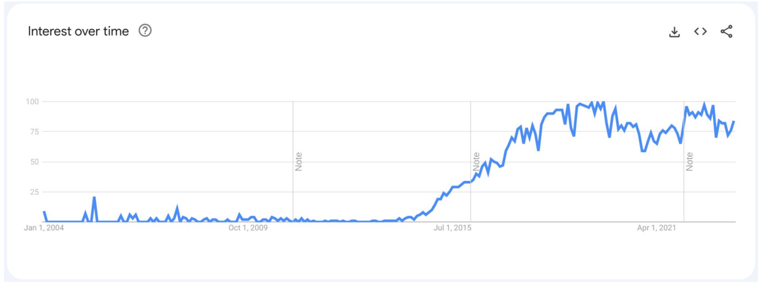
Towards Modern Development of Cloud Applications

Sanjay Ghemawat, Robert Grandl, Srdjan Petrovic, Michael Whittaker,
Parveen Patel, Ivan Posva, Amin Vahdat

Google

Microservices



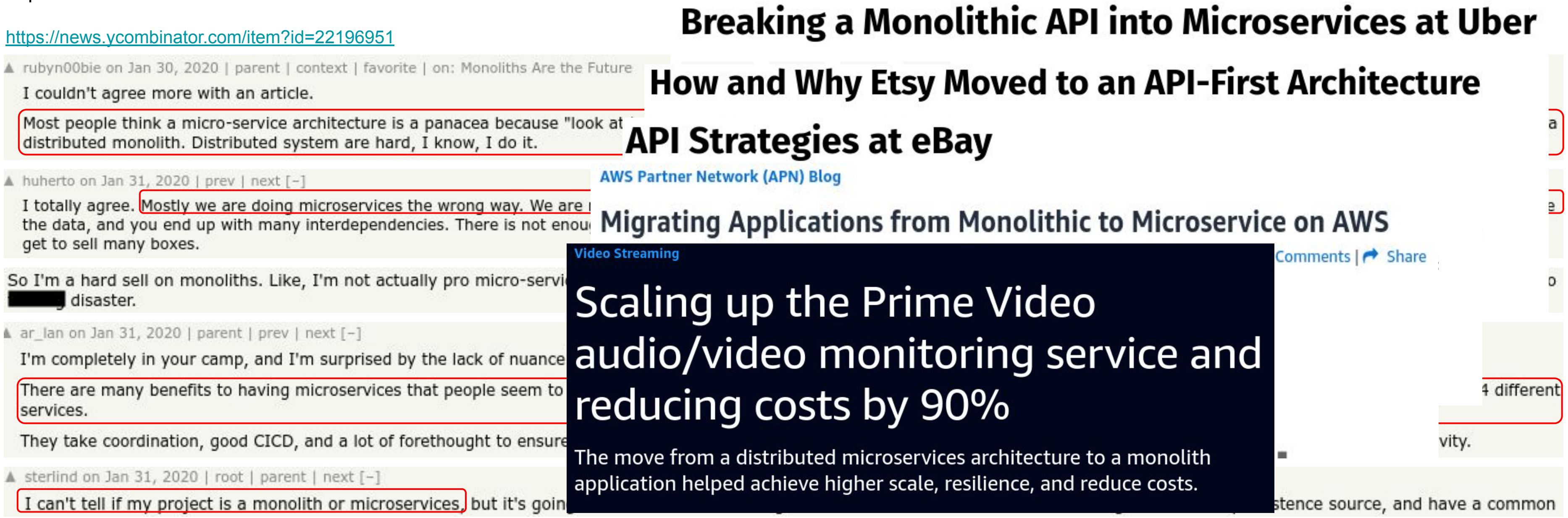
Plethora of technologies that help to grow the microservices ecosystem

docker kubernetes Istio argo OpenTelemetry

Prometheus akka RabbitMQ dapr mongo DB

Opinionated discussions

https://news.ycombinator.com/item?id=22196951



rubyn00bie on Jan 30, 2020 | parent | context | favorite | on: Monoliths Are the Future

I couldn't agree more with an article.

Most people think a micro-service architecture is a panacea because "look at
distributed monolith. Distributed system are hard, I know, I do it.

huherto on Jan 31, 2020 | prev | next [–]

I totally agree. Mostly we are doing microservices the wrong way. We are
the data, and you end up with many interdependencies. There is not enou
get to sell many boxes.

So I'm a hard sell on monoliths. Like, I'm not actually pro micro-servi
[redacted] disaster.

ar_lan on Jan 31, 2020 | parent | prev | next [–]

I'm completely in your camp, and I'm surprised by the lack of nuance

There are many benefits to having microservices that people seem to
services.

They take coordination, good CICD, and a lot of forethought to ensure

sterlind on Jan 31, 2020 | root | parent | next [–]

I can't tell if my project is a monolith or microservices, but it's goin

Breaking a Monolithic API into Microservices at Uber

How and Why Etsy Moved to an API-First Architecture

API Strategies at eBay

AWS Partner Network (APN) Blog

Migrating Applications from Monolithic to Microservice on AWS

Comments | Share

Video Streaming

Scaling up the Prime Video audio/video monitoring service and reducing costs by 90%

The move from a distributed microservices architecture to a monolith application helped achieve higher scale, resilience, and reduce costs.

a

e

o

4 different

vity.

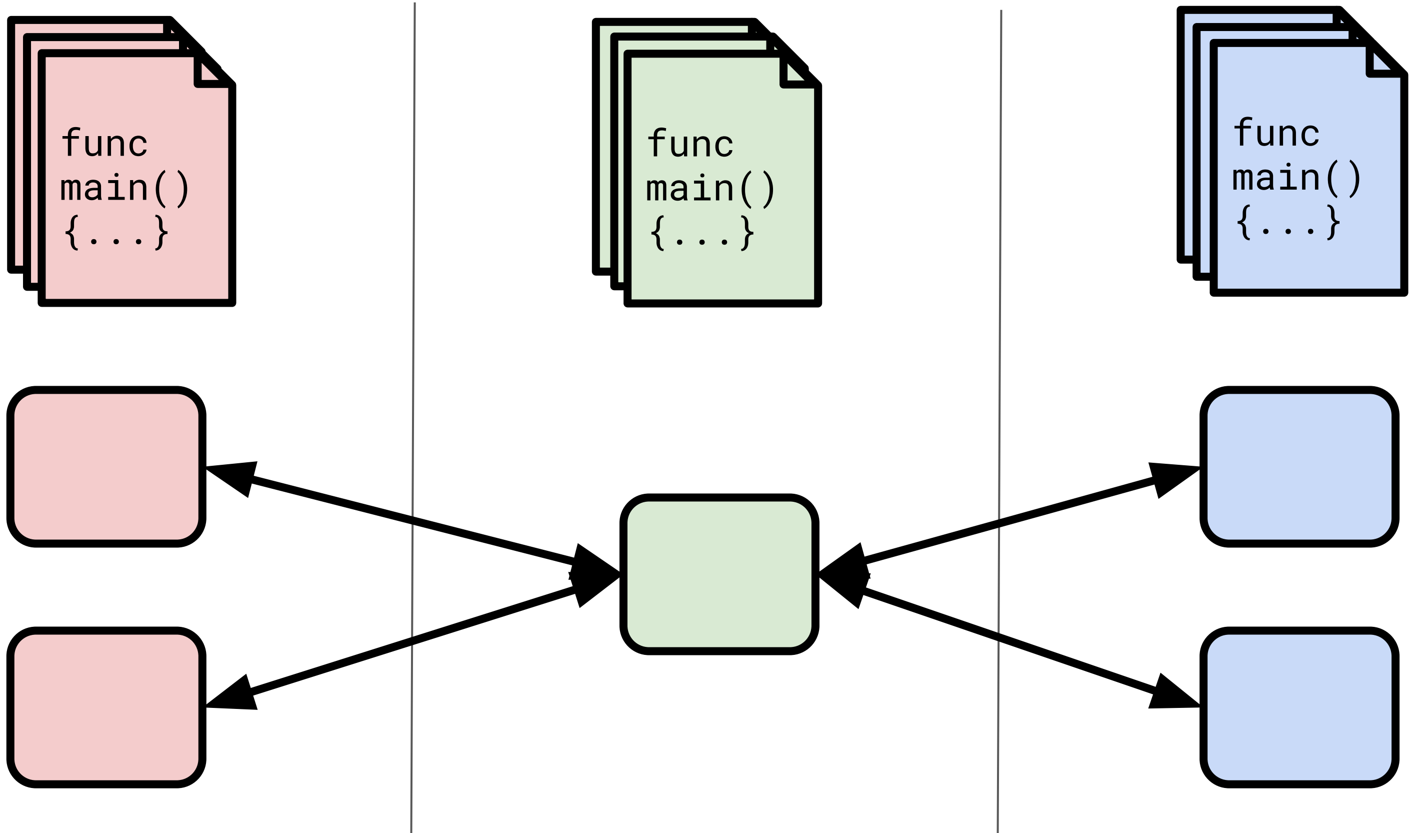
stence source, and have a common

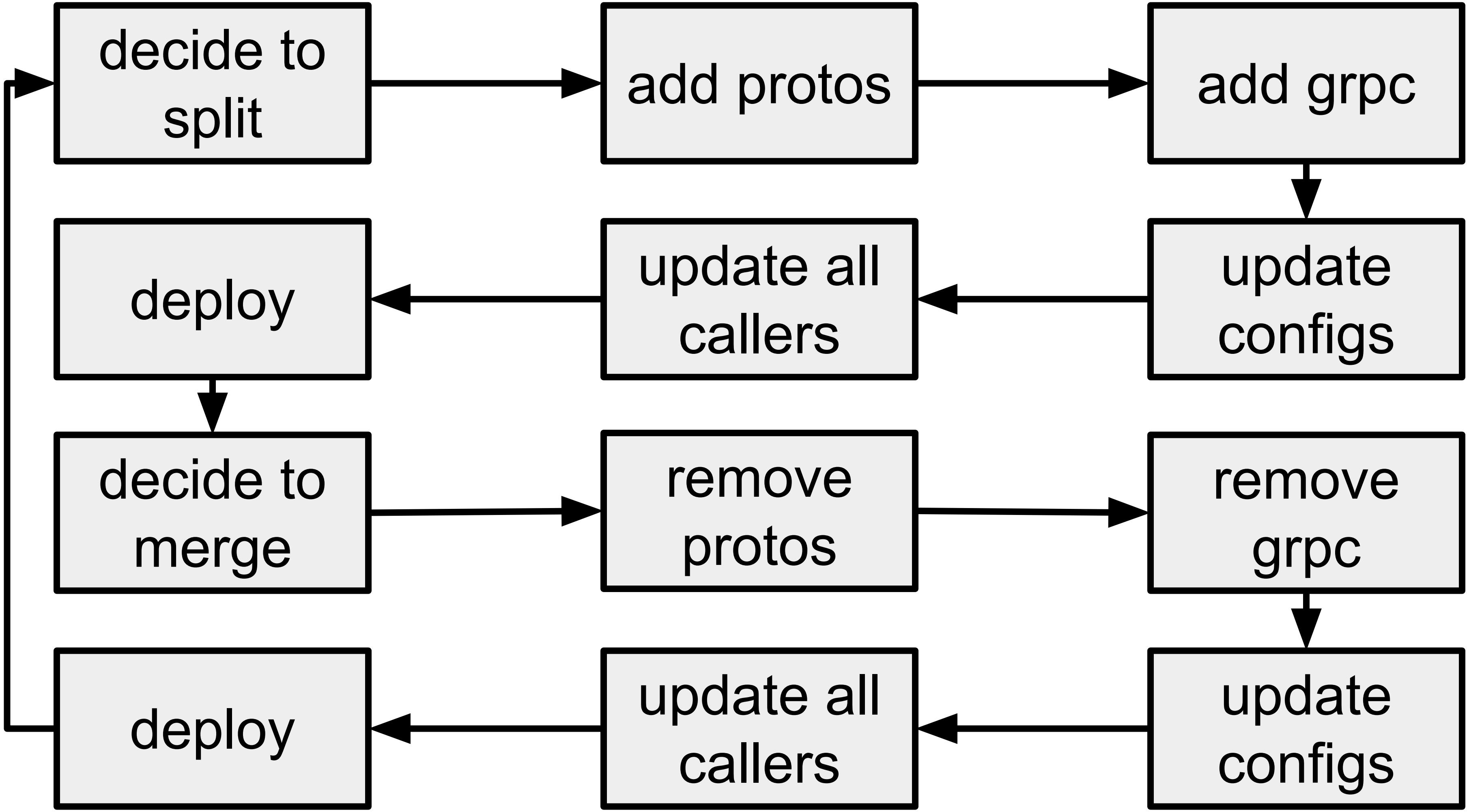
Microservices Pros and Cons

+ Performance 😀
+ Abstraction 😀
+ Fault Tolerance 😀

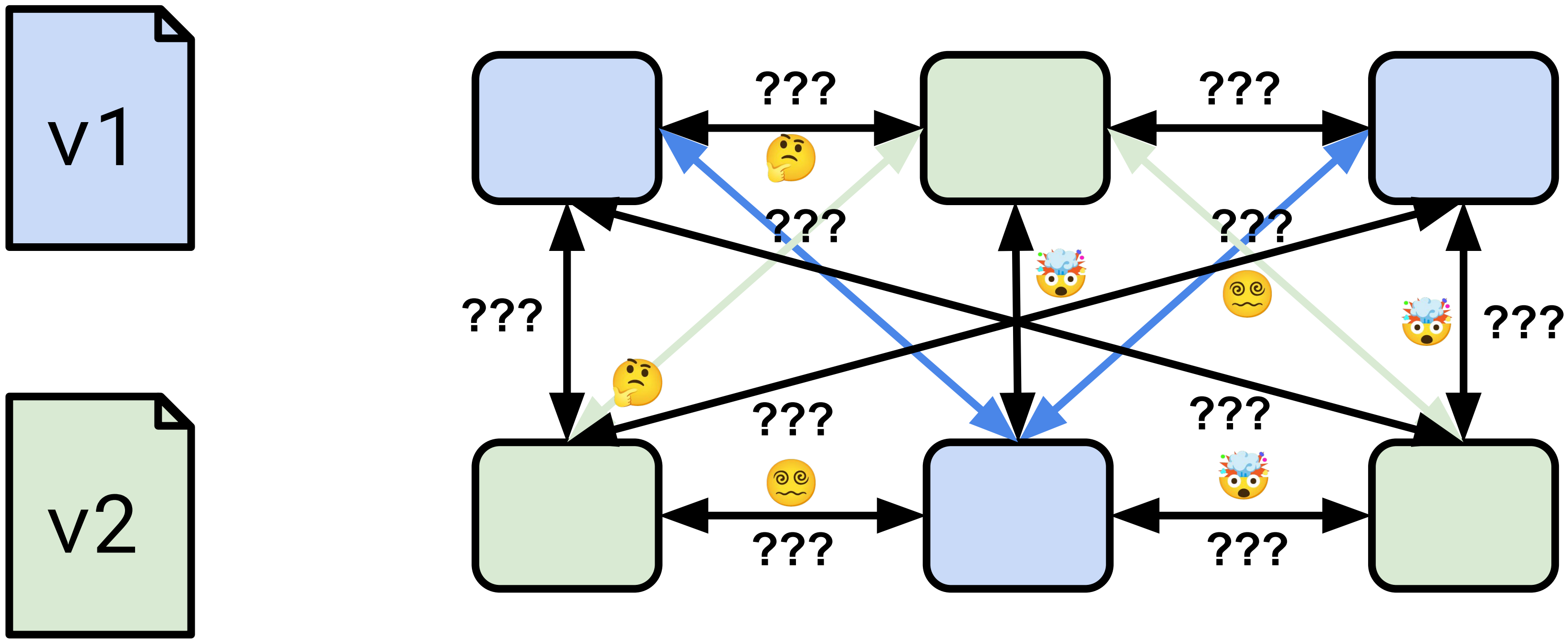
- Performance 😔
- Abstraction 😔
- Fault Tolerance 😔

1 Coupling of Logical and Physical Boundaries




decide to split
add protos
add grpc
update configs
update all callers
deploy
decide to merge
remove protos
remove grpc
update configs
update all callers
deploy

2 Versioning Woes



“About two thirds of update failures are caused by **interaction between two software versions** that hold incompatible data syntax or semantics assumption.”

Understanding and Detecting Software Upgrade Failures in Distributed Systems [SOSP’21]

OUR PROPOSAL

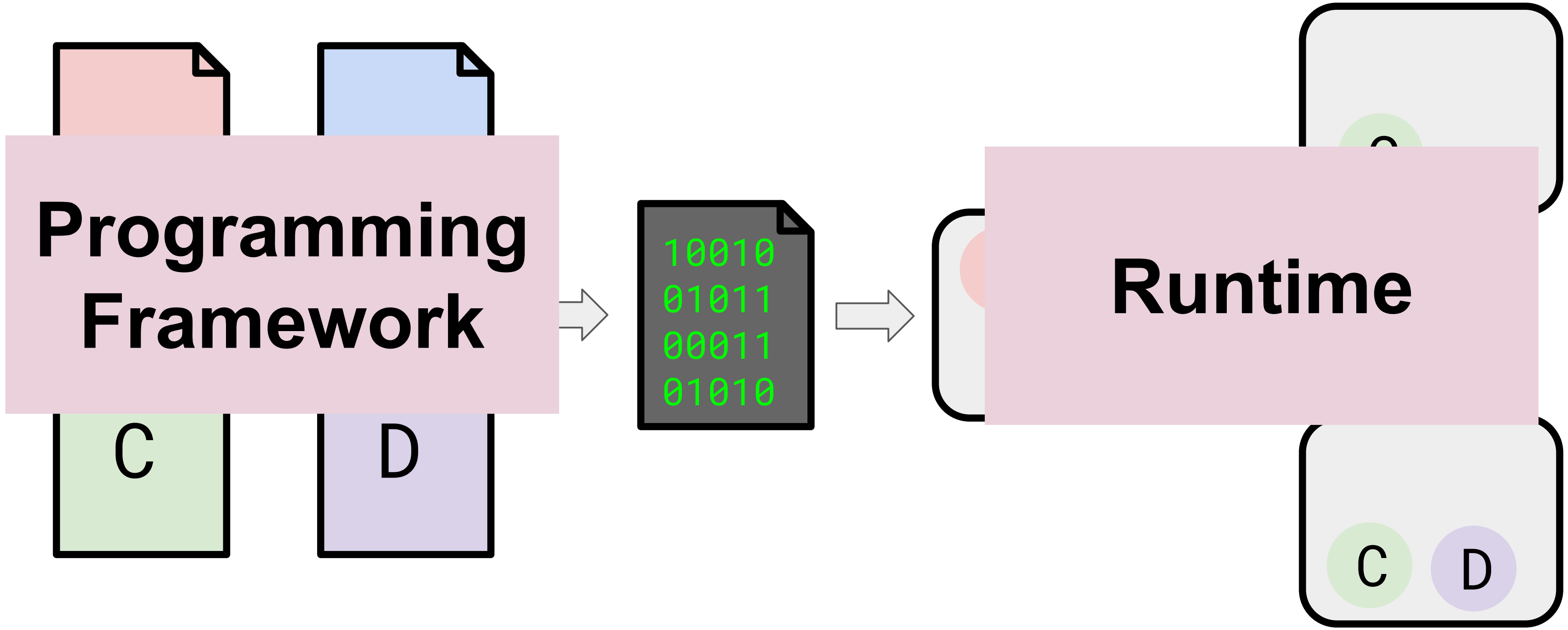
1 Decoupling of Logical and Physical Boundaries

2 Isolated Rollouts

1 Decoupling of Logical and Physical Boundaries

Write as a monolith.

Deploy as a set of microservices.



PROGRAMMING FRAMEWORK

Programming Framework

Components are...

- the key abstraction.
- like actors.
- long-lived.
- possibly replicated.
- soft-state.
- written using native language constructs.

Programming Framework

```go
// Component interface.
type Reverser interface {
    Reverse(string) string
}
```

Programming Framework

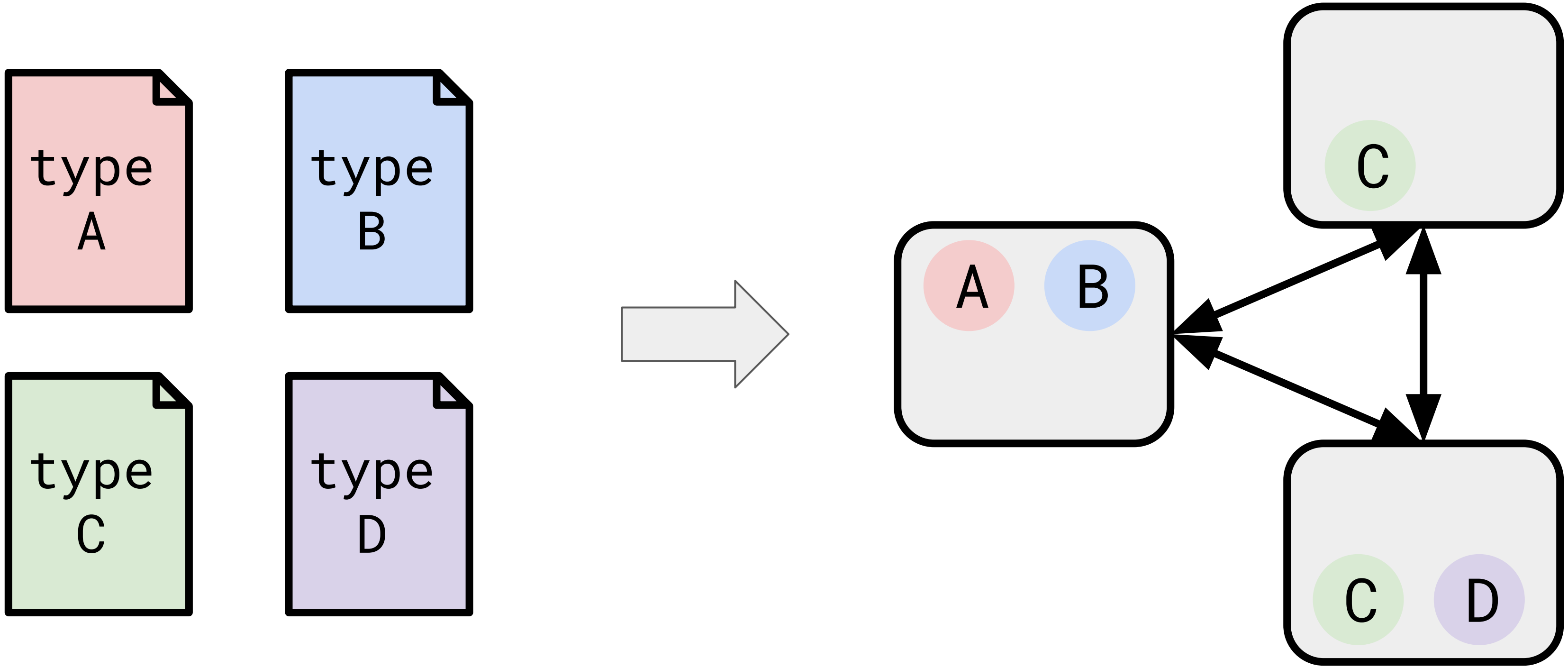
```go
// Component implementation.
type reverser struct {
    weaver.Implements[Reverser]
}

func (r *reverser) Reverse(s string) string {
    runes := []rune(s)
    n := len(runes)
    for i := 0; i < n/2; i++ {
        runes[i], runes[n-i-1] = runes[n-i-1], runes[i]
    }
    return string(runes)
}
```

Programming Framework

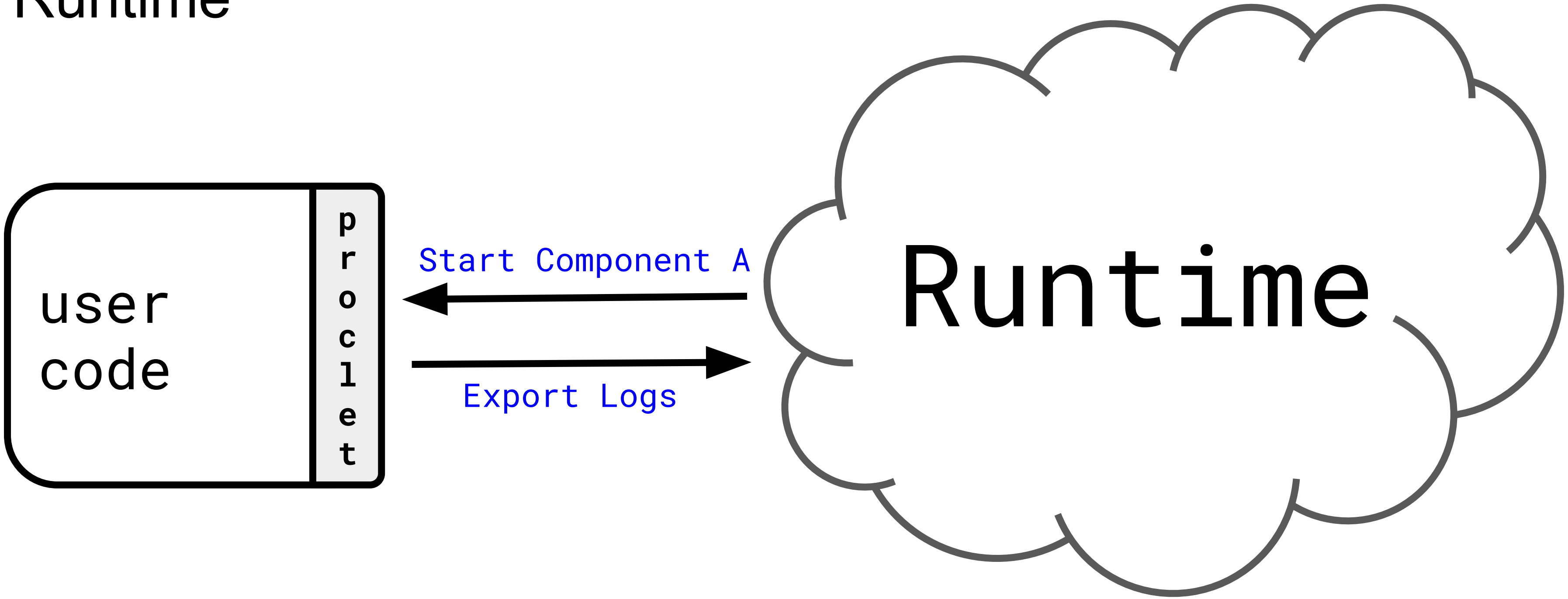
```go
func main() {
    app := weaver.Init()
    r := weaver.Get[Reverser](app)
    reversed := r.Reverse("!dlroW ,olleH")
    fmt.Println(reversed)
}
```

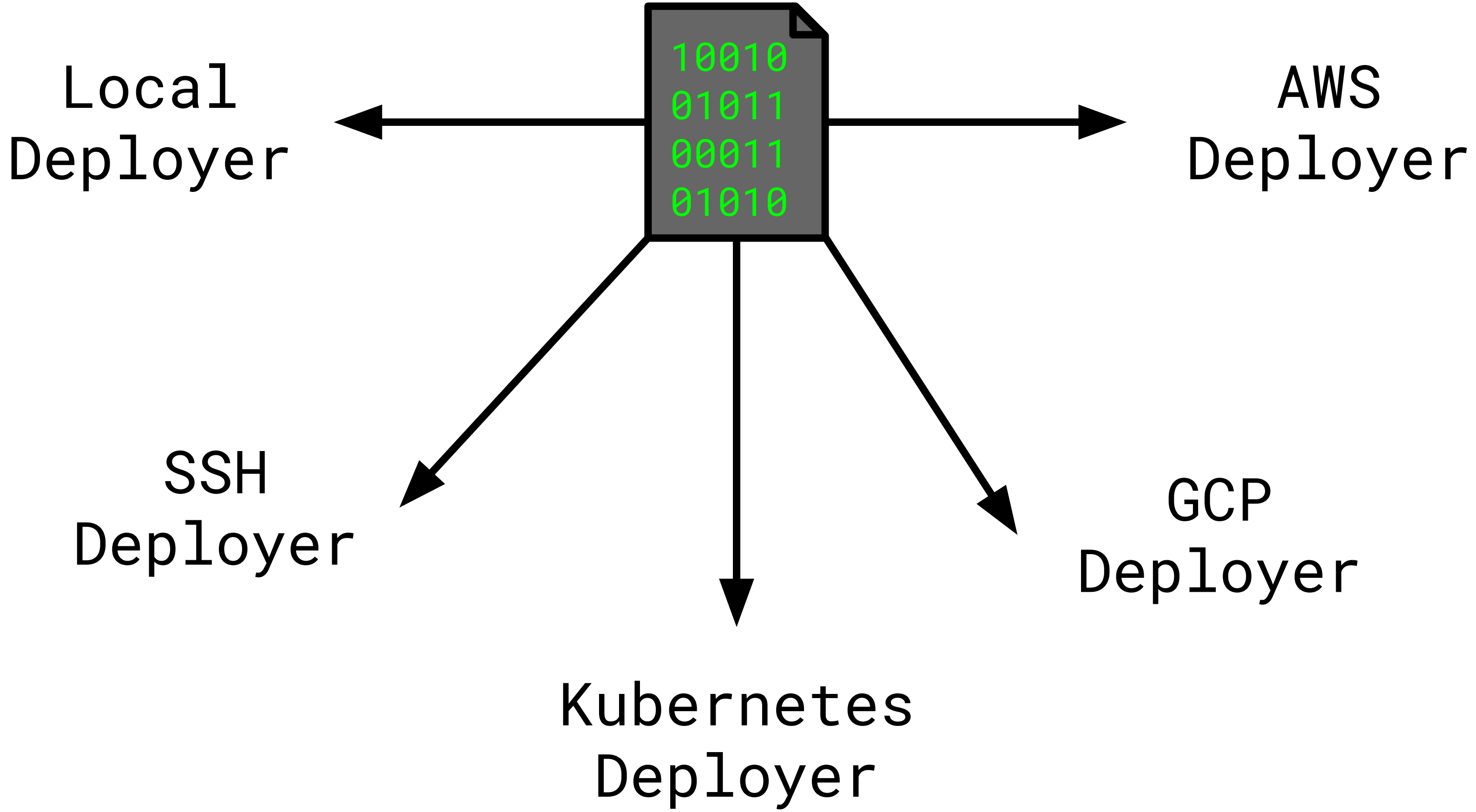
RUNTIME

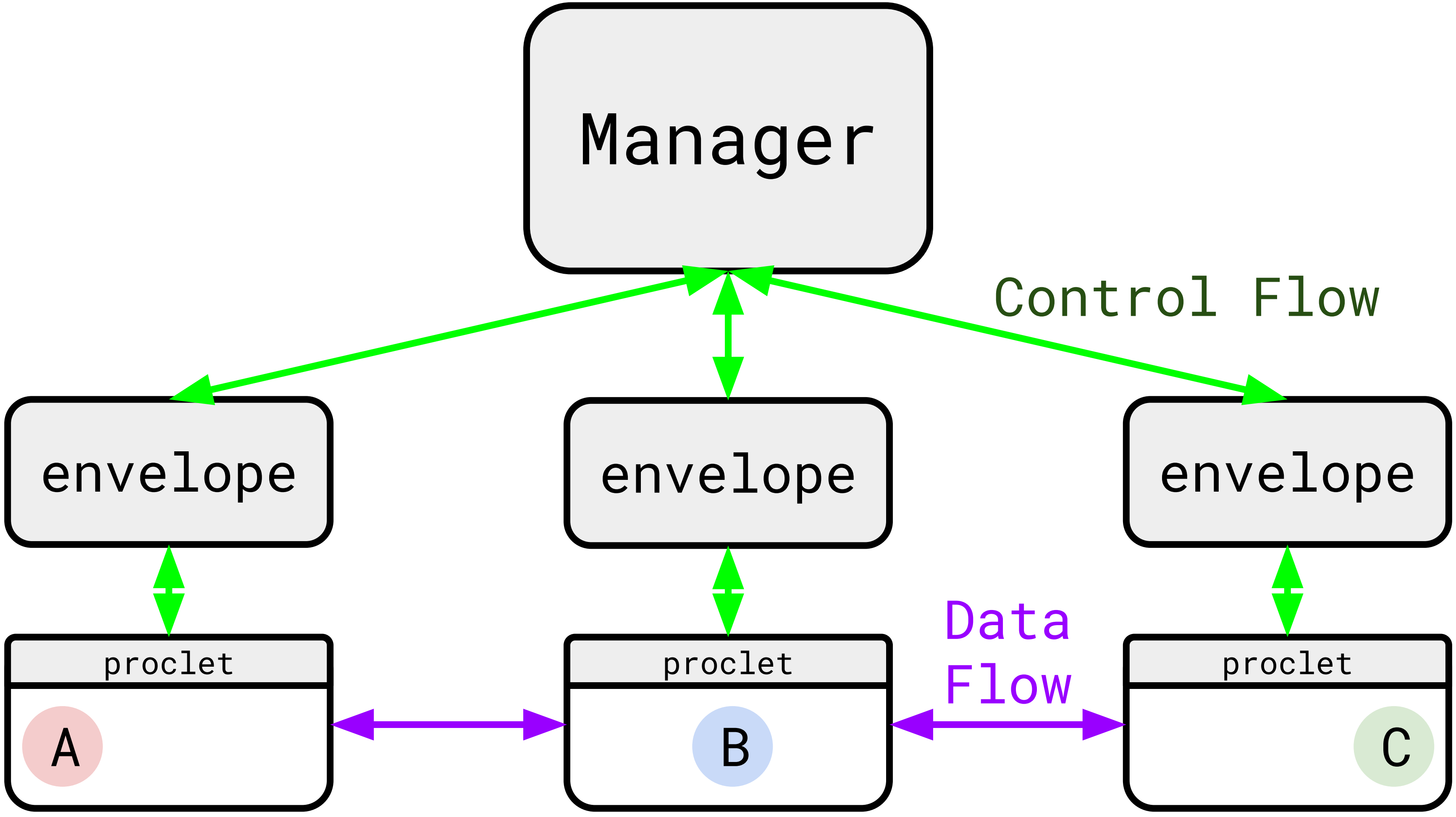
Runtime



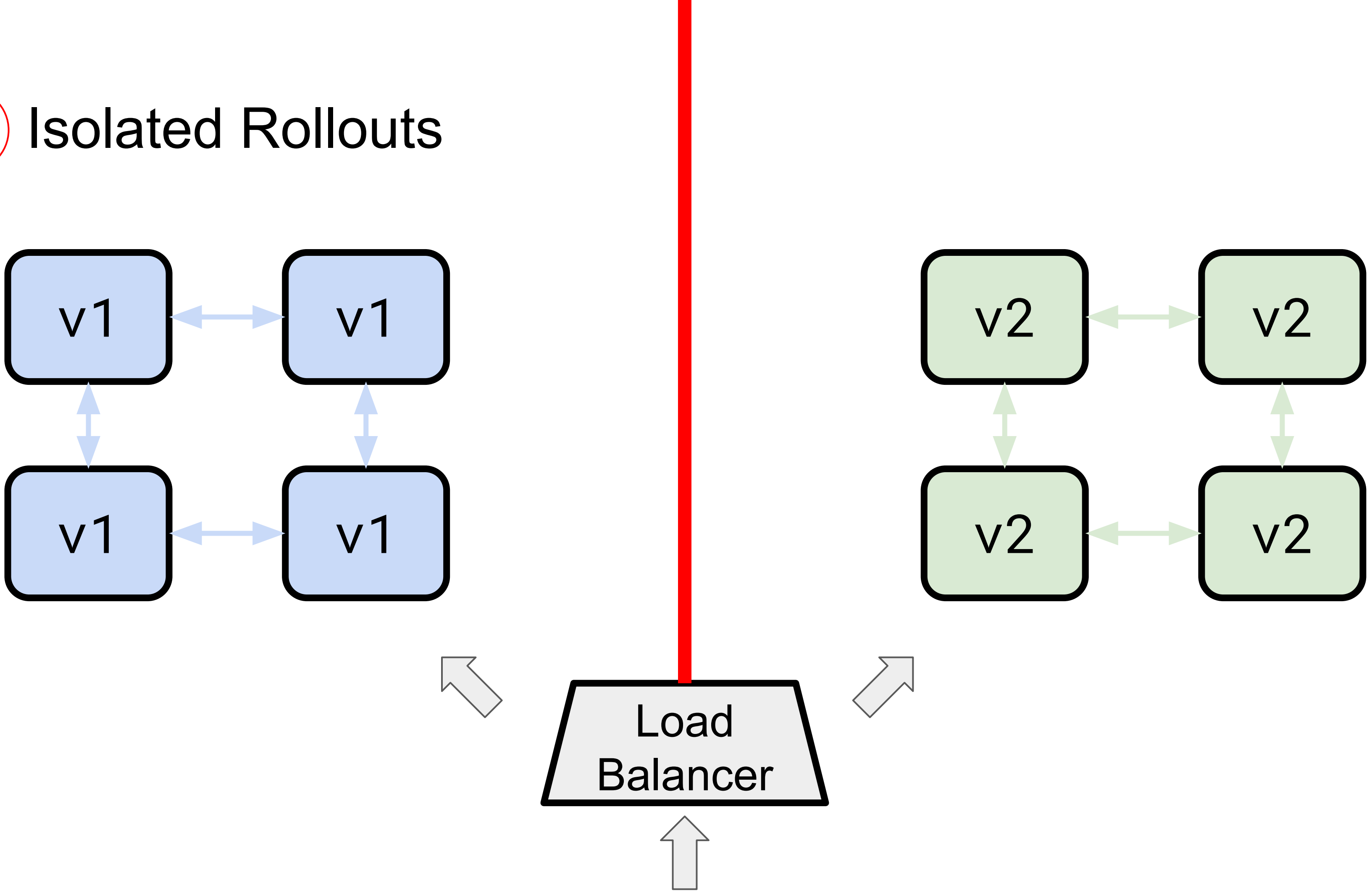
Runtime



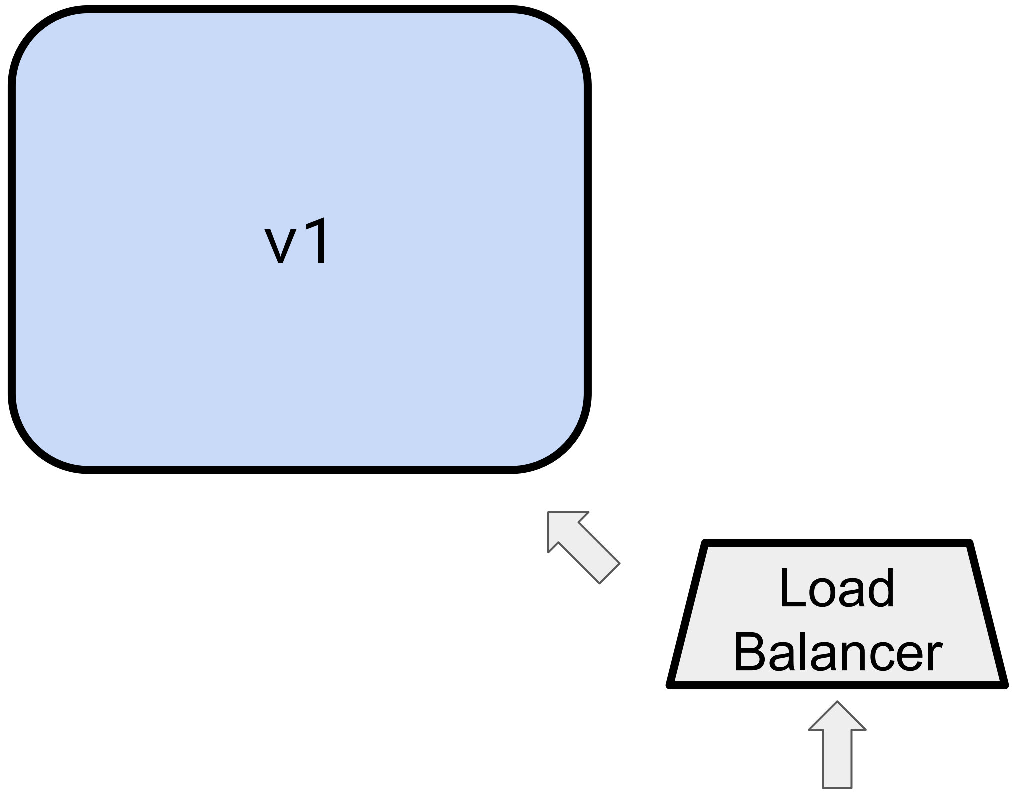

10010
01011
00011
01010
Local Deployer
AWS Deployer
SSH Deployer
GCP Deployer
Kubernetes Deployer


Manager
Control Flow
envelope
envelope
envelope
proclet
proclet
proclet
A
B
C
Data
Flow

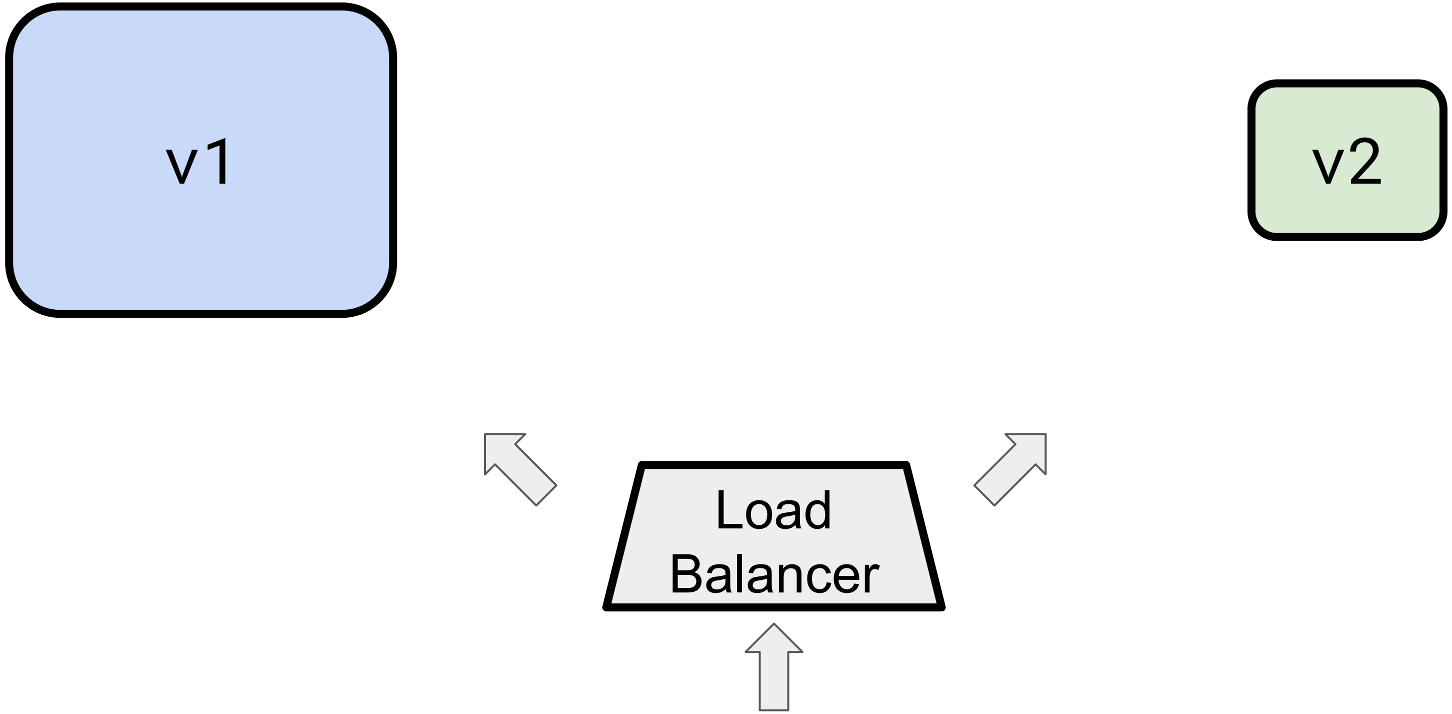
2 Isolated Rollouts



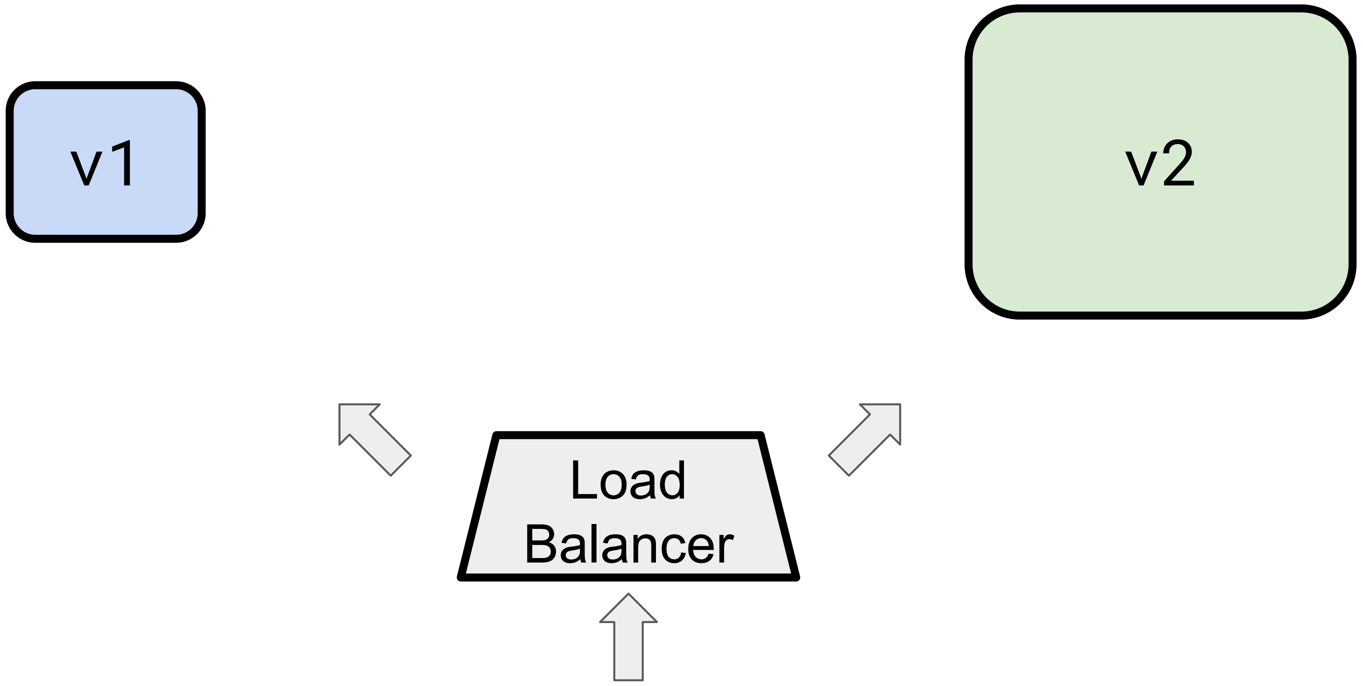
Blue Green Rollouts



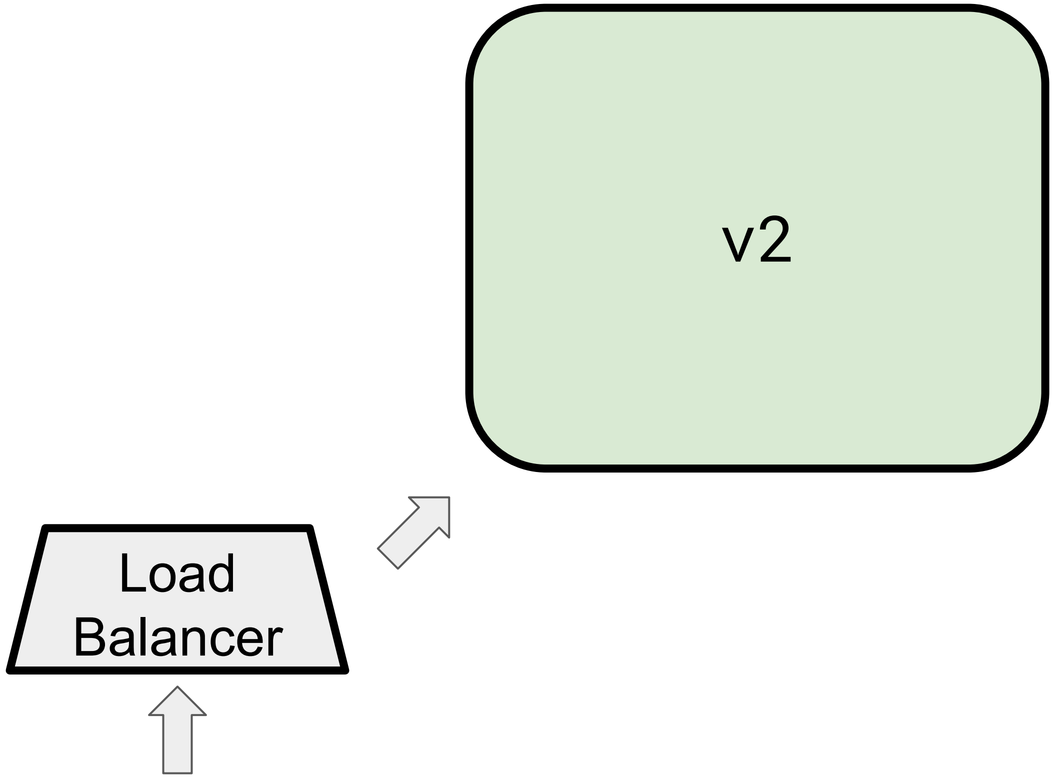
Blue Green Rollouts



Blue Green Rollouts



Blue Green Rollouts



INNOVATIONS

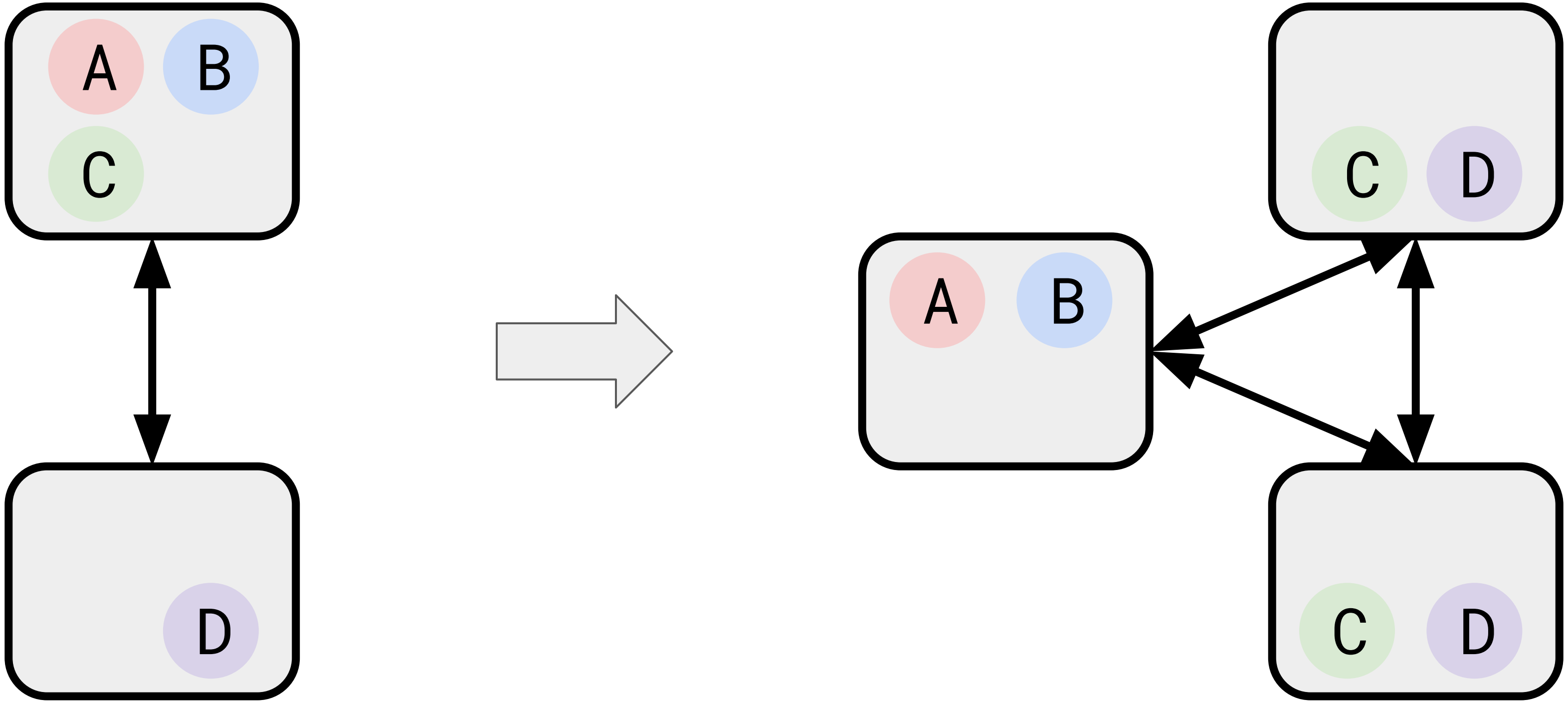
Innovations: Serialization

```
type pair struct {
    x, y int32
}

pair {
    0xAAAAAAAA,
    0xAAAAAAAA,
}
```

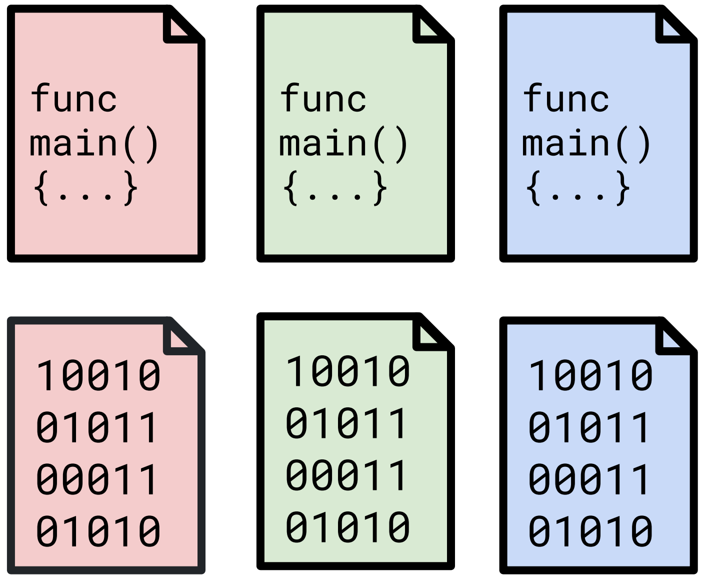
```
Protobuf serialization:

  08 8A D5 AA D5 2A
  10 8A D5 AA D5 2A
```

```
Custom Serialization:

  AA AA AA AA
  AA AA AA AA
```

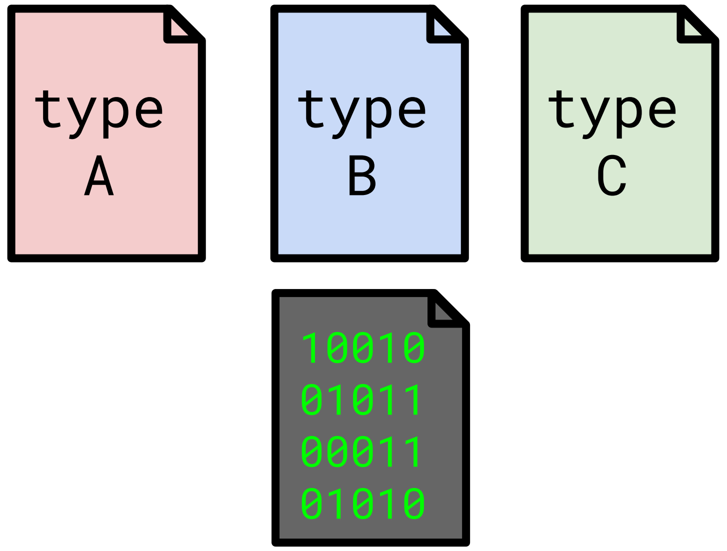
Innovations: Smart Scaling and Placement



Innovation: Testing



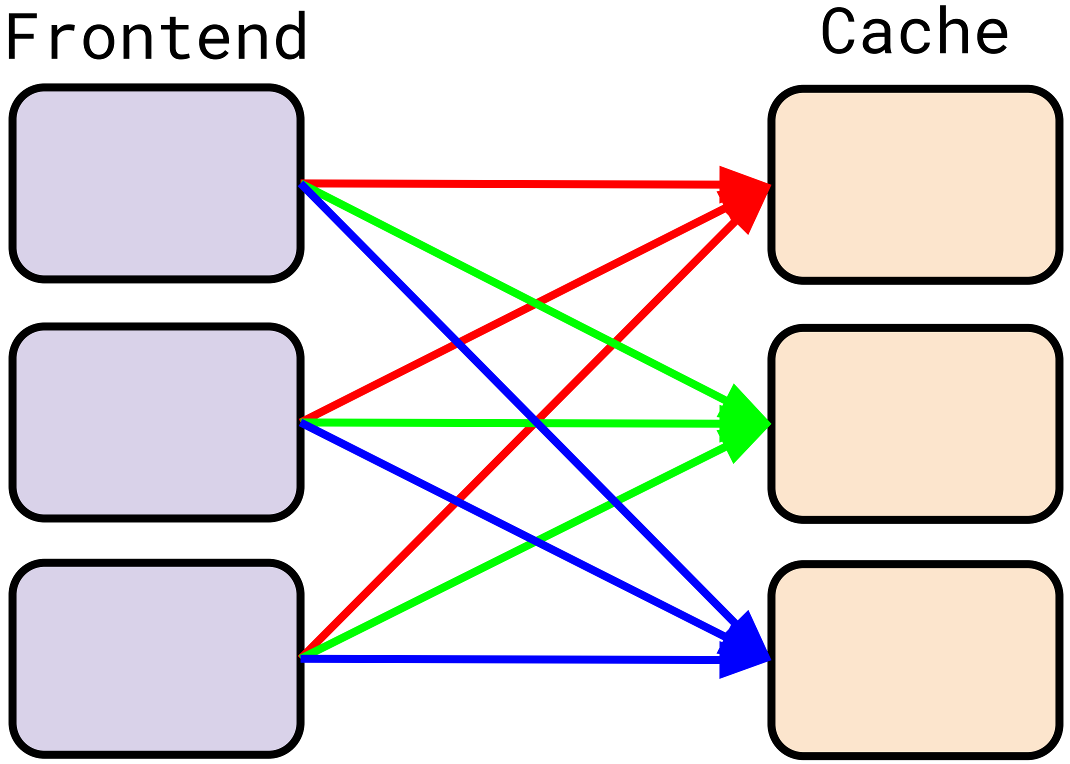
test???



unit tests
failure tests

Innovations: Routing

Fast key-value stores: An idea whose time has come and gone [HotOS’19]



BENCHMARKS

- Online Boutique application
- 11 microservices
- C2-medium-8 nodes on GKE
- 10,800 QPS load

Metric	Baseline	Prototype (split)	Prototype (merged)	Gains
Go code	2647 lines	2117 lines	2117 lines	up to 1.25x
Autoscaled to	77.7 CPU	27.7 CPU	9.11 CPU	up to 8x
Median latency	5.47 ms	2.66 ms	0.38 ms	up to 14x
99p latency	18.87 ms	9.24 ms	2.47 ms	up to 7x

RELATED WORK

Related Work (Actor Frameworks)

- Orleans
- Akka
- Cloudflare Durable Objects
- Ray
- Erlang
- C++ Actor Framework
- ...

Bigger focus on rollouts, versioning, portability, and simplicity.

Related Work (Serverless Functions)

- AWS Lambda
- Cloud Run
- Cloud Functions
- App Engine
- Azure Functions

Easier to integrate multiple services together.

Related Work (Physical and Logical Decoupling)

- Databases
- Data processing systems
- Dataflow systems
- ML training systems
- ...

Same idea, but for serving systems.

https://serviceweaver.dev 🧶

serviceweaver@google.com