

10 najlepszych praktyk dla Infrastructure-as-a-Code



Wojciech Barczynski | Spacelift.io

Co to jest OpenTofu / Terraform?

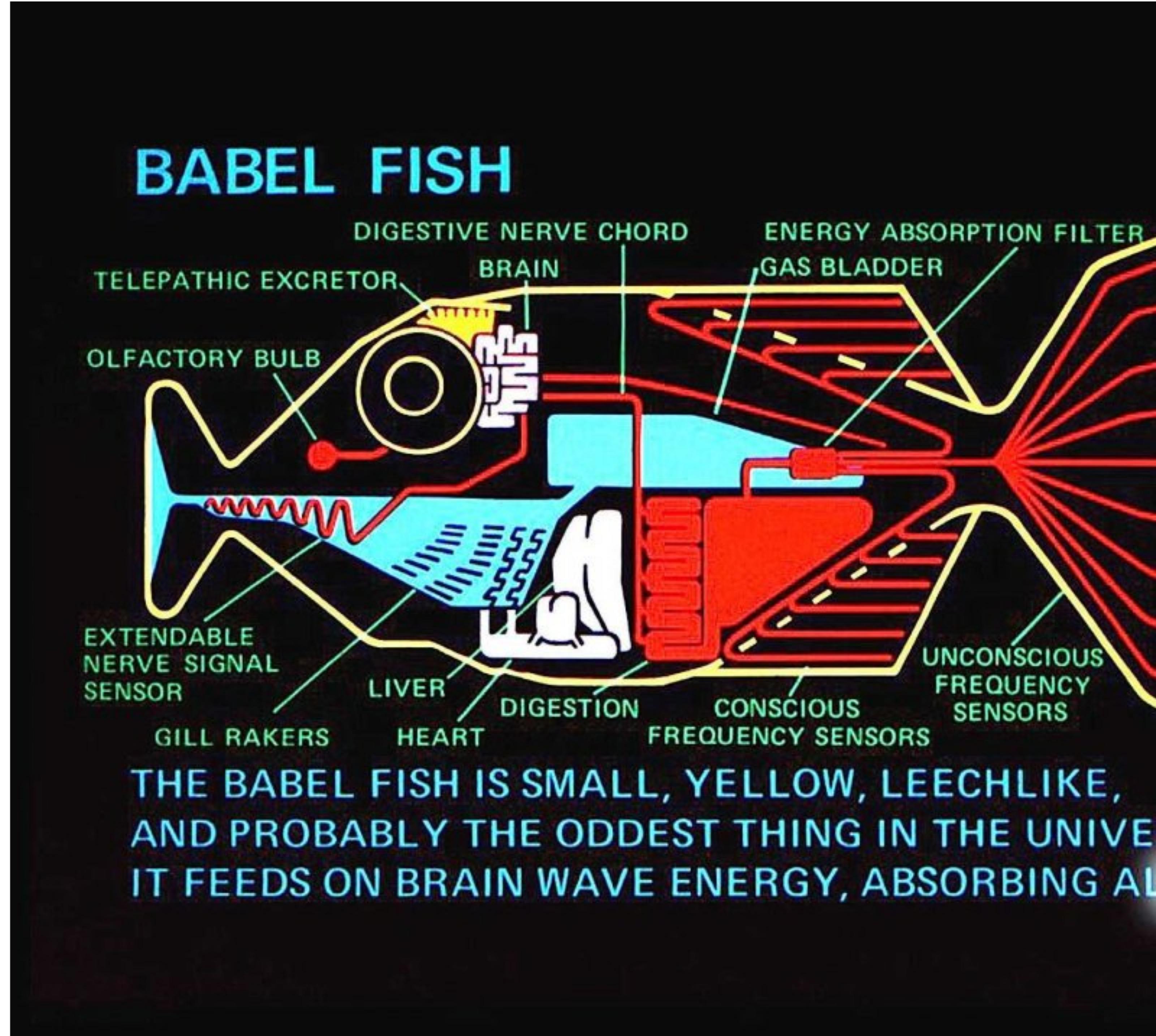
Demo:

```
variable "repository_name" {  
    type    = string  
    default = "my_app"  
}  
  
resource "github_repository" "my_repo" {  
    name          = var.repository_name  
    description   = "My Azure App repository created by OpenTofu!"  
    visibility    = "public"  
}
```

Dlaczego OpenTofu/Terraform FOSS?

1. `tofu plan` – co się zmieni
2. `tofu apply` – wykonanie planu;
3. plik stanu – korzyści ([demo](#)):
 - shifting X left,
 - polityki,
 - drift detection.

Common
Language
Artifacts
Platform



Dlaczego OpenTofu/Terraform FOSS?

- Doskonała dokumentacja ([tf/ot](#)),
- Bogaty ecosystem.

Infrastructure-as-a-Code

Cel, gwiazda północy:

- zaangażowanie najszerzej grupy inżynierów;
- zaangażowanie: można zajrzeć do środka i kontrybuować;
- zaangażowanie: blisko konsumentów zasobów.

Infrastructure-as-a-Code

Niewłaściwe podejście na początku może prowadzić
do trudności później.

1/10: Remote state

Implementacje z locks:

- Azure storage ([docs](#));
- AWS S3 + DynamoDB;
- Google Cloud Storage.

1/10: Remote state

Nie zapomnij o:

- `.gitignore`
- `.dockerignore`

2/10: Struktura projektu

Mono-repo (per env/app):

```
infrastructure.git/
|- modules/
|   |- network
|   \- service-a/
|
|- production/
|   \- service-a/
|
...
```

2/10: Struktura projektu

Close to the application:

```
my_app.git/
|- app/
|- infra/
|   \- ...tf
|
... .
```

2/10: Struktura projektu

- copy&paste jest najłatwiejszym długiem do spłacenia;
- Nie musisz od pierwszego dnia mieć własne moduły;
- Wykorzystuj dostępne moduły z community.

3/10: Wiele mniejszych stack/stanów

- Najmniejszy możliwy, na przykład, per komponent;
- Unikamy czytania wartości z innych stanów;
- Bardzo łatwo o rigid infrastructure :/.

3/10: Rigid infrastructure?

- Wymagane wysokie uprawnienia do zarządzania,
- Twarde zależności między stanami,
- Zbyt (wcześnie) opinionated moduły,
- Cykliczne zależności.

3/10: Rozwiążanie

- odczytywanie stanu zasobów z chmury;
- **key/value store** (Azure Config / AWS SSM / HC Consul);
- Spacelift: **context** oraz **stack dependencies**.

5/10: Konwencje nazewnictwa

1. Używaj podkreśleń(_) jako seperatorów w nazwach.
Pisz nazwy małymi literami.
2. Nie dodawaj typu zasobu do nazw zasobów.
3. Zawsze używaj opisowych nazw dla zmiennych
wejściowych (variables) i wyjściowych (outputs).
Używaj opisów! (description)

5/10: Konwencje nazewnictwa

Więcej na:

- [Developer Terraform Docs](#),
- [Microsoft Azure](#),
- np., [terraform-null-label](#).

6/10: Pinning wersji dla wszystkiego

```
terraform {  
    required_version = ">= 0.12.26"  
  
    backend "s3" {}  
}
```

- Providers – minimum version
- Modules i wersje OpenTofu/Terraform - exact

6/10: Narzędzia

1. **tofu plan** - daje nam możliwość sprawdzenia zmian przed ich zaaplikowaniem,
2. Czyli możemy weryfikacje przesunąć w „lewo” - sprawdzamy co zostanie uruchomione i/lub zmienione jeszcze przed zmianą infrastruktury.

6/10: Narzędzia

Warto dodać do CI/CD albo git .pre-commit:

- Lintery: `tofu fmt & tflint`
- Bezpieczeństwo: `tfsec`, `kics` i `checkov`
- Estymacja kosztów: `tf-cost-estimation` & `infracost`

6/10: Narzędzia

Warto dodać do CI/CD albo git .pre-commit:

- Polityki: [conftest](#) & [OpenPolicyAgent](#):
 - Ostrzeż, że kasujemy lub odtwarzamy,
 - Etykiety na zasobach wymagane.

TACOS takie jak Spacelift, Env0, Scalr mają gotowe integracje

7/10: Continuous Deployment

1. Warto zacząć od Continuous Integration;
2. Kiedy demokratyzujemy dostęp i skalujemy,
Continuous Deployment jest niezbędne;

Opcje: generic CI, Atlantis (open source), Env0, Scalr,
Spacelift.io.

8/10: Demokratyzacja

To tylko kwestia czasu:

- Rośnie zespół,
- Trudno zatrudnić Platform czy System Cloud/DevOps inżynierów,
- Product teams.

8/10: Demokratyzacja

1. Zaczynamy od jednej osoby w zespole, np. infra champion,
2. Opcjonalnie - przydzielmy jedną osobę z zespołu DevOps / platformy (tymczasowo).

8/10: Demokratyzacja

Będzie łatwiej:

1. Jeśli nie mamy hard dependencies między stanami;
2. Nie potrzeba najwyższych uprawnień do postawienia każdego z serwisów.

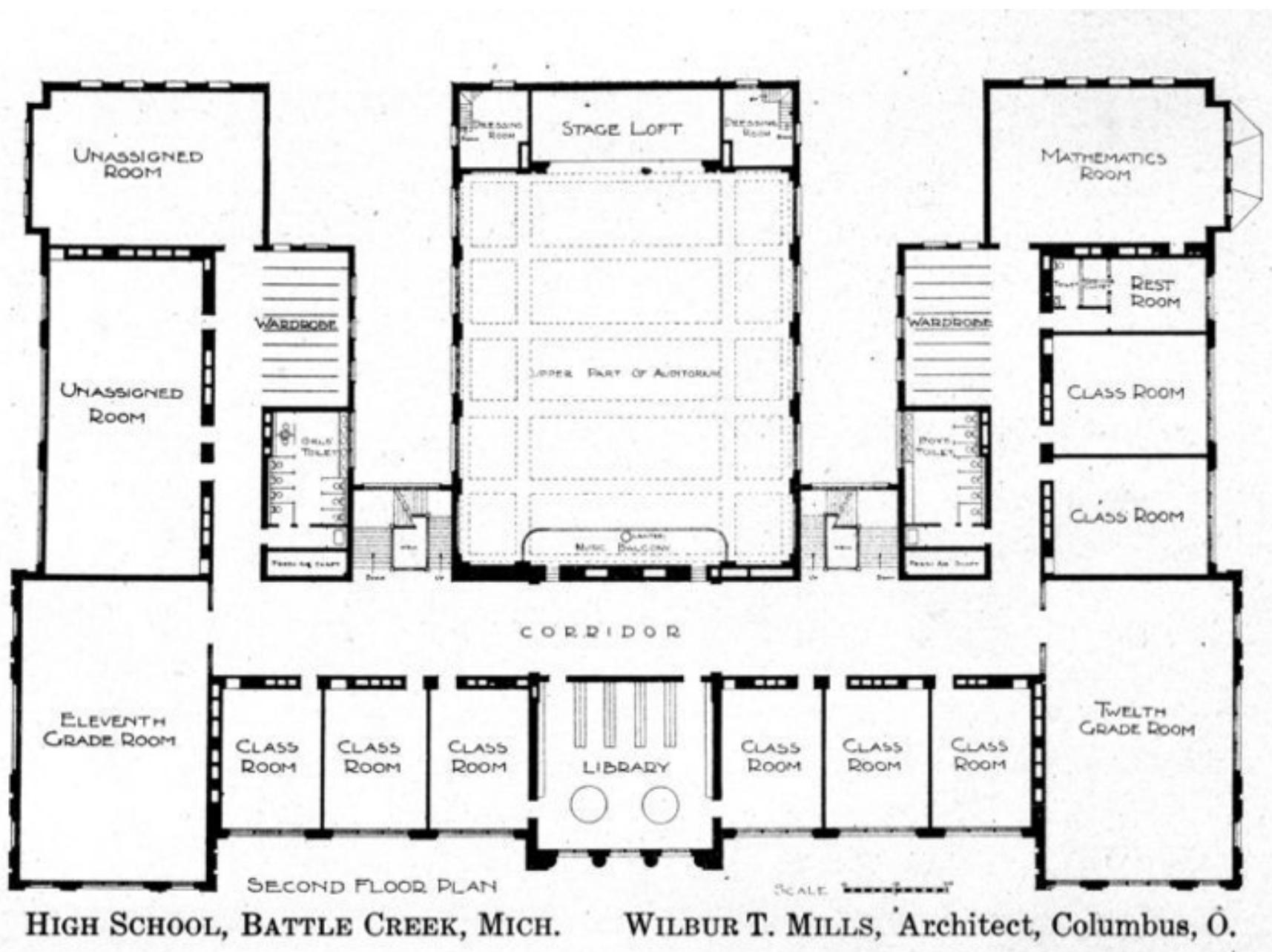
8/10: Demokratyzacja

Będzie łatwiej:

- Dobrze mieć CD na generycznym CI lub TACOS,
- Praca oparta o Pull Request'ach,
- W pierwszej iteracji, może być copy&paste planu w Pull Request,
- Więcej wspólnych (małych) modułów.

8/10: Demokratyzacja

- Zacząć od wspólnych konwencji... potem narzędzia/moduły



9/10: Skalowanie

- Template-y dla nowych serwisów,
- Współdzielenie najlepszych praktyk i polityk,
- Więcej wspólnych modułów,
- Tymczasowe środowiska,
- Drift-detection.

9/10: Skalowanie

Template-y dla nowych serwisów:

- Konwencje,
- No-black magic,
- Nie jestem fanem, custom YAML czy JSON,
- Możliwe uruchomienie lokalne.

9/10: Skalowanie

Drift detection:

- Zmiany w konsoli na chwilę,
- Upewnienie się, że rzadko dotykany moduł działa,
- Sprzątanie po eksperymentach.

9/10: Skalowanie

Mono repozytorium czy repozytorium per komponent:

- To zależy od firmy.

10/10: Metryki

- Pierwsze sukcesy łatwo odczuć,
- Później trudniej mierzyć postępy,
- Co więc mierzyć?

High delivery performance

- Lead Time
- Deployment frequency
- Mean time to Recovery
- Change Fail Percent



10/10: Metryki

Deployment frequency:

1. Łatwe do uchwycenia przy gitops,
2. Zakończenie pipeline lub po prostu merge do
brancha produkcyjnego.

10/10: Metryki

Lead time:

1. Od utworzenia Pull Request jako draft
2. Do wprowadzenia na produkcję (merge do master/prod)

10/10: Metryki

Obecnie używam:

- Google Colab Notebook,
- Exportery do Google BigQuery z: Github oraz ClickUp.

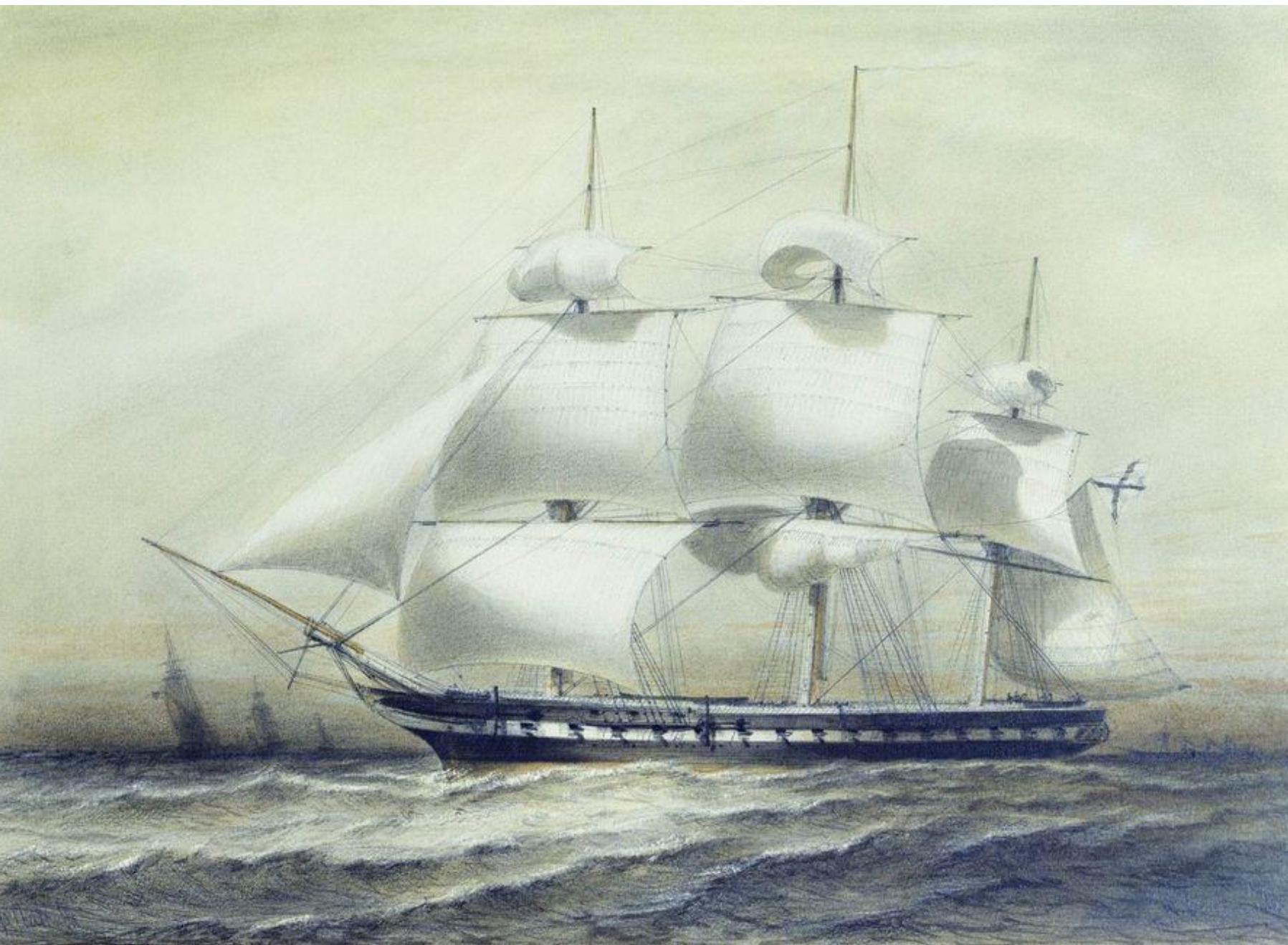
Podsumowanie

- Celem jest zaangażowanie najszerzej grupy inżynierów,
- Terraform/Opentofu jest najbardziej dojrzałą platformą,
- Większość tych praktyk, stosuje się do innych platform od AWS CloudFormation, CDK, CDK-TF, po Kubernetes.

Kubernetes and OpenTofu/Terraform

Często zadawane pytanie:

- OpenTofu/Terraform do postawienia K8S + ArgoCD, Crossplane, lub flux (last mile);
- Gitops na Kubernetesie przejmuje life-cycle aplikacji.



Dziękuję
za uwagę

github.com/wojciech12/talks

BACKUP SLIDES

Kubernetes and OpenTofu/Terraform

Często padające pytanie:

- OpenTofu/Terraform do postawienia K8S + ArgoCD, Crossplane, lub flux (last mile);
- Gitops na Kubernetesie przejmuje life-cycle aplikacji.

Rabbit holes everywhere...

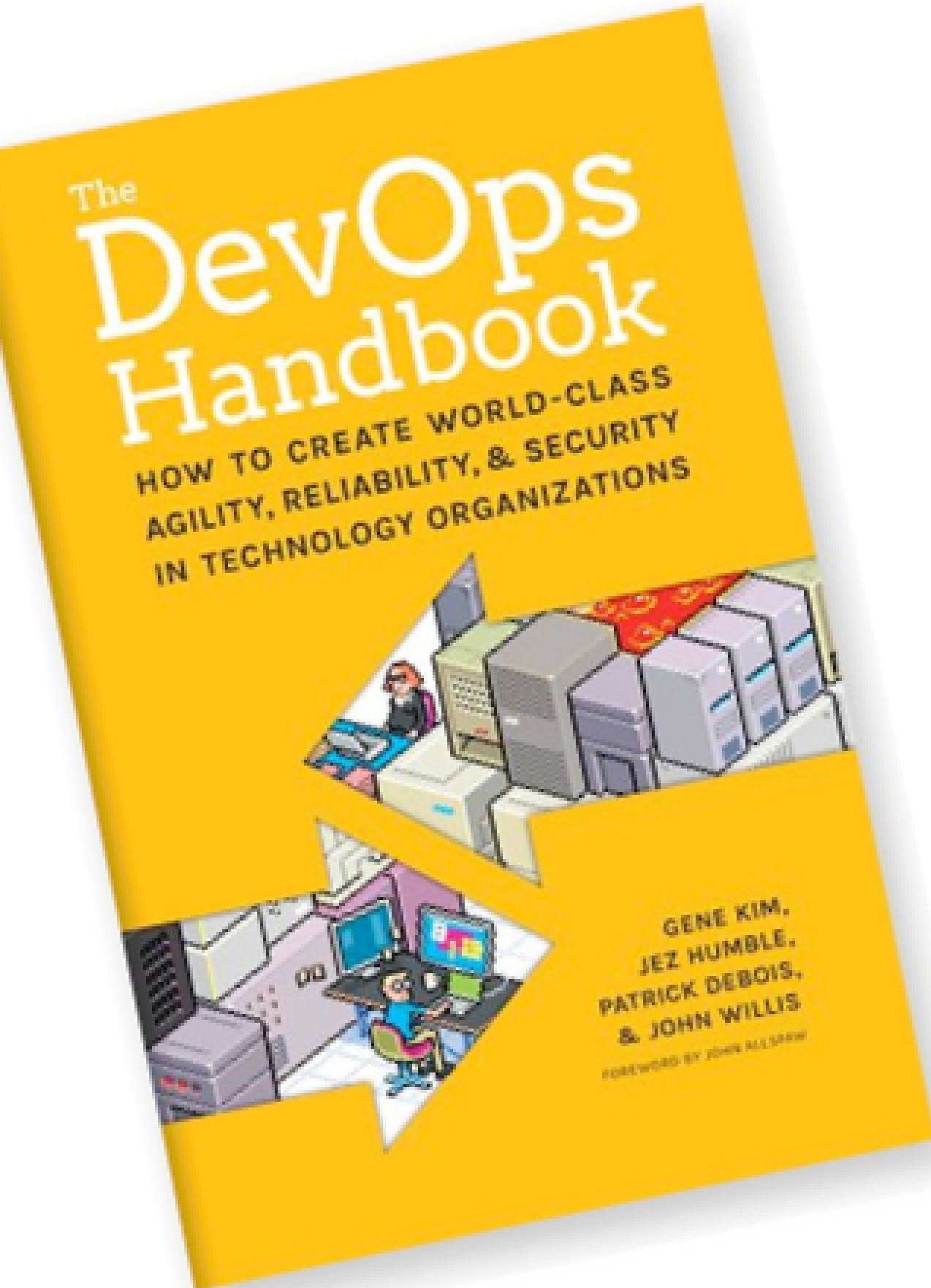
Approach:

- The iteration, decision, and deliver
- As soon as possible to get into the cycle Patch Patch Patch

Alternative take:

- Tracer bullet development,
- Lean v1/v2.

Recommended read

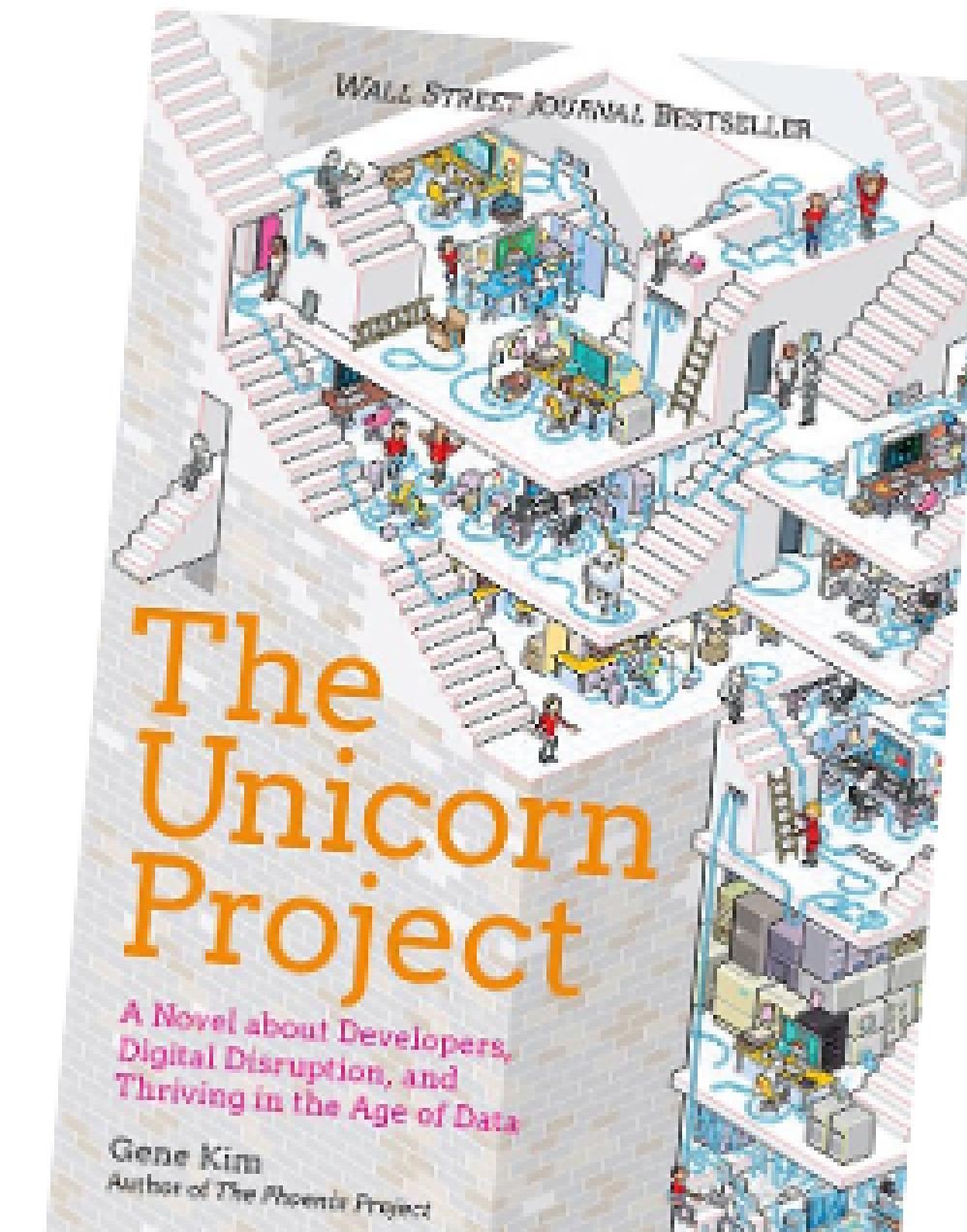


1

Concrete, do A, do B,
because C
(DevOps here as culture ☺)

2

The first 4 chapters,
the rest if you have time.



3

A story,
not as straight forward
as (1)



OODA

