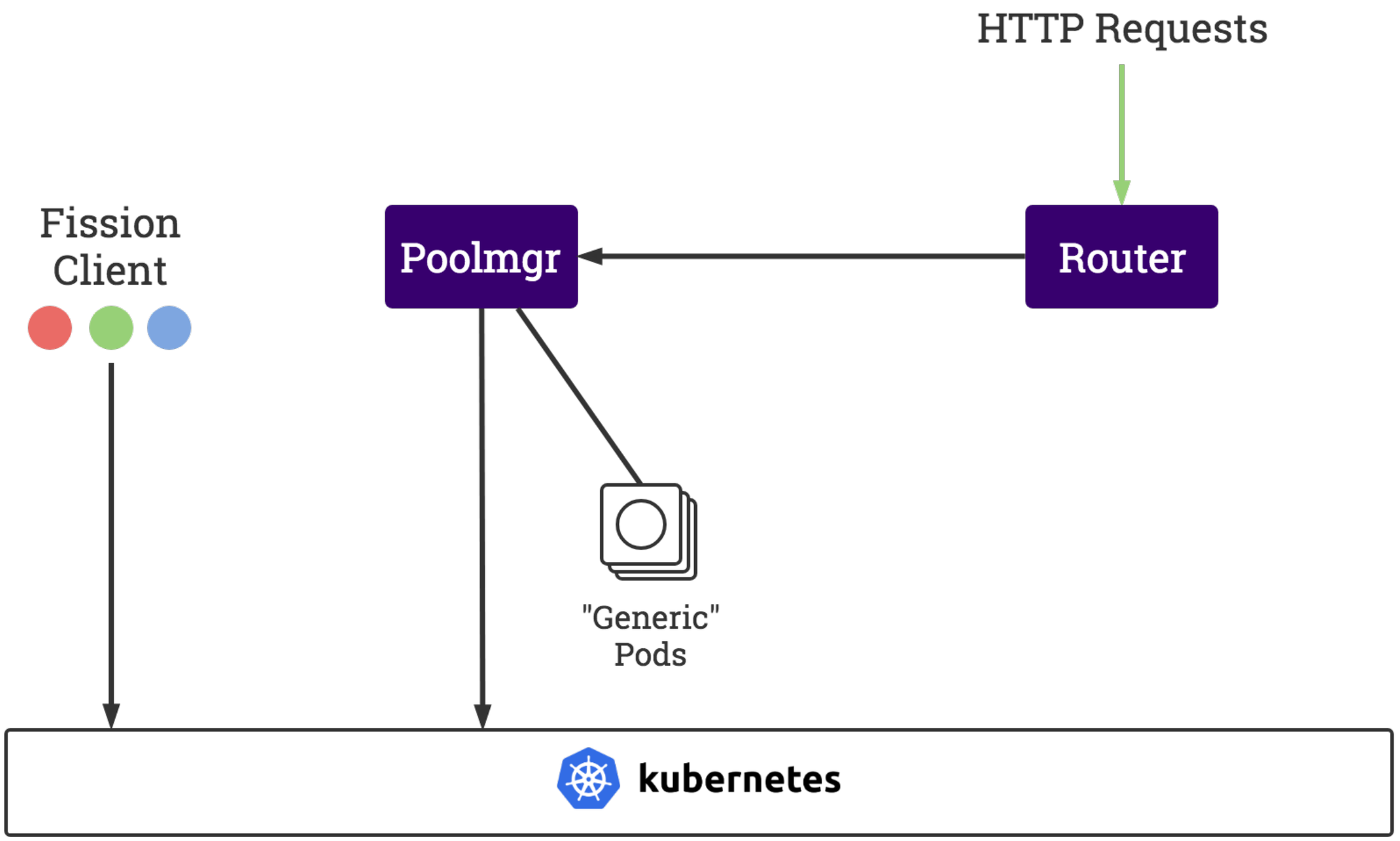
Fission
Client

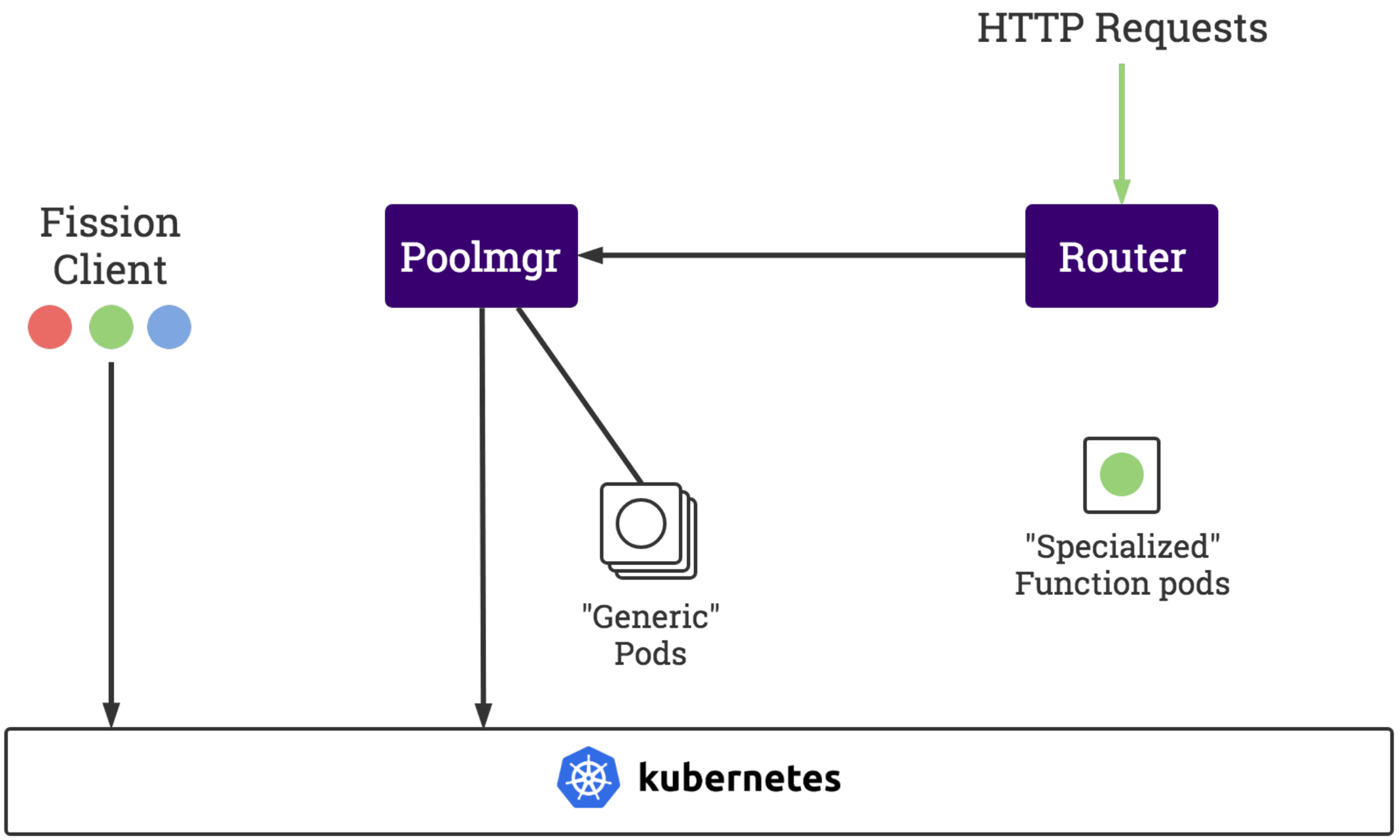

Fission
Client
kubernetes

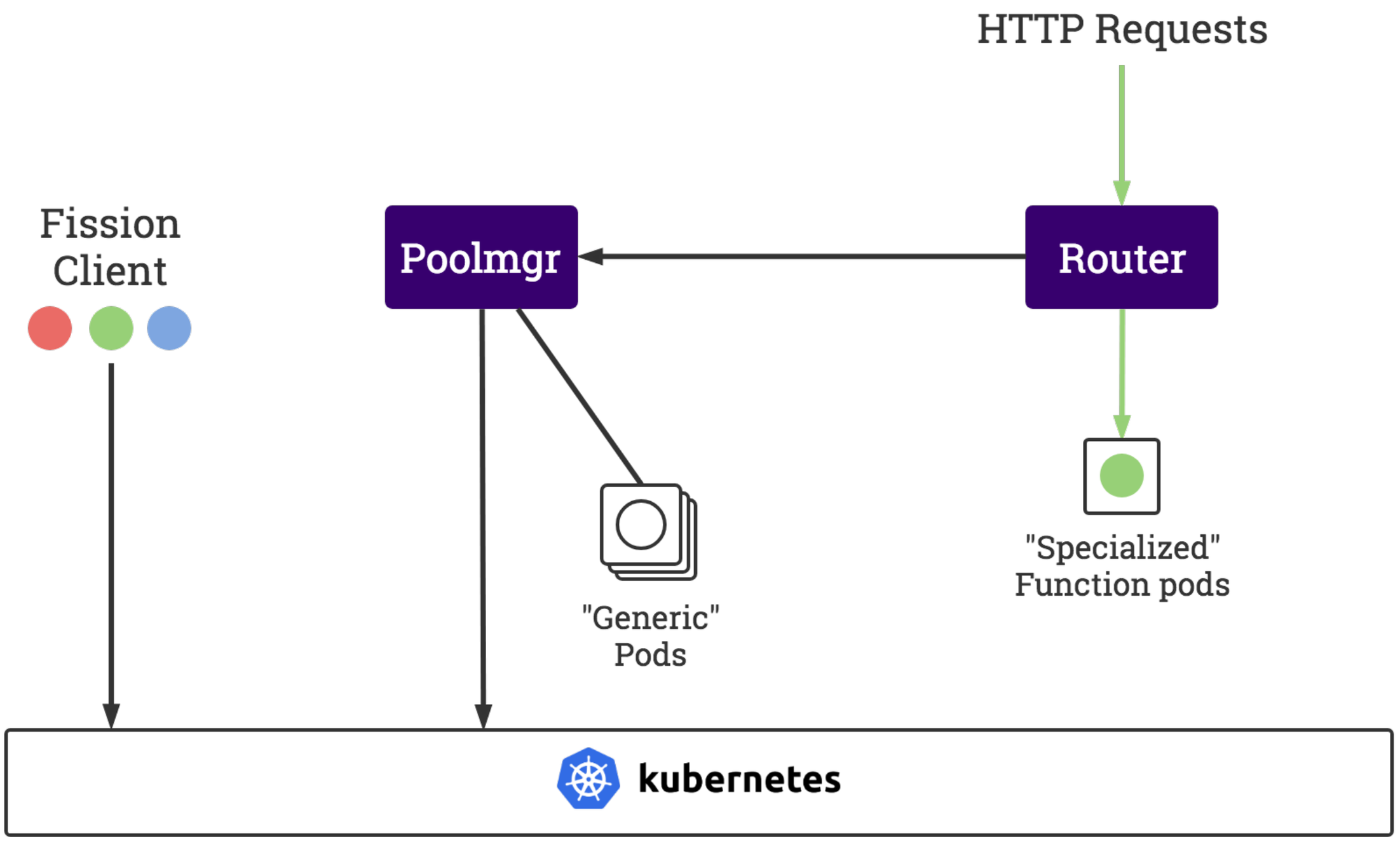
Fission
Client

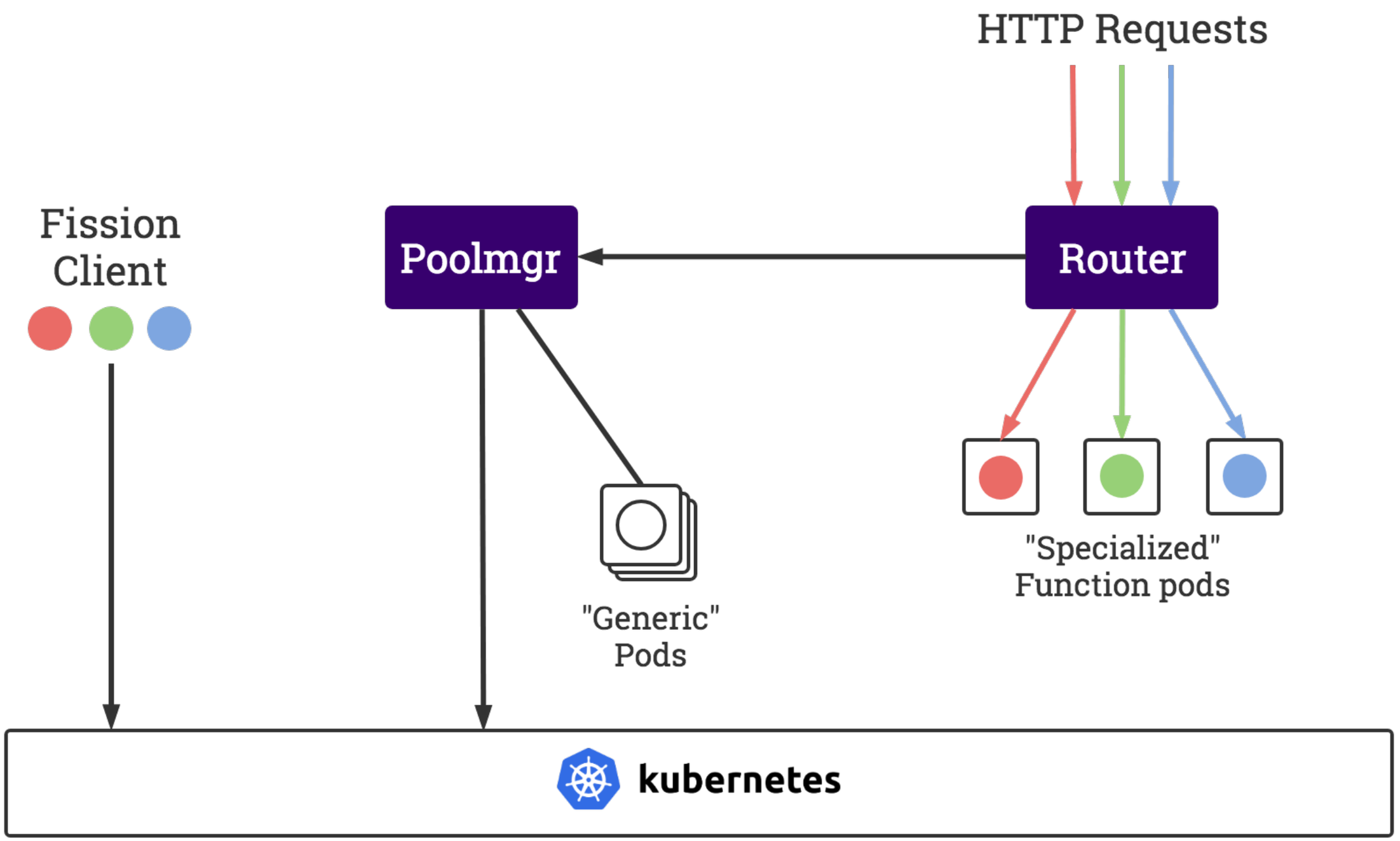
kubernetes

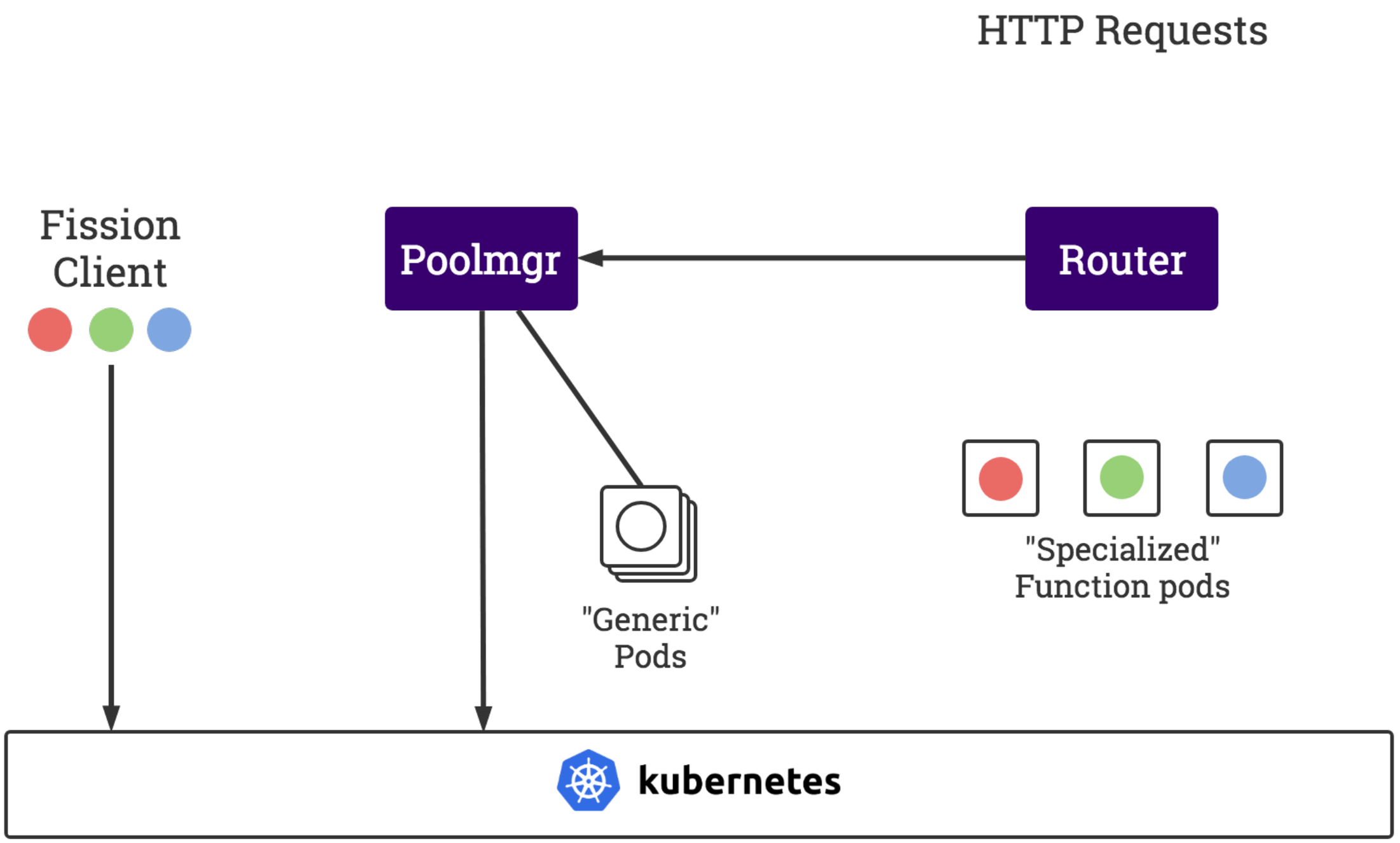

Fission
Client
Poolmgr
"Generic"
Pods
kubernetes

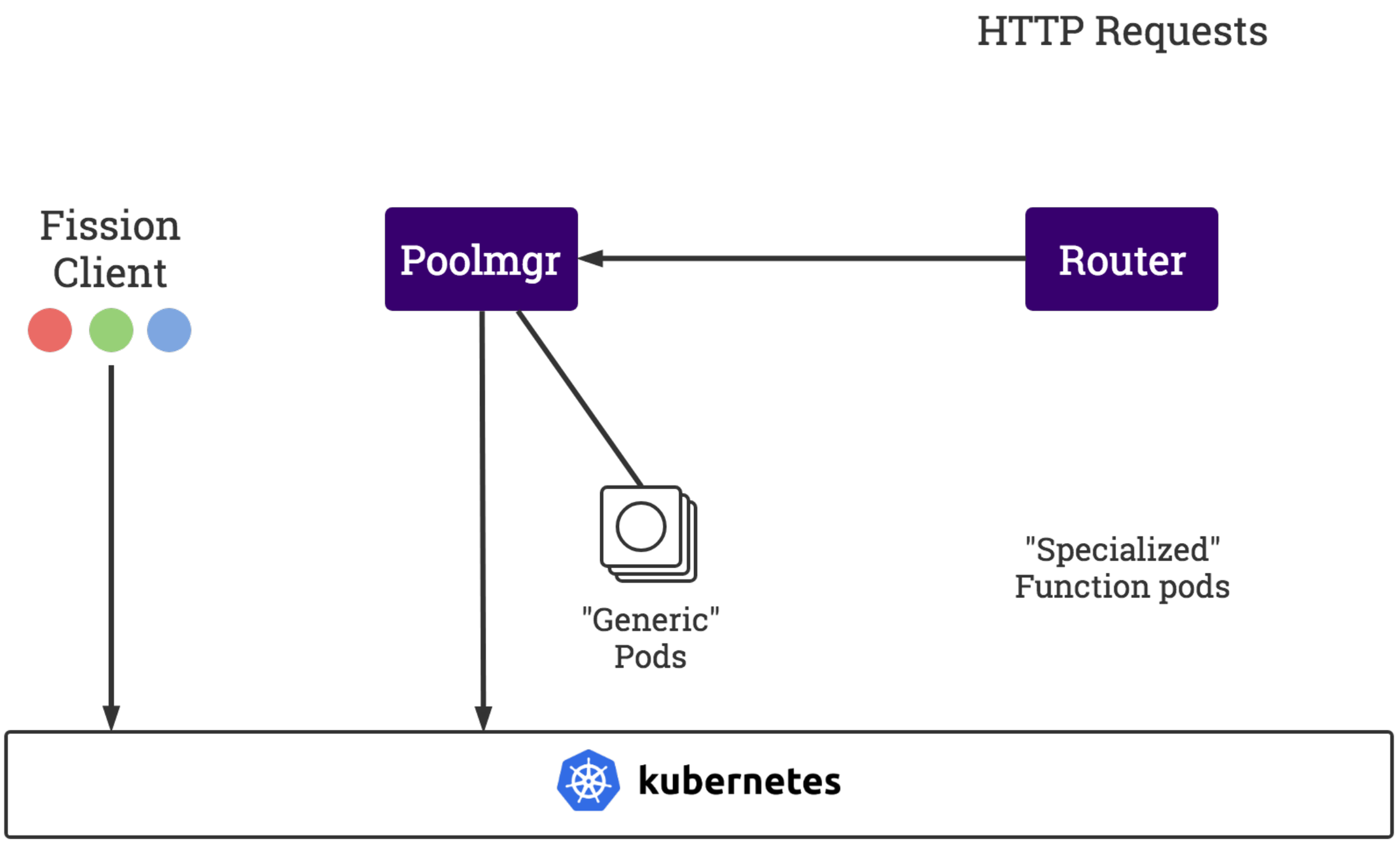

HTTP Requests
Fission
Client
Poolmgr
Router
"Generic"
Pods
kubernetes

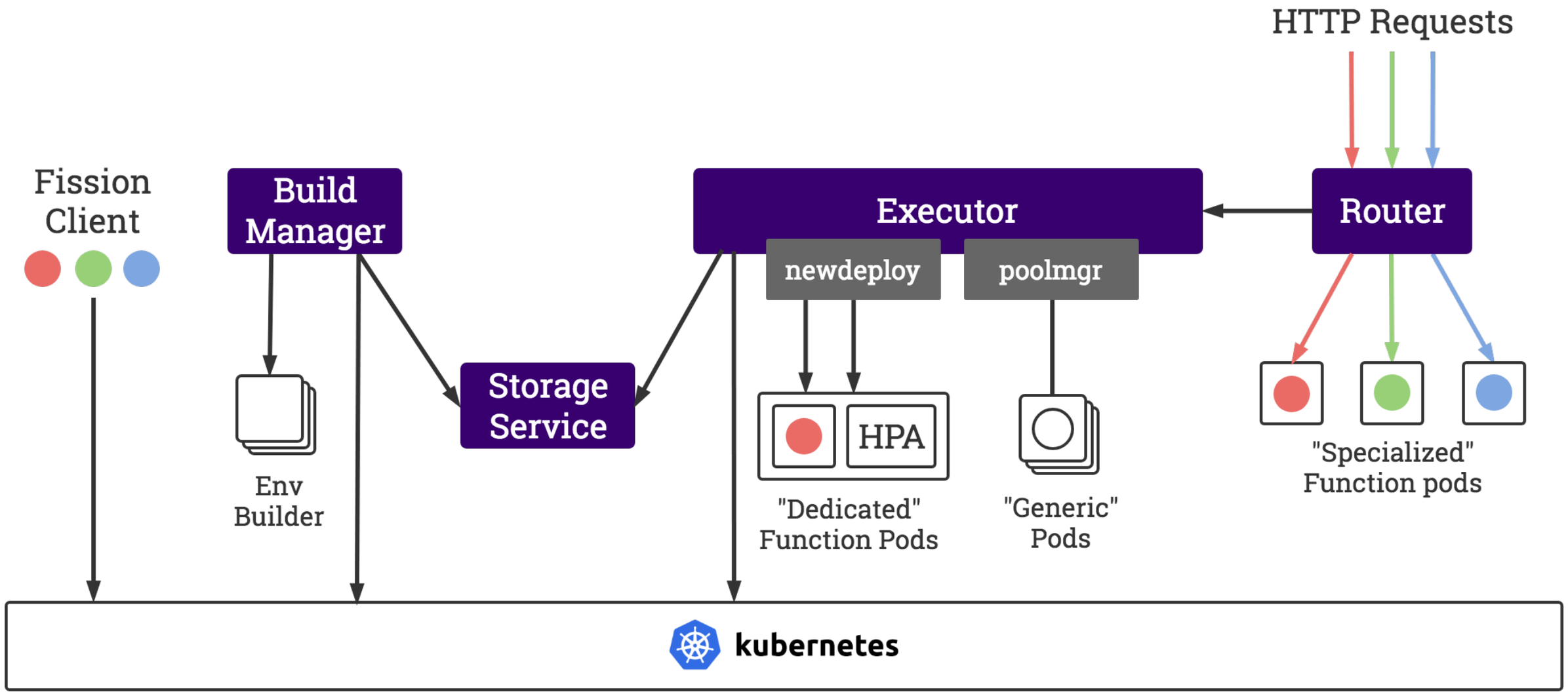

HTTP Requests
Fission
Client
Poolmgr
Router
"Generic"
Pods
"Specialized"
Function pods
kubernetes


HTTP Requests
Fission
Client
Poolmgr
Router
"Generic"
Pods
"Specialized"
Function pods
kubernetes


HTTP Requests
Fission
Client
Poolmgr
Router
"Generic"
Pods
"Specialized"
Function pods
kubernetes

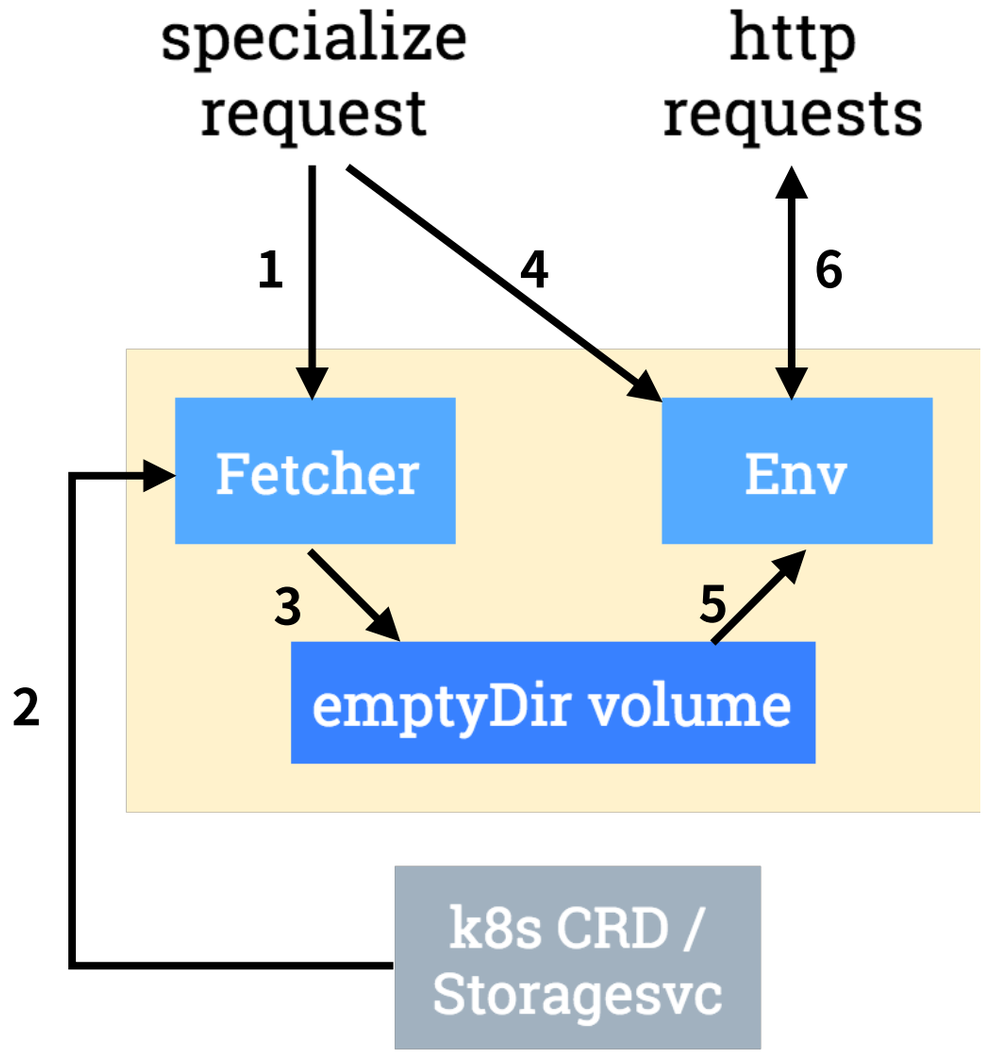

HTTP Requests
Fission
Client
Poolmgr
Router
"Generic"
Pods
"Specialized"
Function pods
kubernetes

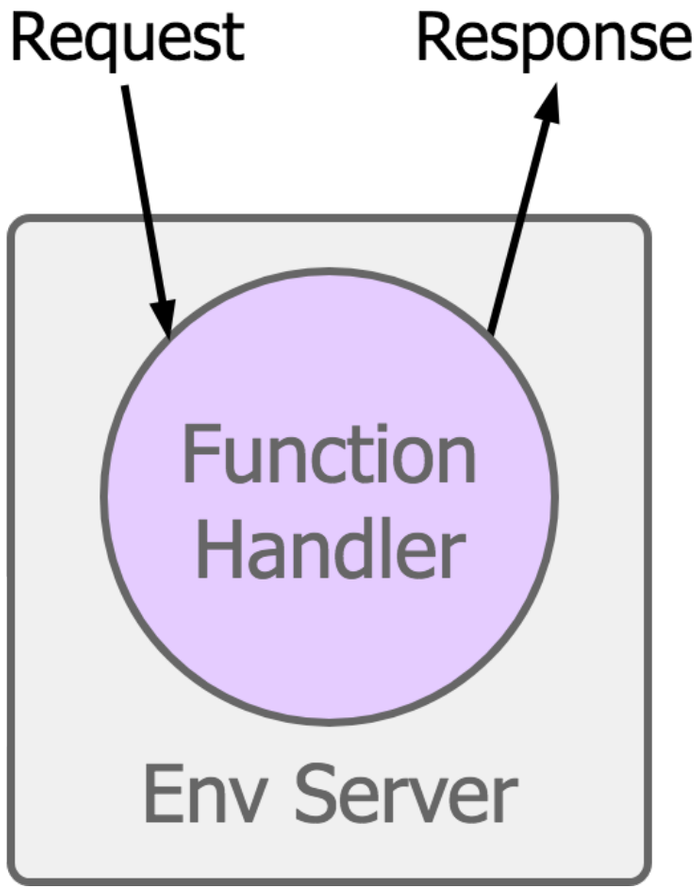

HTTP Requests
Fission
Client
Poolmgr
Router
"Generic"
Pods
"Specialized"
Function pods
kubernetes


HTTP Requests
Fission Client
Build Manager
Executor
Router
newdeploy
poolmgr
Storage Service
HPA
Env Builder
"Dedicated" Function Pods
"Generic" Pods
"Specialized" Function pods
kubernetes

Function Pod - Specializing Process

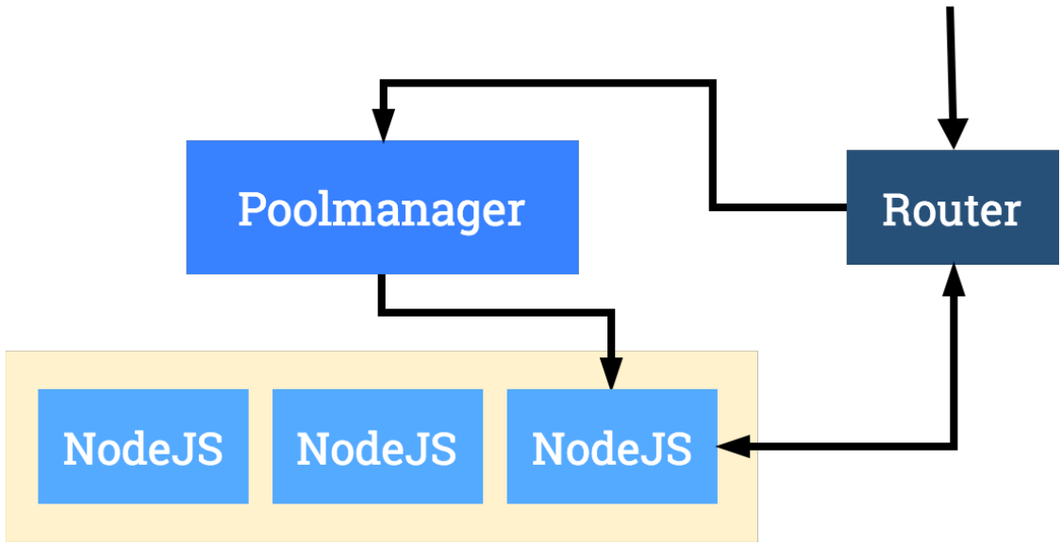
- Pod: A set of containers (container group)

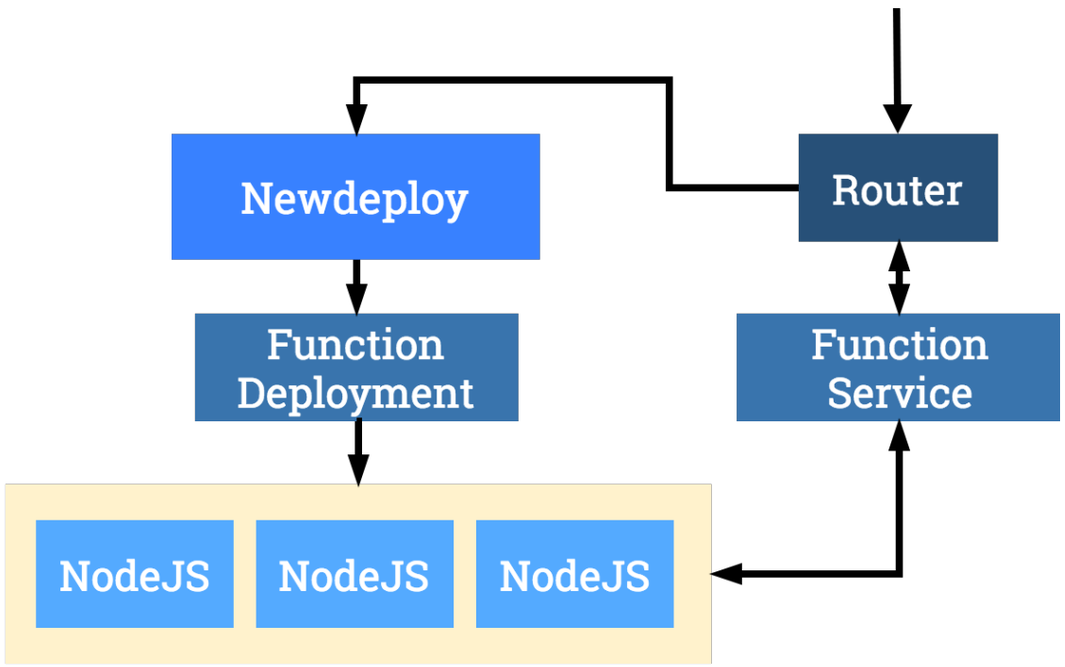




FISSION CONCEPTS

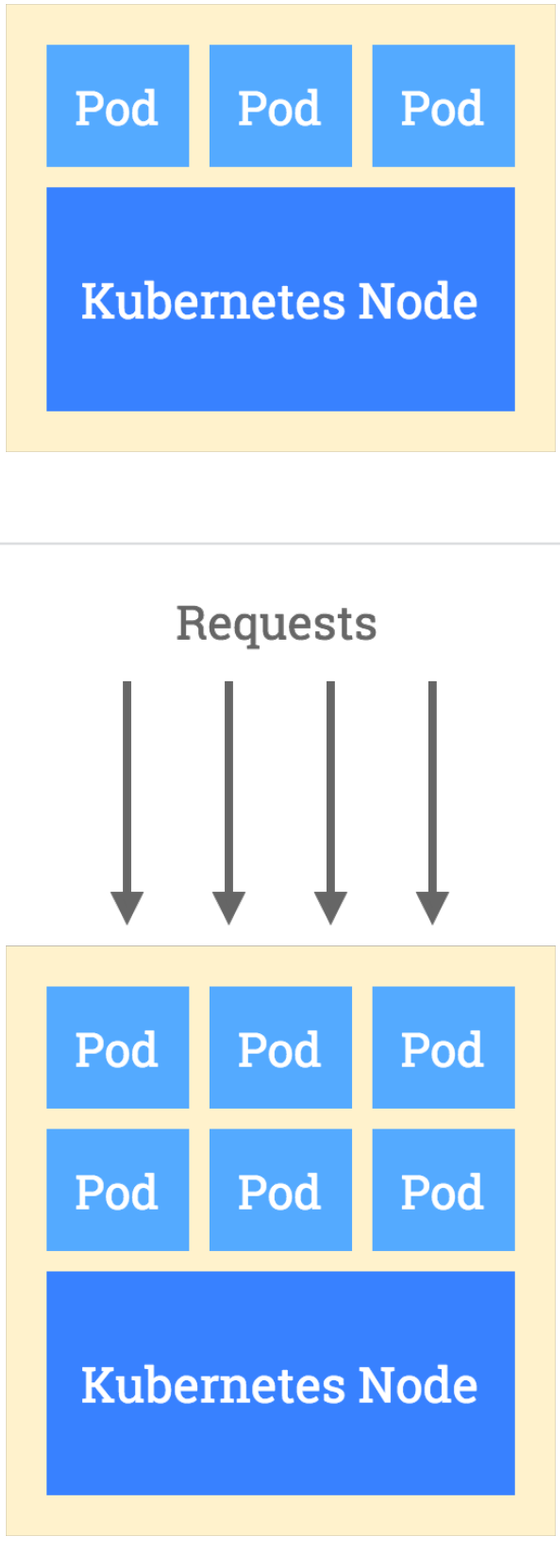
TUNABLE COST/PERFORMANCE TRADEOFFS





Execution Strategies

- Pool-based executor (Poolmanager)
  - Low-latency, small idle overhead
  - Pre-warmed pool of env containers
  - Functions loaded on-demand
- New-deployment executor (Newdeploy)
  - High latency, small idle overhead
  - Autoscaling: Min, Max instances
  - Min: Idle cost v.s Burst tolerance
  - Max: Throughput v.s Cost



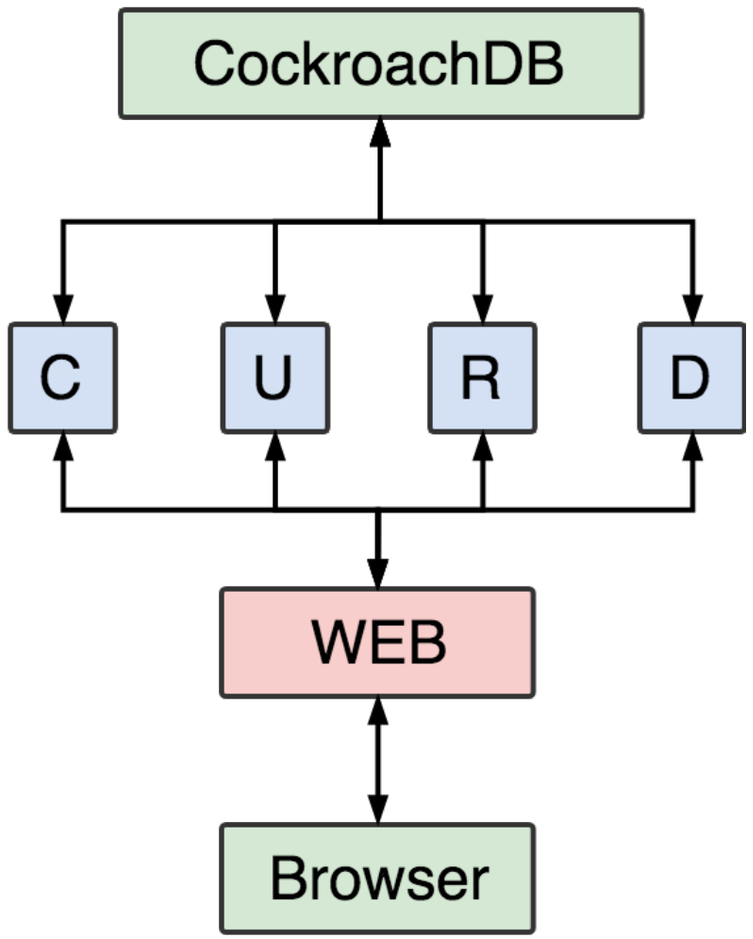
Cost Optimization

- Careful configuration of resources limits
- Autoscaling makes cluster more efficient
- Public Cloud
  - Pay for what you use
  - Reserved/Spot/Preemptible VMs
- On-premises
  - Resources should be proportional to actual demand

HANDS-ON TIME

GOAL TODAY

A serverless guestbook application
consists of 4 REST functions



TODO items

- Write a "hello world" function
- Add dependencies to function
  - Access database
  - Write "R" function
- Access URL parameters
  - Write "CUD" functions
- Deploy application with fission spec

HANDS-ON TIME

HELLO WORLD (HOLA)

Node

```
module.exports = async function(context) {
    return {
        status: 200,
        body: "hello, world!\n"
    };
}
```

- fission env create --name <env-name> --image fission/node-env
- fission fn create --name <fn-name> --code <file>
- fission fn test --name <fn-name>

Python

```python
def main():
    return "Hello, world!\n"
```

- fission env create --name <env-name> --image fission/python-env
- fission fn create --name <fn-name> --code <file>
- fission fn test --name <fn-name>

Go

```go
package main

import (
    "net/http"
)

func Handler(w http.ResponseWriter, r *http.Request) {
    msg := "Hello, world!\n"
    w.Write([]byte(msg))
}
```

- fission env create --name <env-name> --image fission/go-env \
  --builder fission/go-builder
- fission fn create --name <fn-name> --src <file>
- fission pkg info --name <pkg-name>
- fission fn test --name <fn-name>

HANDS-ON TIME

HTTP TRIGGER
ACCESS URL PARAMETERS AND QUERY STRINGS

HTTP trigger URL

```
$ fission httptrigger create --method GET \
    --url "/<username>/guestbook" --function <fn-name>
```

- curl http://$FISSION_ROUTER/tc/guestbook
- Create routes for each different URL variation
  - "/<username>/guestbook"
  - "/<username>/guestbook/"
  - "/<username>/guestbook/{MsgId}"
  - "/<username>/guestbook/{MsgId}/"
- Not support wildcard('*') path

Create route with URL parameters

```
1 http://example.com/guestbook/messages/{MsgId}
```

- Command:
  - fission httptrigger create --name <trigger name> --url <URL>
- Support regular expression:
  - "/guestbook/messages/{MsgId}"
  - "/bank/{html:[a-zA-Z0-9\.\/]+}"
  - "/articles/{category}/{id:[0-9]+}"

Create route with URL parameters

```
1 http://example.com/guestbook/messages/{MsgId}
2 X-Fission-Params-Msgid: <val>
3
4 http://example.com/guestbook/{CaTeGoRy}/{MsgId}
5 X-Fission-Params-Category: <val>
6 X-Fission-Params-Msgid: <val>
```

- The router puts value of URL params at HTTP header
- URL params' name will be converted to canonical MIME format with prefix "X-Fission-Params-"

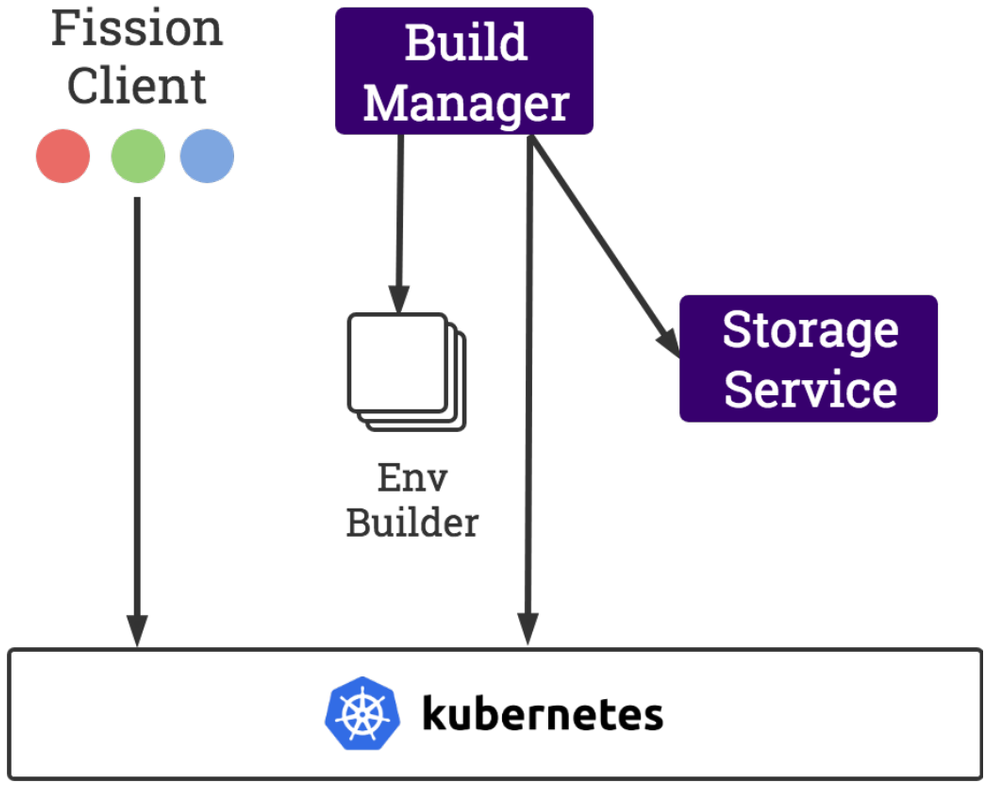
Access query strings

```
1 http://example.com/guestbook/messages/{MsgId}?user=foobar
```

- No need to specify query strings when creating http trigger.
- The router leave query string intact, you can access query string as usual in function.

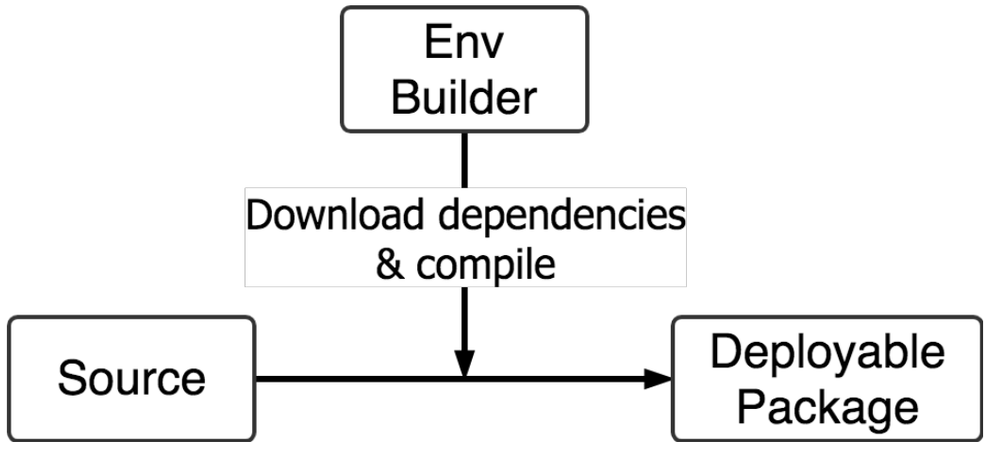
HANDS-ON TIME

ADD DEPENDENCIES



From source to deploy pkg

- Add dependencies management files
- Write function source code
- Zip them
- Create source package
- Wait for build result
- Create function using the package



Github Repo

- https://github.com/life1347/fission-workshop

Node

- READE: fission-workshop/node/dependency-example

```
{
  "name": "fission-nodejs-runtime",
  "engines": {
    "node": ">=7.6.0"
  },
  "dependencies": {
    "moment": "*"
  }
}
```

```
const momentpackage = require('moment')

module.exports = async function(context) {

    return {
        status: 200,
        body: momentpackage().format()
    };
}
```

Python

- READE: fission-workshop/python/dependency-example

```
1 pyyaml
```

```python
import sys
import yaml

document = """
  a: 1
  b:
    c: 3
    d: 4
"""

def main():
    return yaml.dump(yaml.load(document))
```

Go

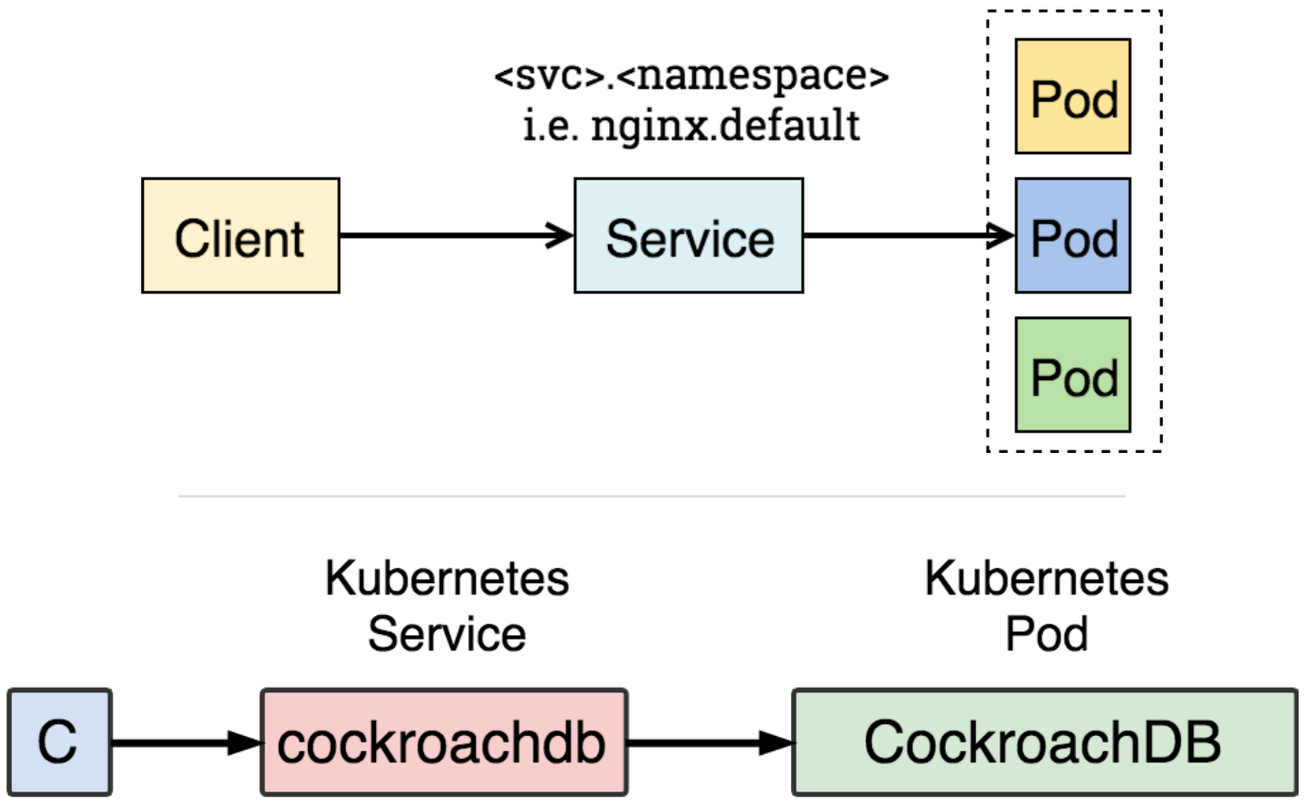
- READE: fission-workshop/go/dependency-example

```
$ glide init --non-interactive
$ glide get "github.com/golang/example/stringutil"
$ glide install -v
```

```
package main

import (
    "net/http"

    "github.com/golang/example/stringutil"
)

// Handler is the entry point for this fission function
func Handler(w http.ResponseWriter, r *http.Request) {
    msg := stringutil.Reverse(stringutil.Reverse("Vendor Example Test"))
    w.Write([]byte(msg))
}
```

Connection details

- README: https://github.com/life1347/fission-workshop



HANDS-ON TIME

WRITE REST CURD FUNCTIONS FOR GUESTBOOK

Q&A

Get Started with Fission!

Visit: **fission.io**

Github: **github.com/fission/fission**

Slack: **slack.fission.io**

Facebook: **Fission User Group Taiwan**