Improving Go Backend Developer Experience in Grab

Chew Chee Ming

Grab
Engineering Manager,
Developer Experience

Chew Chee Ming

🇲🇾 Durian eater in Dali, Yunnan

🏃 Runs & Love the outdoors

☕ Makes and drinks Coffee

GopherChina 2021

目 录

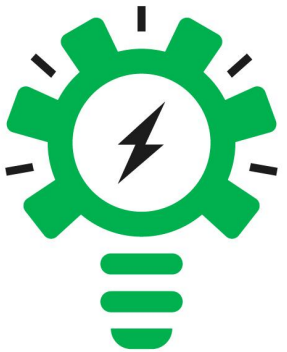
第一部分

Developer Experience

Developer Experience

GOAL

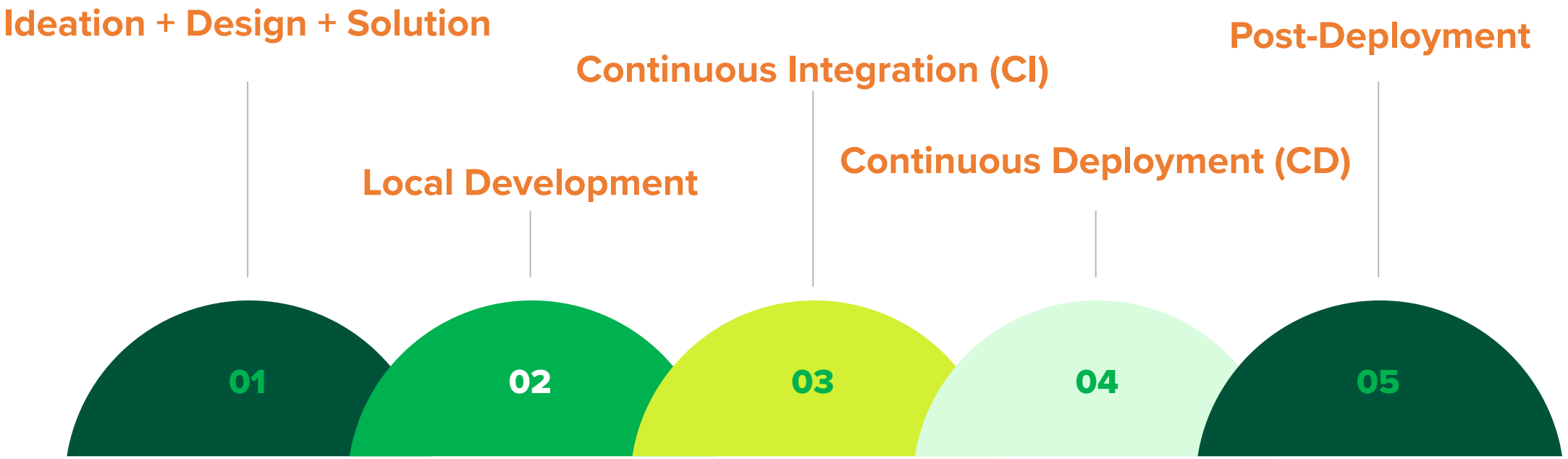
Increase engineering productivity and velocity, while improving overall system availability and stability.



DevExp

“Joy & Excellence”

Shipping a new feature

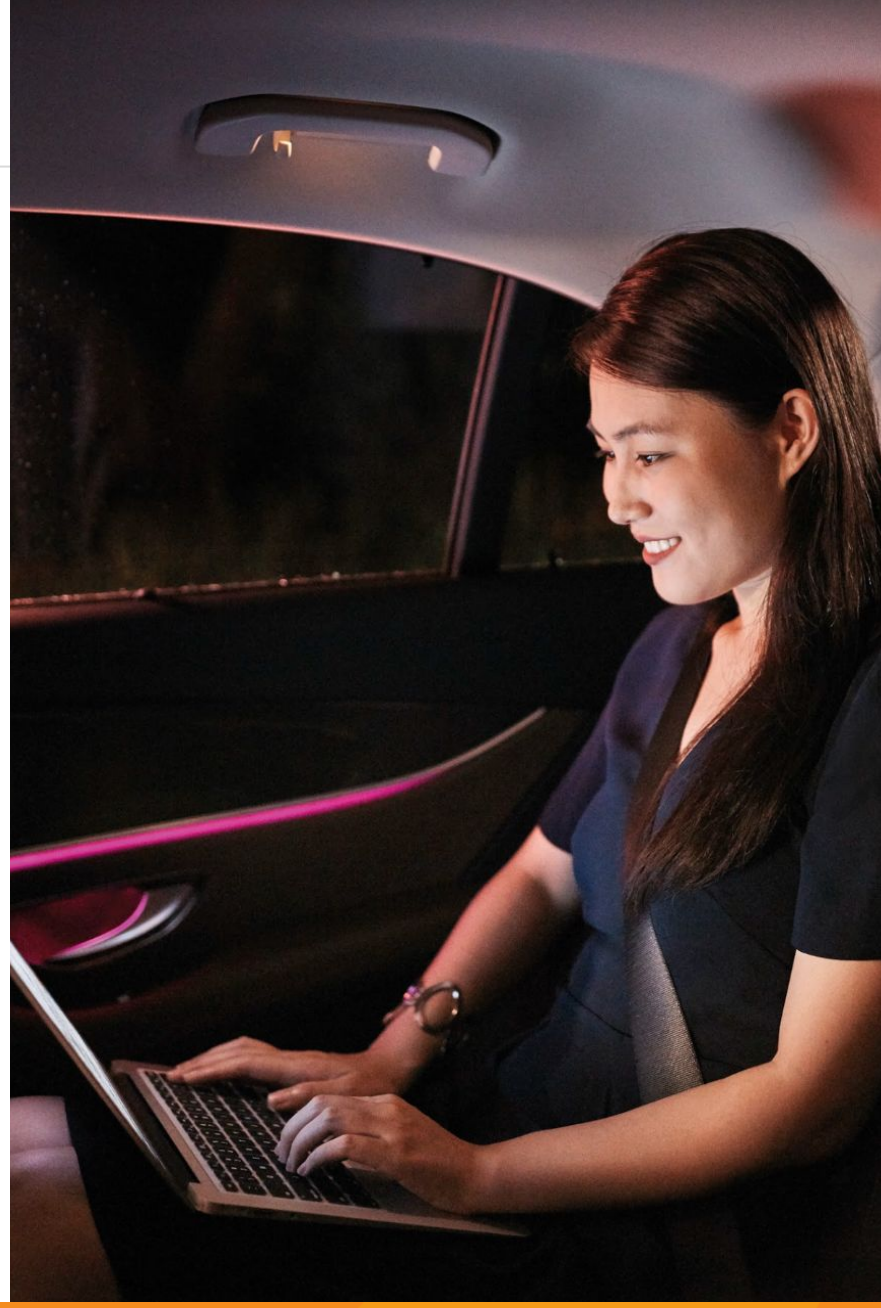


01. We look at the overall developer experience... and look for ways to improve it

Let’s say we want to build a new feature

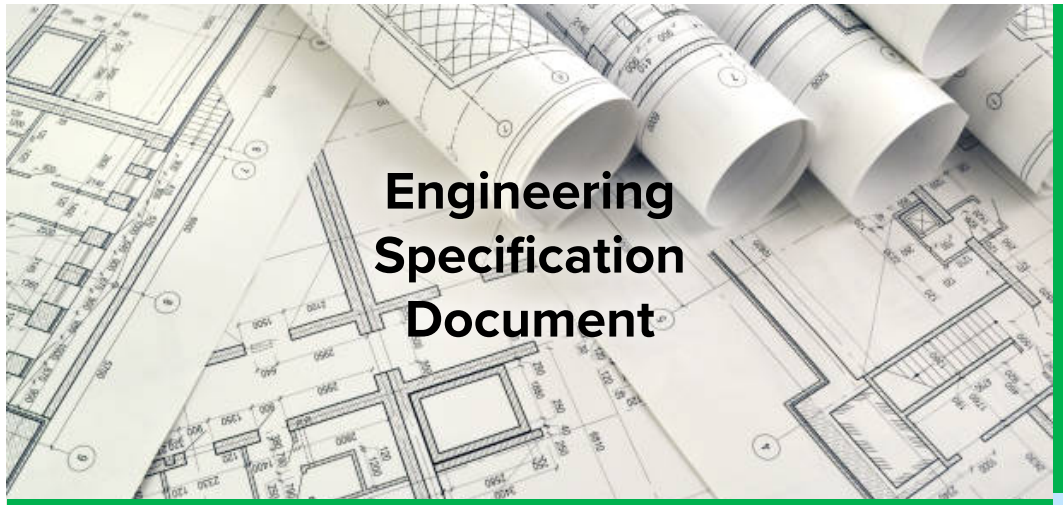
NEW IDEA

As a customer, I would like to listen to my favourite song when I enter the car.



01. Charlie is a Grab engineer who is going to work on this project.

Ideation + Design + Solution



Request for
Comments (RFC)

Proof of Concept
(POC)

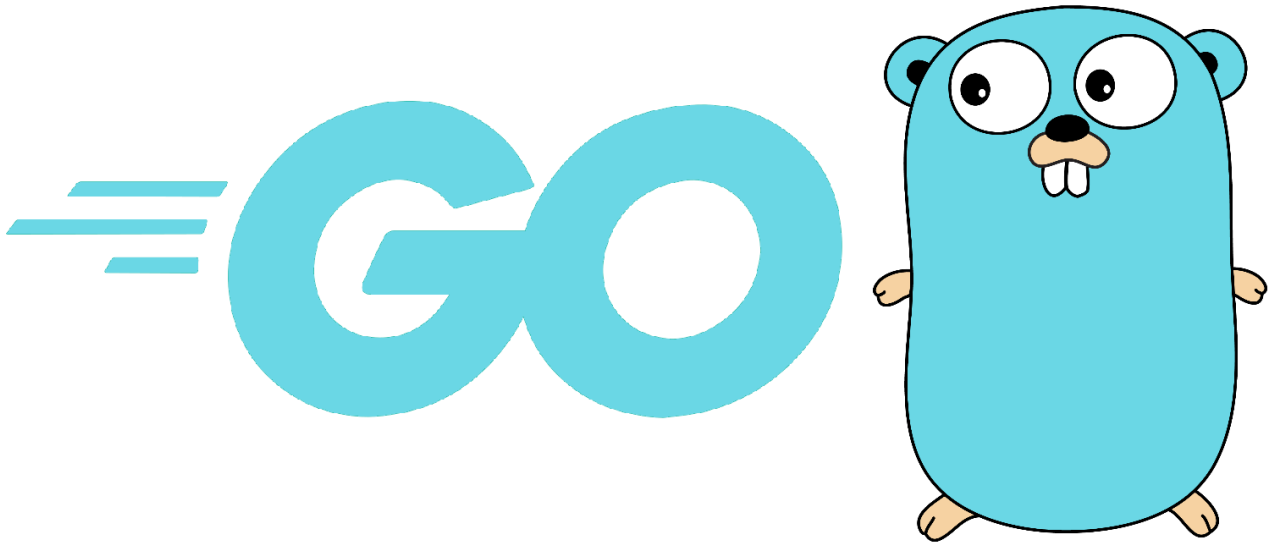
第二部分

Local Development

Writing Code: IDEs

Visual Studio Code

Writing Code: Go Lang



Current: v1.16

Writing Code: New Service

Automated Service Creation → Eden

- Scaffold new code
- Automate basic configurations
- Setup CI/CD pipelines
- Run “hello world” service

02. Charlie needs to create a new service for this feature.

Writing Code: Grab-Kit (GK)

WHAT

Grab-Kit, an RPC framework for microservices, creates skeleton microservices and generates boilerplate code from API definitions.

WHY

- Save time & effort
- Standardisation
- Good defaults
- Best practices

HOW

- Code generation
- DTOs Integrated with Protobuf
- Full scaffolding
- Runnable by default
- DAO generated
- Middlewares
- Client Library

GK: Code Generation

```
? Scaffold kafka Consumers?  No
? Scaffold kafka Producers?  No
? Brief service description (single line) PlayMyMusic
? Wiki location for your service (single line) https://wiki.grab.com/display/TECH/CHANGEME
? Slack channel for your service (single line) #CHANGEME
? Slack alias for your service (single line) @CHANGEME
? Phabricator Tag for your service (single line) #CHANGEME
•  Not flattening the request / response objects in generated stub handlers. Using DTO types directly.
•  Created config file grabkit.yaml
•  Your service name will be referred to as playmymusic
•  Creating new service Playmymusic in /Users/cheeming.chew/gopath/src/gitlab.myteksi.net/gophers/go/playmymusic...
•  Creating bin/
✅  Scaffold complete
•  Generating code...
✅  Applying patches...
 14 / 14 🚦▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒🏁 100.00% 1s

🎉  Service scaffolding done! Refer to the readme in /Users/cheeming.chew/gopath/src/gitlab.myteksi.net/gophers/go/playmymusic for next steps.
💡  PLAYMYMUSIC_CONF was added to scripts/env-vars.sh. Run 'source scripts/env-vars.sh "" -ci' or set PLAYMYMUSIC_CONF manually before starting the service.

✅  Generated gandalf test files for service...
 1 / 2 🚦▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒🐌____________________🏁  50.00%
✅  Generated Mock server files for service...
 3 / 2 🚦▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒🏁 150.00% 8s
Additional processing complete

🌈☀️ Process complete. Enjoy your new service!
```

GK: Structure and Important Files

./grabkit.yaml
./config-ci.json
./pb/playmymusic.proto
./client/*
./cmd/server/main.go
./server/serve.go
./server/config/config.go
./server/handlers/handlers.go
./internal/*
./examples/*

GK: Protobuf

Here is a Hello World example that is generated by default by our Grab-Kit to let engineers learn how to use things with sample code.

```
syntax = "proto3";

package playmymusicpb;

import "google/protobuf/empty.proto";
import "google/api/annotations.proto";
import "protoc-gen-swagger/options/annotations.proto";

// Generated by Grab-Kit - do not change this package
option go_package = "gitlab.myteksi.net/gophers/go/playmymusic/pb";

// The base path is only for Swagger and does not affect the routing - do not change this.
option (grpc.gateway.protoc_gen_swagger.options.openapiv2_swagger) = {
  base_path: "/playmymusic"
};

service Playmymusic {
  rpc Hello(HelloRequest) returns (HelloResponse) {
    option idempotency_level = NO_SIDE_EFFECTS;
    option (google.api.http) = {
      post: "/hello"
      body: "*"
    };
  }
}

message HelloRequest {
  // Use underscore_separated_names for field names. See
https://developers.google.com/protocol-buffers/docs/style

  string name = 1;
}

message HelloResponse {
  string message = 1;
}
```

GK: Handler

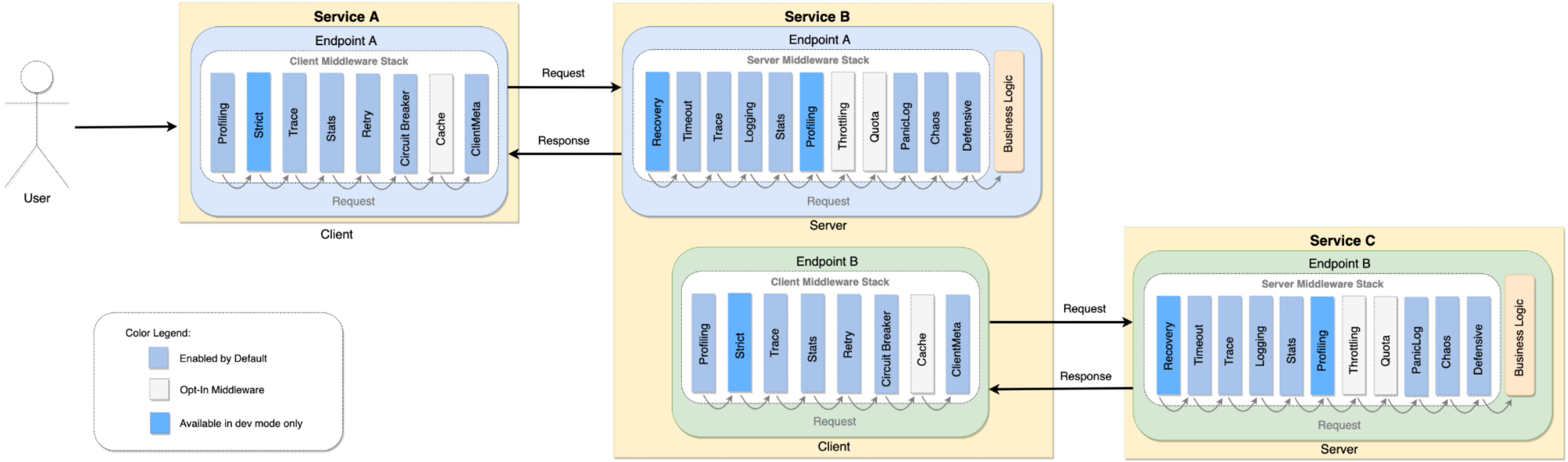
So this is the handler function that maps to the definition in the protobuf.

```go
// Copyright (c) 2012-2021 Grabtaxi Holdings PTE LTD (GRAB), All Rights Reserved.

// @generated

package handlers

import (
    "context"
    "errors"
    "fmt"

    "gitlab.myteksi.net/gophers/go/playmymusic/dto"
)

// PlaymymusicServiceDNB implements z_api.GKPlaymymusicServiceDNB
type PlaymymusicServiceDNB struct{}

// NewPlaymymusicServiceDNBImpl returns a new z_api.GKPlaymymusicServiceDNB implementation
func NewPlaymymusicServiceDNBImpl() *PlaymymusicServiceDNB {
    // This function should take your service's dependencies as parameters and store them
in the Service object for use by
    // the handlers. This can be used for dependency injection.
    return &PlaymymusicServiceDNB{
        // TODO: Add dependencies here
    }
}

// Hello implements the Hello endpoint
func (s *PlaymymusicServiceDNB) Hello(ctx context.Context, req *dto.HelloRequest)
(*dto.HelloResponse, error) {
    if req == nil {
        return &dto.HelloResponse{Message: "<nil>"}, errors.New("nil response sent")
    }
    return &dto.HelloResponse{Message: fmt.Sprintf("hello %s", req.Name)}, nil
}
```

GK: It is alive!

```
ITCN000428-MAC:go cheeming.chew$ curl http://localhost:8088/playmymusic/hello -H 'Content-
Type: application/json' --data '{"name":"gophers cn 2021"}'
{"Message":"hello gophers cn 2021"}
```

GK: Middlewares



Coding Complete!

Code changes needs to get reviewed.

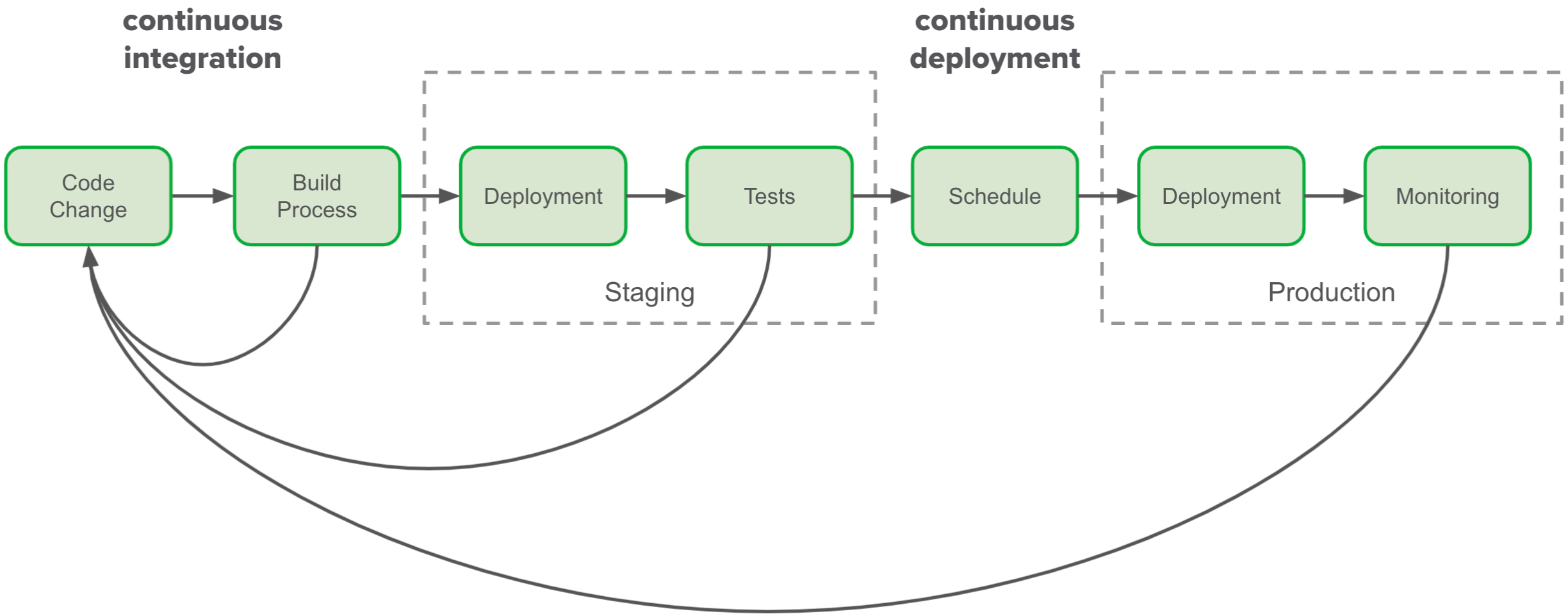
This keeps the quality high.

- Create a Merge Request
- Other engineers will review
- Repeat, until comments resolved
- Land the code into master

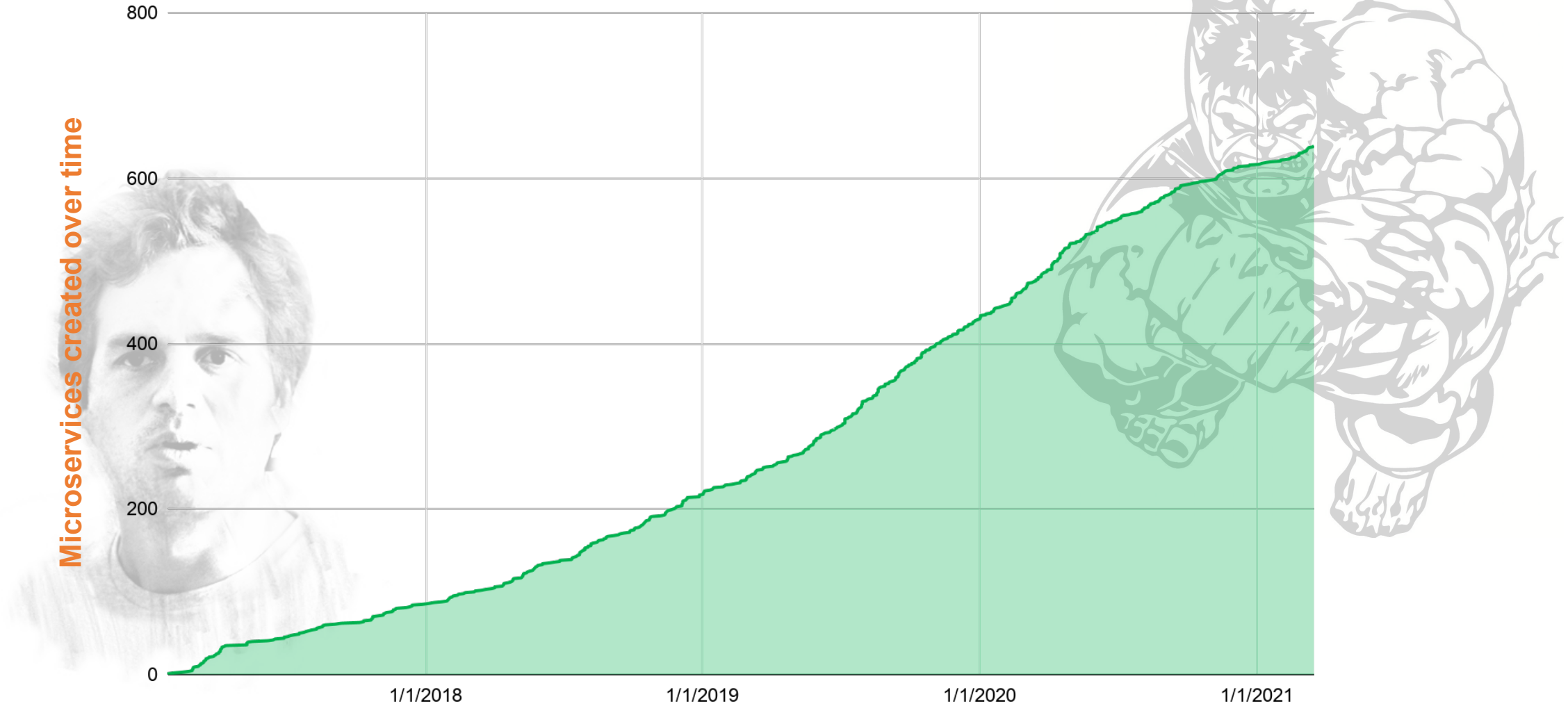
第三部分

CI/CD

CI/CD


continuous integration
continuous deployment
Code Change
Build Process
Deployment
Tests
Schedule
Deployment
Monitoring
Staging
Production

Go monorepo & microservices (2017-2021)



4 Years later...

600
MICROSERVICES

140
COMMITS /
WORKING DAYS

1,000
ACTIVE
CONTRIBUTORS

210K
TOTAL
COMMITS

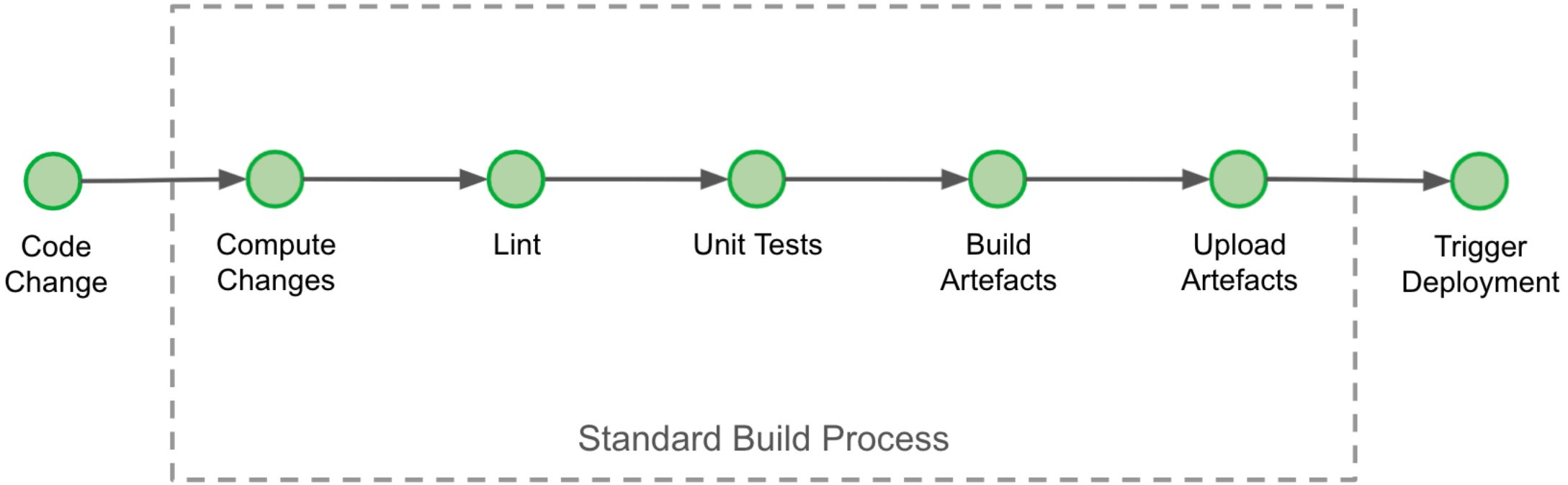
272
STAGING
DEPLOYMENTS / DAY

6M
UNIT TESTS / DAY

225K
INTEGRATION TESTS / DAY

60
PRODUCTION
DEPLOYMENTS / DAY

Build System (CI)



Build Times

- Breaks flow of engineers, lag time leads to lower productivity
- 1 min increased build time could result in landing code 1 hour later
- Build times too long

  p50 ~15mins

  p90 ~38mins
- Start fixing the problem

  focus on p50
- Finally, p50 ~10mins

Linting...

convention/context_package
convention/copyright
convention/early_return
convention/error_string
convention/errorf
convention/exported_comments
convention/format
convention/incdec
convention/names
convention/package_comments
convention/ranges
convention/readme
convention/receiver_name
convention/return_value_order
convention/testdelay
convention/unkeyed_literal
convention/vardecls
convention/
zero_value_constructor
lint/assert_check
lint/blank_import
lint/bool_check
lint/build_tag
lint/context
lint/copy_lock
lint/deep_equal_check
lint/define_const
lint/duplicate_check
lint/duration_check
lint/errcheck
lint/import_dot
lint/loop_variable
lint/nilfunc
lint/printf
lint/shift
lint/struct_align
lint/struct_check
lint/struct_tag
lint/unconvert
lint/unexported_return
lint/unreachable_code
lint/varcheck
metric/cyclomatic_complexity
metric/function_length
metric/interface_length
metric/nesting_level
metric/struct_length

Lint: Duration Check Analyser

Issues

L4000: Implicit conversion to time.Duration

The analyser detects any implicit conversion to `time.Duration`.

Fix: Explicitly specify the type for time.Duration.

```
time.After(60) // flagged

// Fix
time.After(60*time.Second)
```

Convention: Early Return Analyser

Issues

C1300: Prefer early `return` over `else`

The analyser found an `if` with an `else` clause. There is a shared path of execution after the if-else construct, and you can duplicate ("hoist") the code to remove the `else` clause.

Fix: Duplicate the common code path for both if and else clauses, then move the contents of the else block one level up. As an example, replace the assignment to `c` with a `return` statement, and remove the `else` clause.

```
1  func HoistAndReturn() int {
2     var c int
3     if true {
4        c = 3
5     } else { // unnecessary else
6        c = 4
7     }
8     return c
9  }
10
11 // Fix
12
13 func HoistAndReturn() int {
14    if true {
15       return 3
16    }
17    return 4
18 }
```

Metric: Nesting Level Analyser

Issues

M1500: Function is deeply nested [%d/%d]

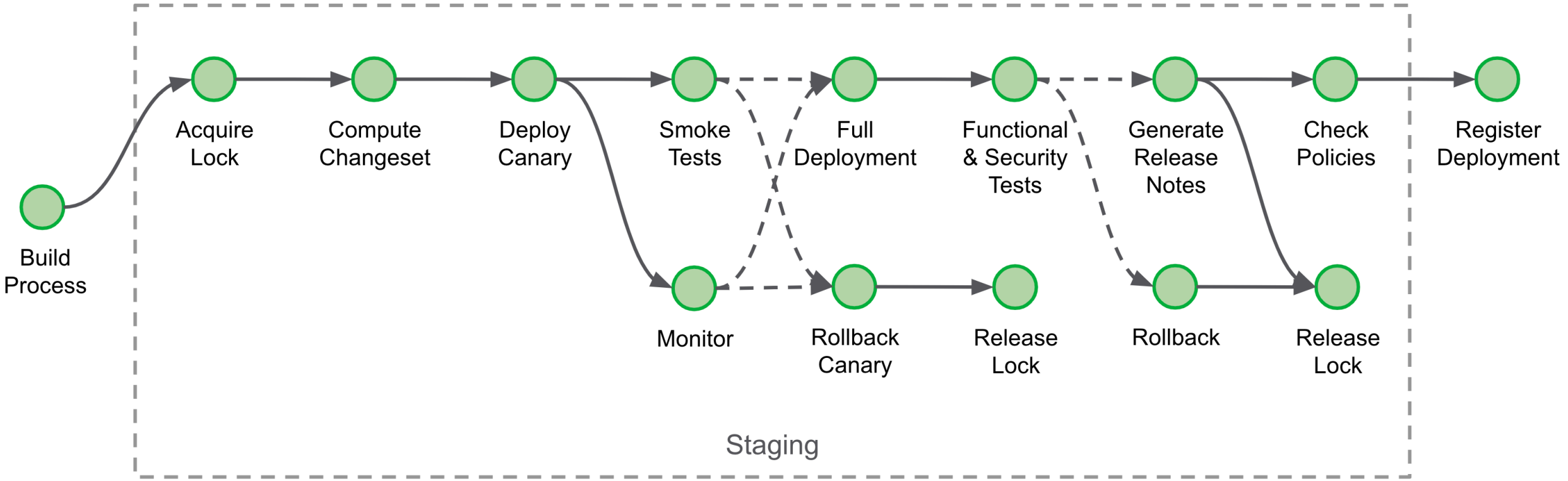
The analyser flags out deeply nested functions. The following example shows functions that are deeply nested using if and for constructs.

Fix: Refactor the function to have fewer nesting levels. Extract the flagged area to a separate function, or to a lambda in the same function.

Deeply nested functions

```
func DeeplyNested1() {
        if true {
                if true {
                        if true {
                                fmt.Println("Deeply nested code")
                        }
                }
        }
}

func DeeplyNested2() {
        for {
                for {
                        for {
                                fmt.Println("Deeply nested code")
                        }
                }
        }
}
```

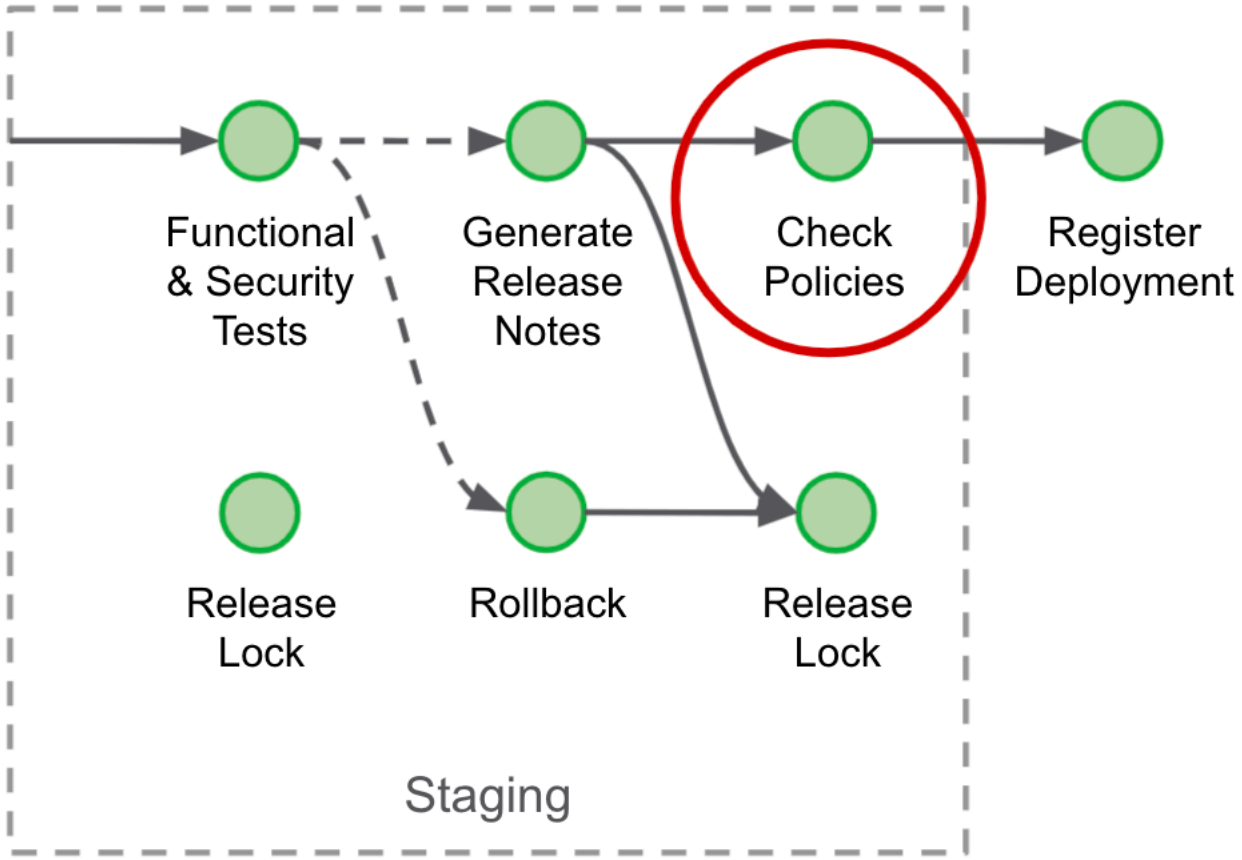
Staging Deployment Overview



Canary in the coal mine...

- Use canaries as early detectors
- We run “smoke tests” on canaries
- If we detect errors, we rollback
- If all good, we go full deployment

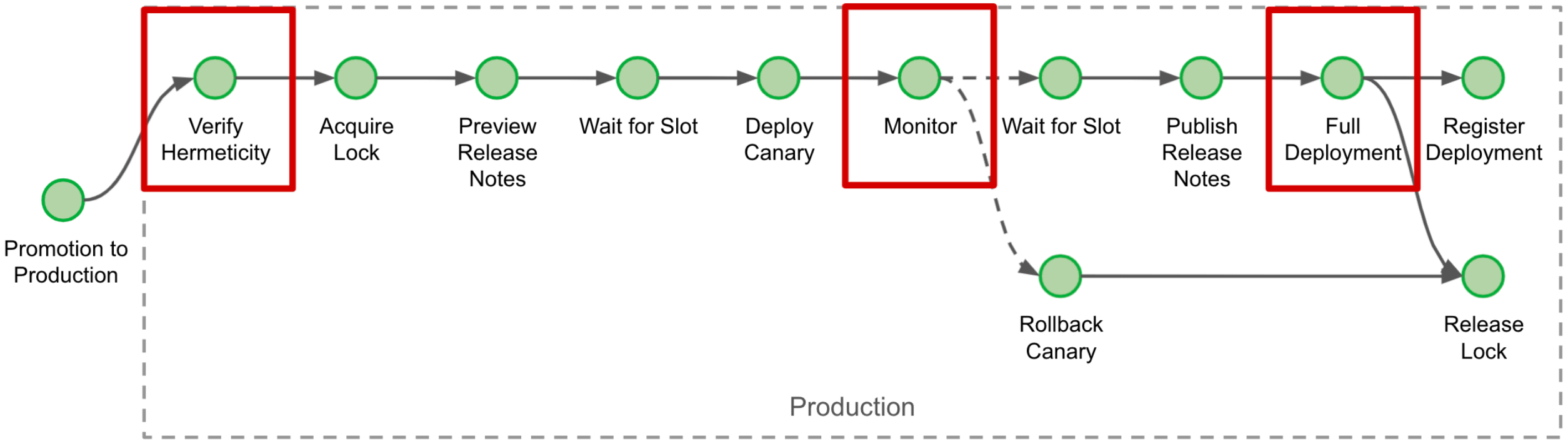
Policy Engine



Is it okay to promote to production?

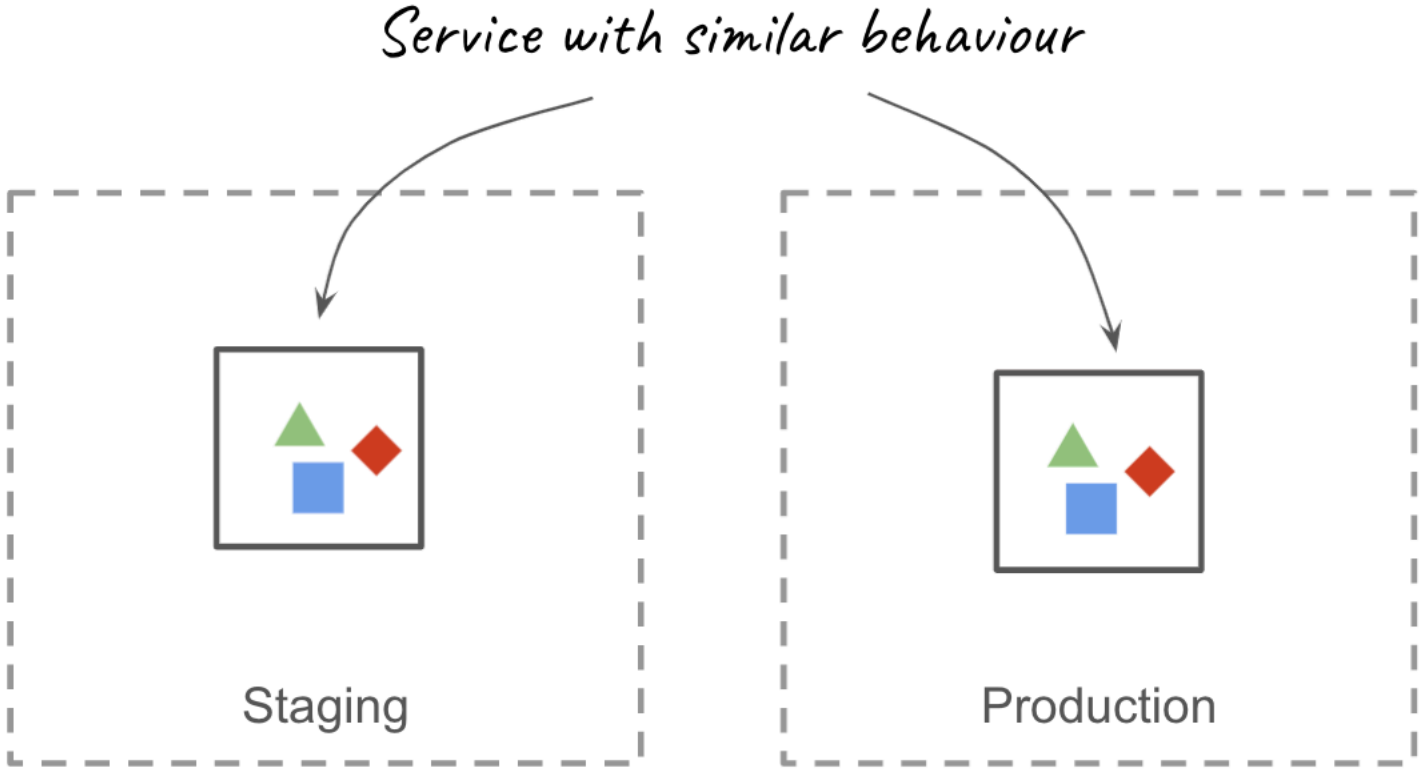
- Check Hygiene
  - Service Metadata
  - **Image Version**
- Quality metrics
  - **API Test Coverage**
  - Flakiness
- Security metrics
  - **Security Test Coverage**

Production Deployment Overview

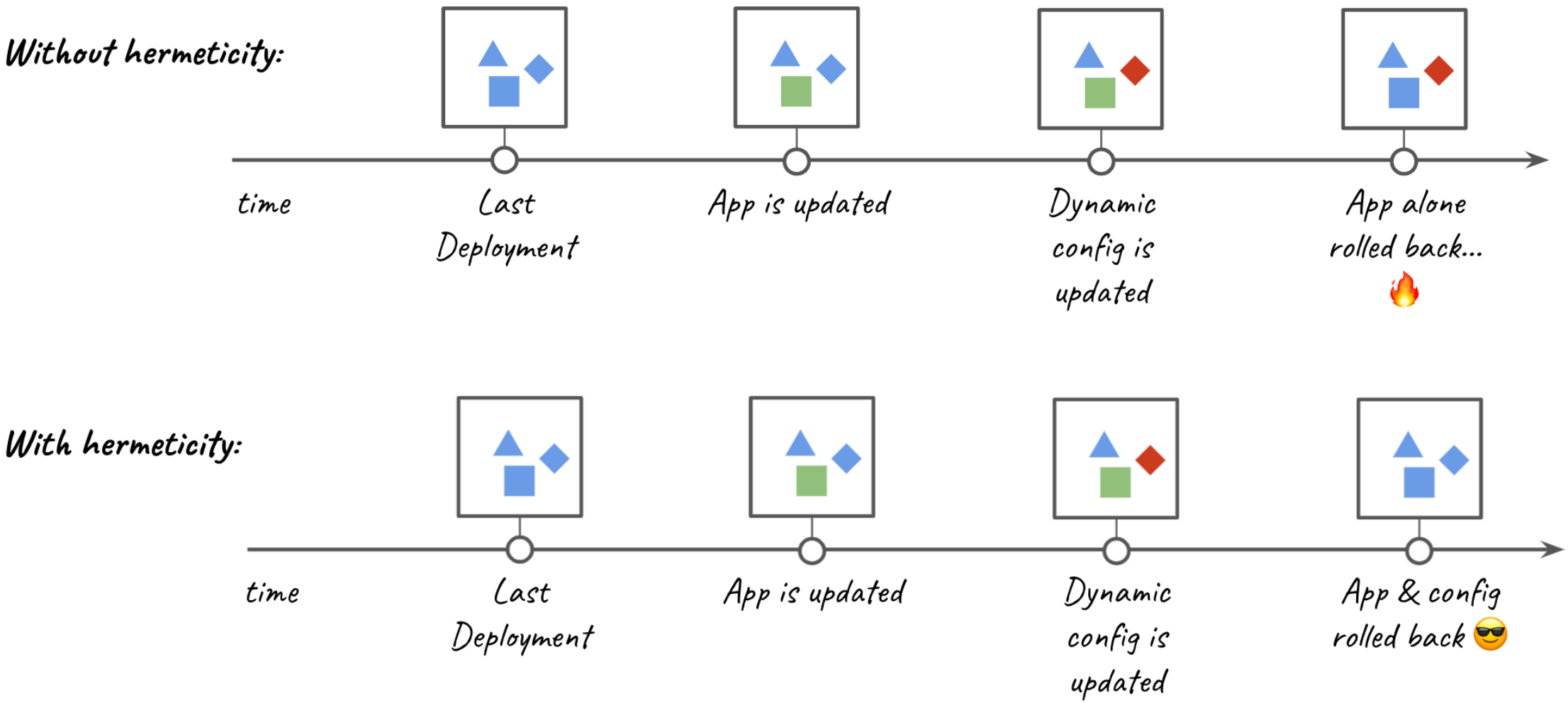


What did we actually test?

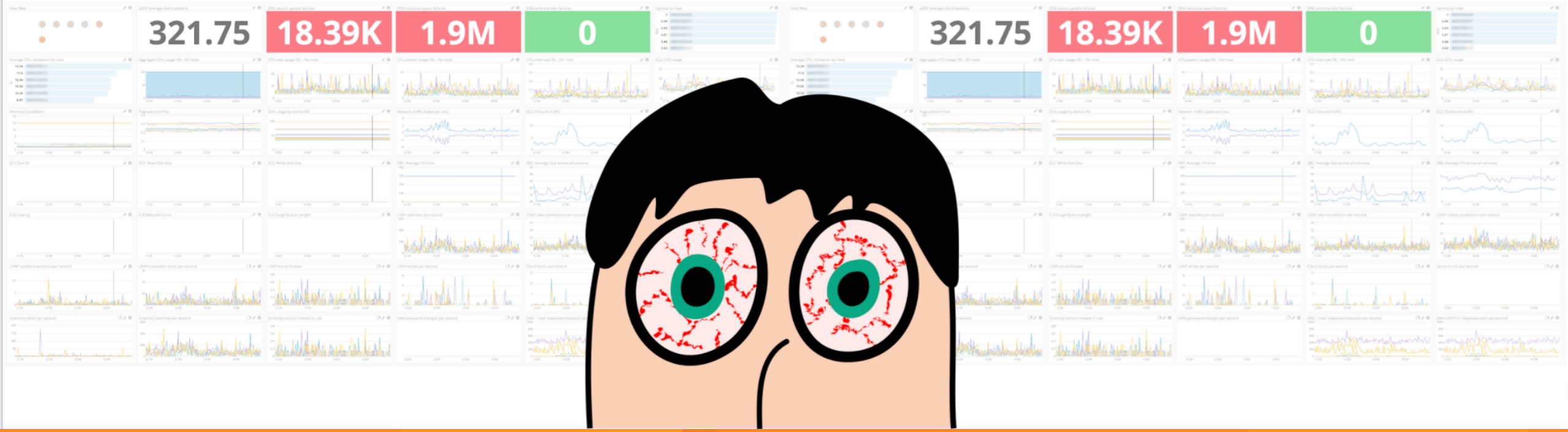
- Application version
- Static config version
- Dynamic config version



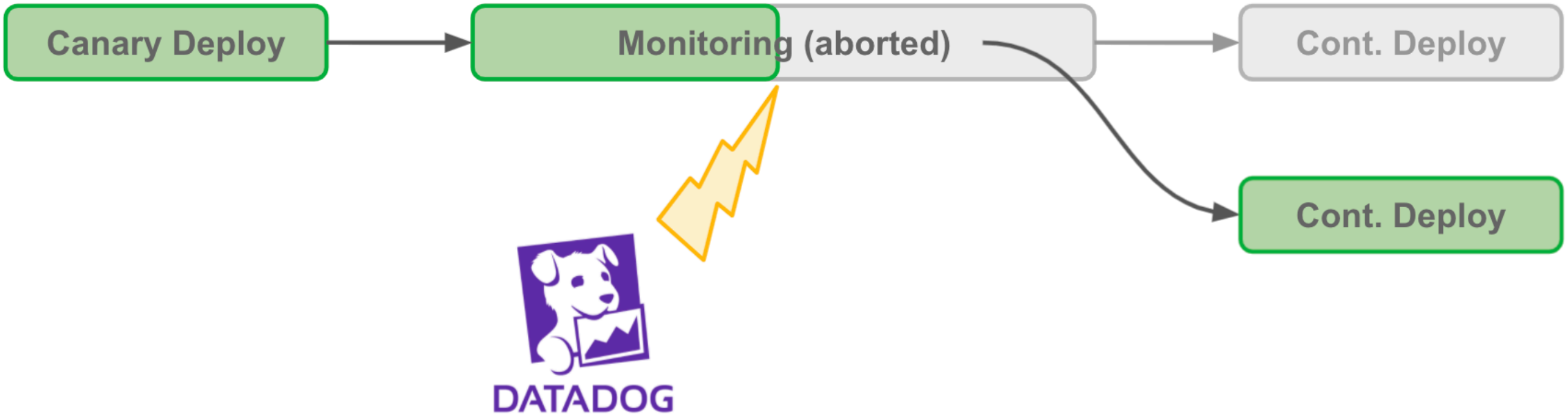
Make Deployments Reproducible: Hermeticity



Too many graphs....

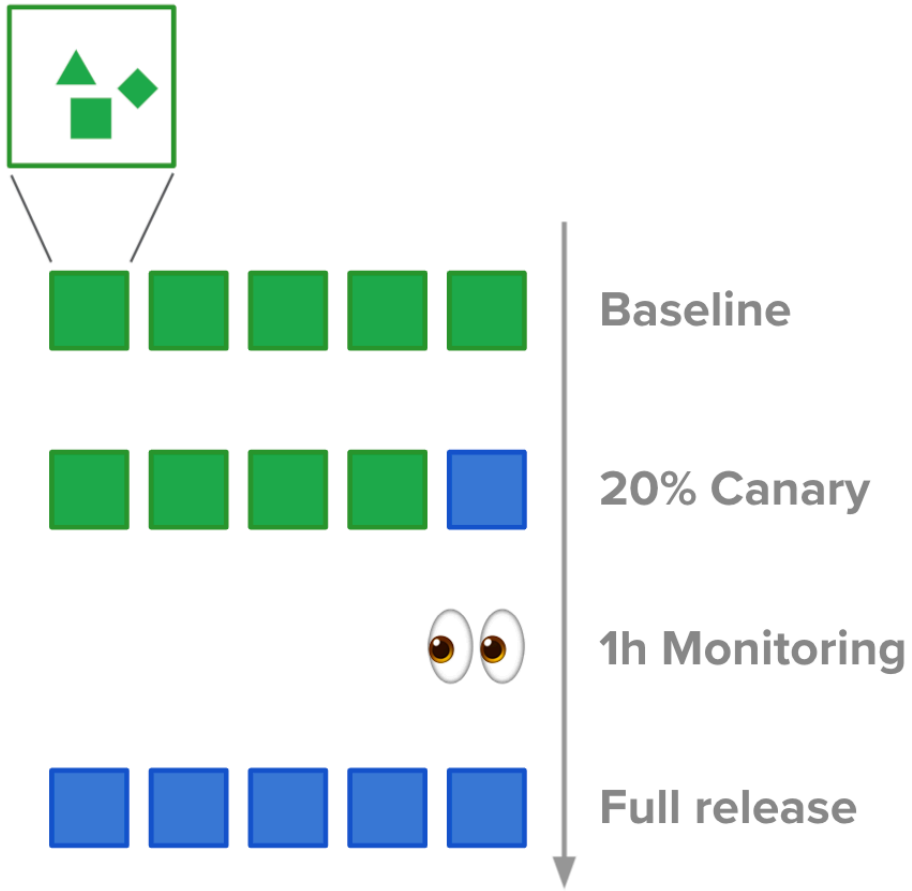


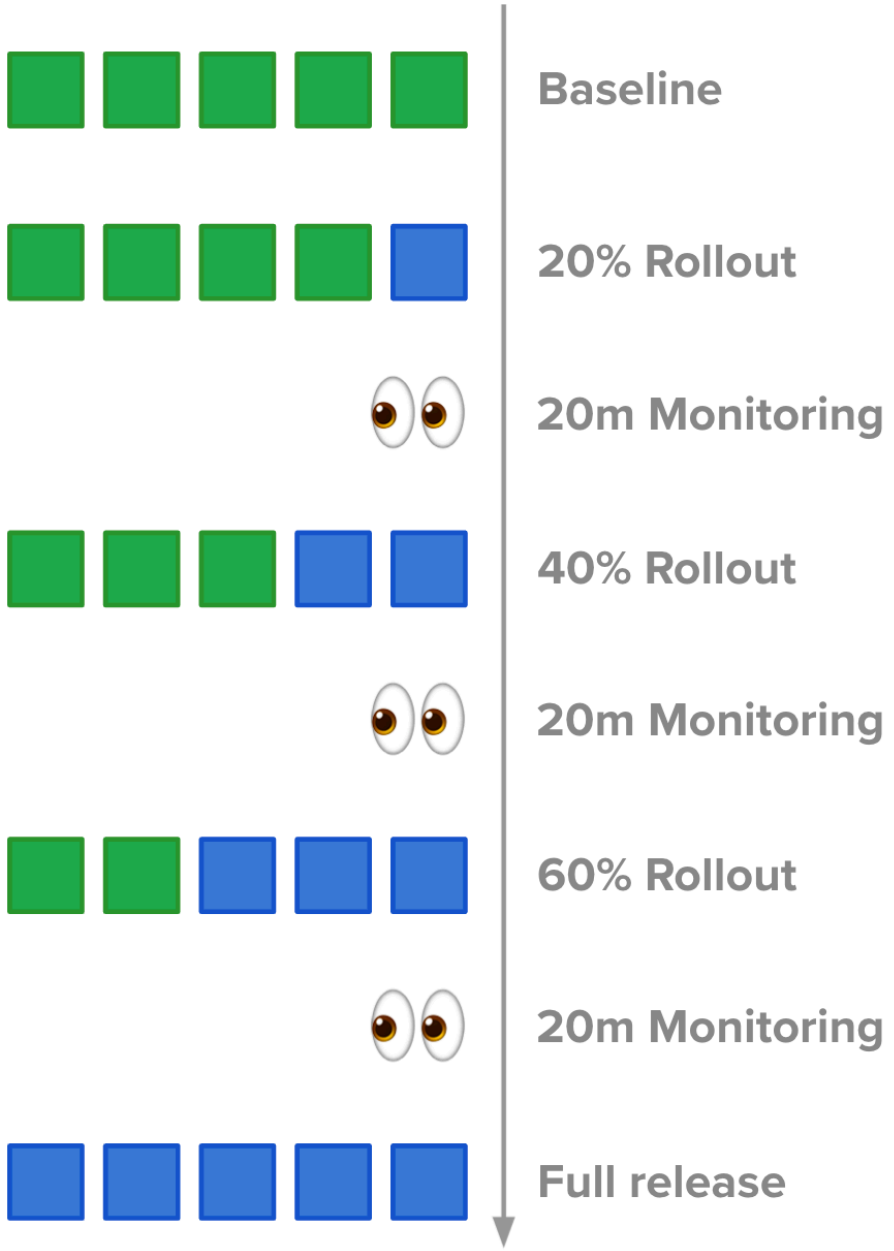
Self-healing where possible



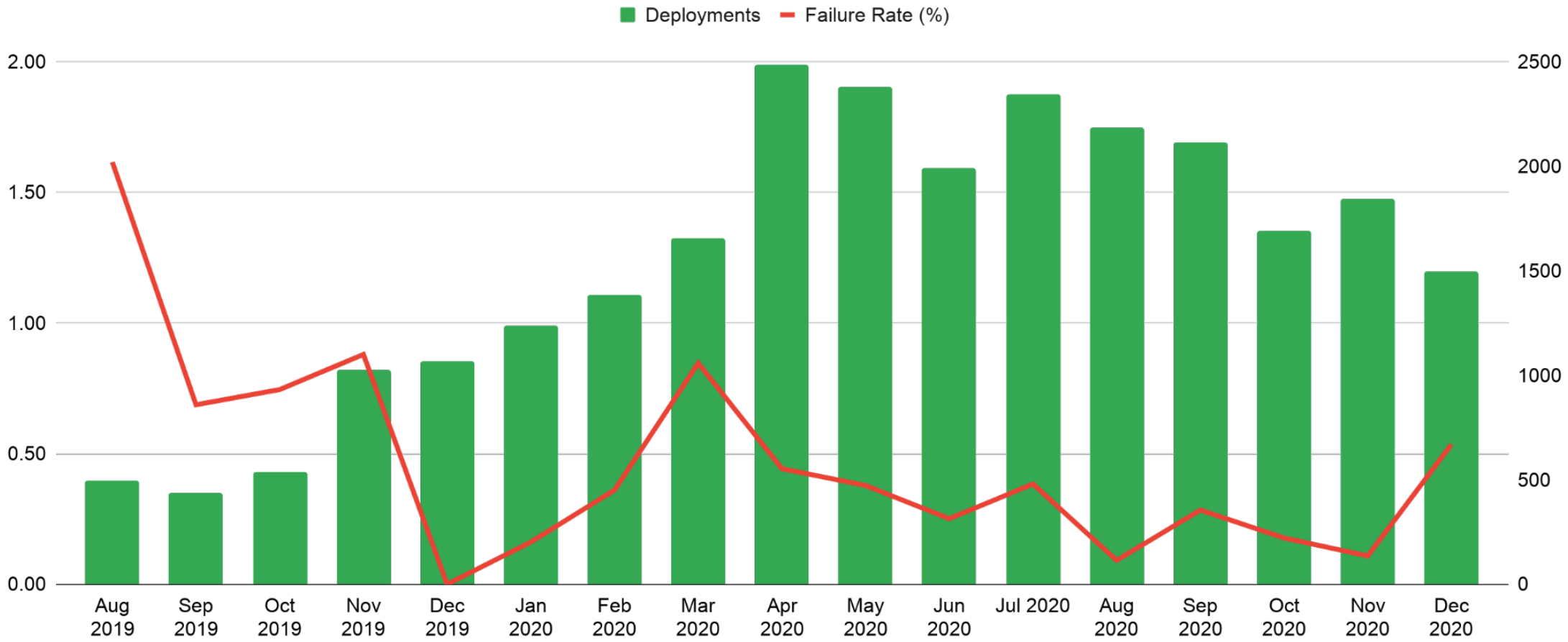
Faster reaction time than human (reverts within <1 minute). Less fatigue.

Deployment Strategies





Slow and steady wins the race!



第四部分

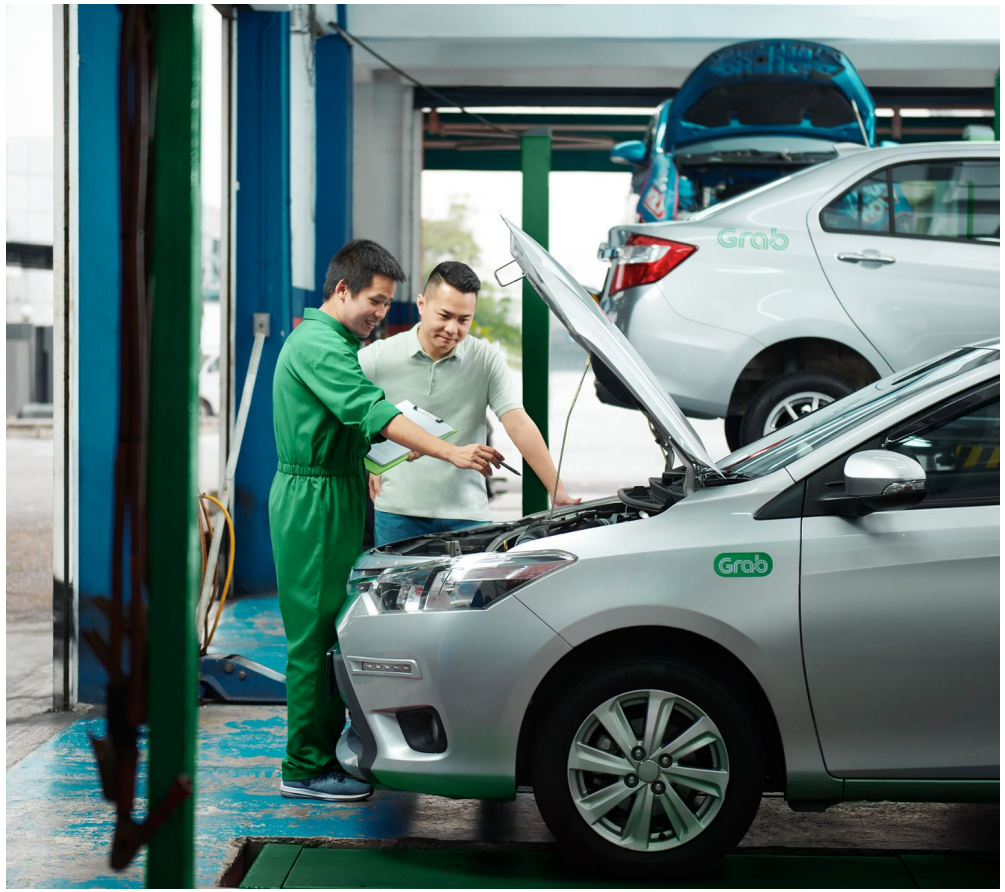
Post-Deployment

Helping engineers post-deployment...

When things are broken....
You want to know what happened?

- **Logging**
- **Metrics**
- **Tracing**

Structured Logging



t	app_name	ucm
t	availability_zone	ap-se-1a
t	common_endpoint	getRawConfig
#	common_endpoint_elapsed	1,492
t	common_request	[not shown]
t	common_response	[not shown]
t	common_service	ucm.ucm
t	common_trace_id	3c7e5903-e310-42c7-9fa2-8180a45992e2
t	event_type	middleware.logging
t	fileset.module	grab
t	fileset.name	structuredlog
t	hostname	ip-10-0-60-135
t	log_level	INFO

第五部分

Better Primitives

Better Primitives

Some areas we invested in:

01 Configuration System

02 Monitoring-as-code

03 Workflows System

04. Making developers life even better with better building blocks!

“

KEY TAKE AWAYS

01 Developer Experience

02 Local Development

03 CI/CD

04 Post-Deployment

05 Better Primitives

“Curiosity and Creativity”

👇👇

“Joy & Excellence”

Thank You!

Prepared by Chew Chee Ming

26th June 2021 | Version 1

This deck was made possible with the combined effort from:
Sylvain Bougerel, Biju Joseph Jacob, Lijuan Gao