Najlepsze praktyki dla Infrastructure-as-a-Code na przykładzie Terraforma

Wojciech Barczyński
VP of Engineering

Dlaczego Terraform?

```
provider "github" {
  owner = "wojciech12"
}

resource "github_repository" "my_repo" {
  name        = "tf_example"
  description = "My awesome TF repository"

  visibility = "public"
}
```

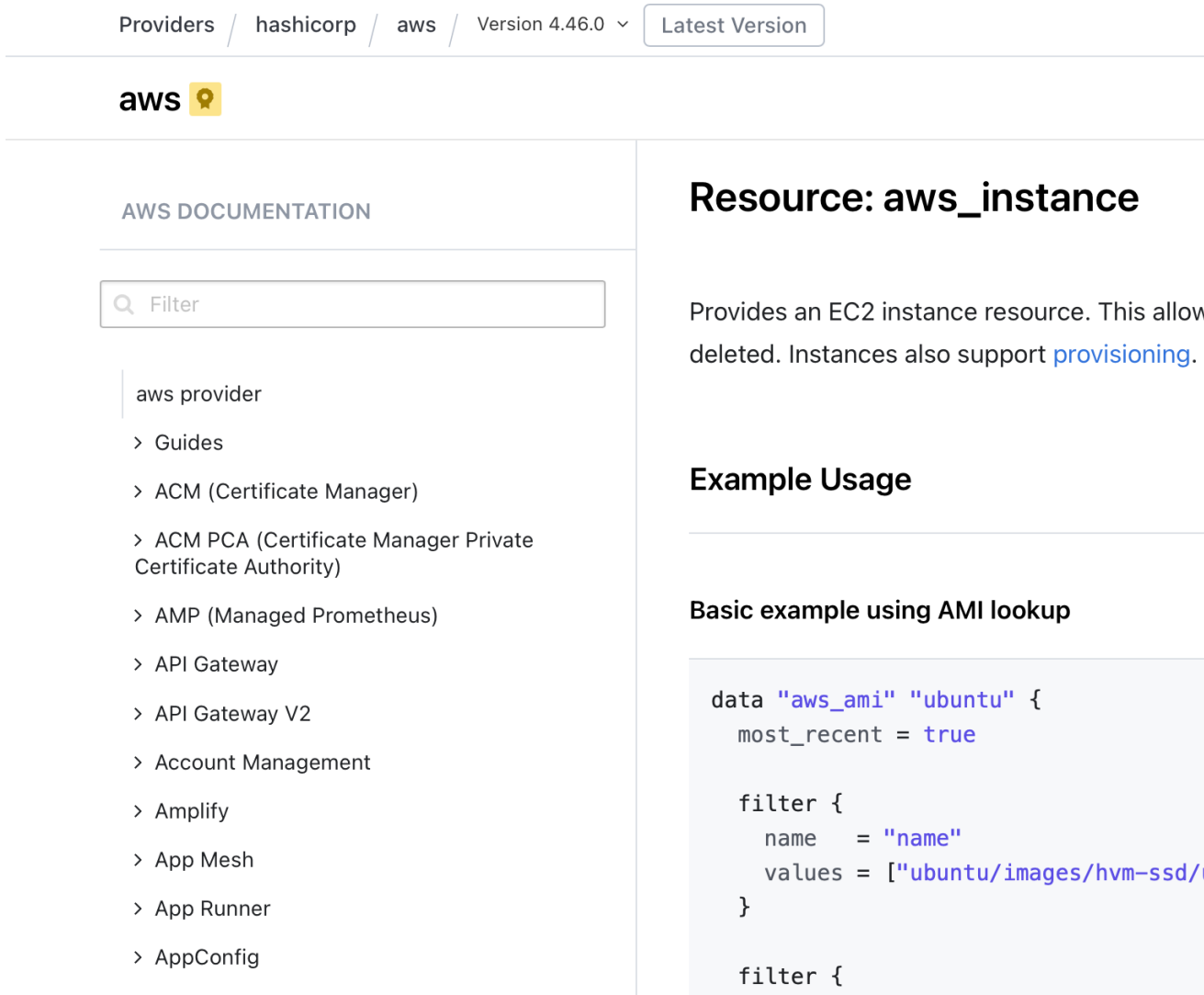
HashiCorp
Terraform

Dlaczego Terraform?

- `terraform plan` – co się zmieni;
- `terraform apply` – wykonanie planu.

Dlaczego Terraform?

- Deklaratywny język;
- Doskonała dokumentacja;
- Bogaty ecosystem;
- Śledzenie zmian, współpraca.



1. Remote state

- AWS s3 + DynamoDB;
- Google Cloud Storage;
- Azure storage.

```
terraform {
  backend "s3" {
    bucket = "mybucket"
    key    = "path/to/my/key"
    region = "us-east-1"
  }
}
```

Wystarczy po prostu copy&paste z dokumentacji.

1. Remote state

Nie zapomnij o:

- `.gitignore` ;
- `.dockerignore .`

2. Struktura projektu

```
tf.git/
 |- production/
 |    |- service-a/
 |    |- service-b/
 |    \- service-c/
 |
 |- staging/
 |   |- service-a/
 |   \-...
 \- dev
```

2. Struktura projektu

```
tf.git/
 |- modules/
 |   \- service-a/
 |
 |- production/
 |    \- service-a/
 |
...
```

2. Struktura projektu

- Pamiętaj Copy&Paste jest najprostszym długiem do spłacenia;
- Nie musisz od dnia zero, budować swoje własne moduły;
- Wykorzystuj dostępne moduły na terraform registry.

3. Wielkość stanu

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

3. Wielkość stanu

Co to jest rigid infrastructure?

- Wymagane wysokie uprawnienia wymaganymi dla danego komponentu;
- Twarde zależności między stanami.

4. Przekazywanie wartości

Jeśli nie za stanu to jak?

- AWS Systems Manager Parameter Store;
- Terraform Consul;
- Spacelift Context.

5. Naming conventions

- Podążaj za wskazówkami z dokumentacji Terraforma.

6. Pinning wersji dla wszystkiego

- Providers – minimum version
- Modules
- Wersji Terraforma

```
terraform {
  required_version = ">= 0.12.26"

  backend "s3" {}
}
```

7. Narzędzia

- `terraform plan`

  daje nam możliwość sprawdzenia zmian przed ich zaaplikowaniem

7. Narzędzia

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

1. Formatowanie: `terraform fmt`
2. `terraform validate`
3. Linting: tflint

7. Narzędzia

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

4. Bezpieczeństwo: tfsec & checkcov
5. Estymacja kosztów: tf-cost-estimation & infracost

7. Narzędzia

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

6. Polityki: conftest & opa

7. Narzędzia

- Warto zacząć od CI;
- Później, szczególnie kiedy musimy zdemokratyzować dostęp/skalować,
  Continuous deployment jest niezbędne;
- Opcje: Atlantis (open source), TF Cloud, and Spacelift.io

8. Demokratyzacja

To kwestia czasu:

- Rośnie zespół,
- Trudno zatrudnić Platform

  czy System Cloud/DevOps inżynierów,
- Product teams.

8. Demokratyzacja

1. Zaczynamy od jednej osoby w zespole,
2. Jeśli nie mamy hard dependencies między stanami, łatwiej będzie oddać część zarządzania komponentami;
3. Dobrze mieć przy najmniej CI w pierwszej iteracji.

8. Demokratyzacja

4. Praca oparta o Pull Requesty,
5. W pierwszej iteracji,

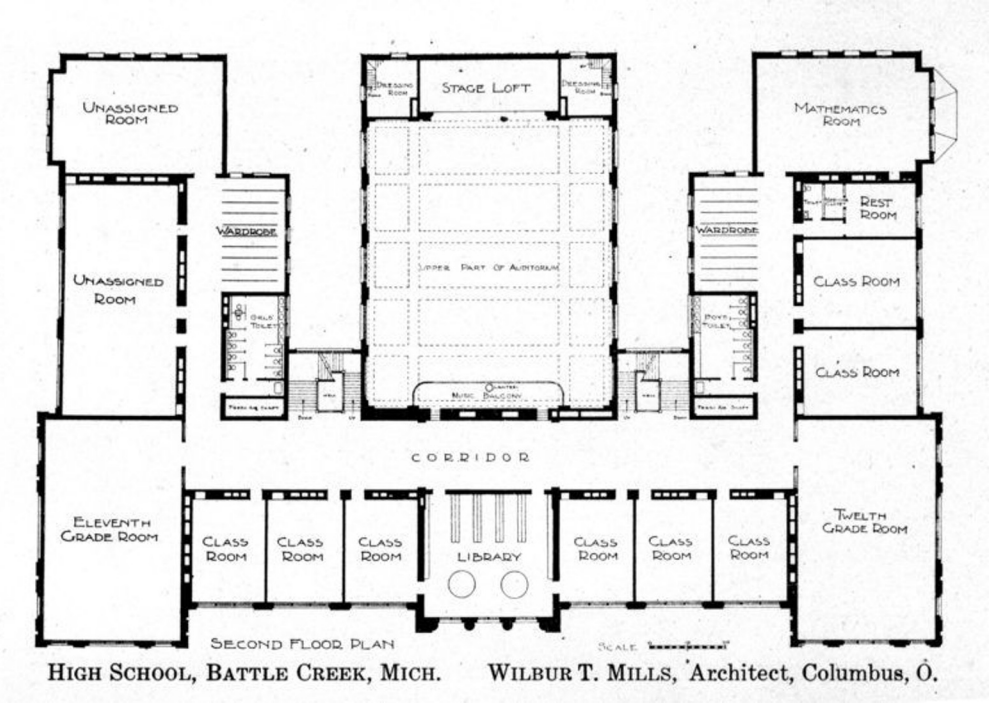
   może być copy&paste planu w Pull Request,
6. Więcej wspólnych modułów.

9. Skalowanie

9. Konwencje przed narzędziami

New repository:

- We want people to change the generated code
- We just ask for keeping the common conventions

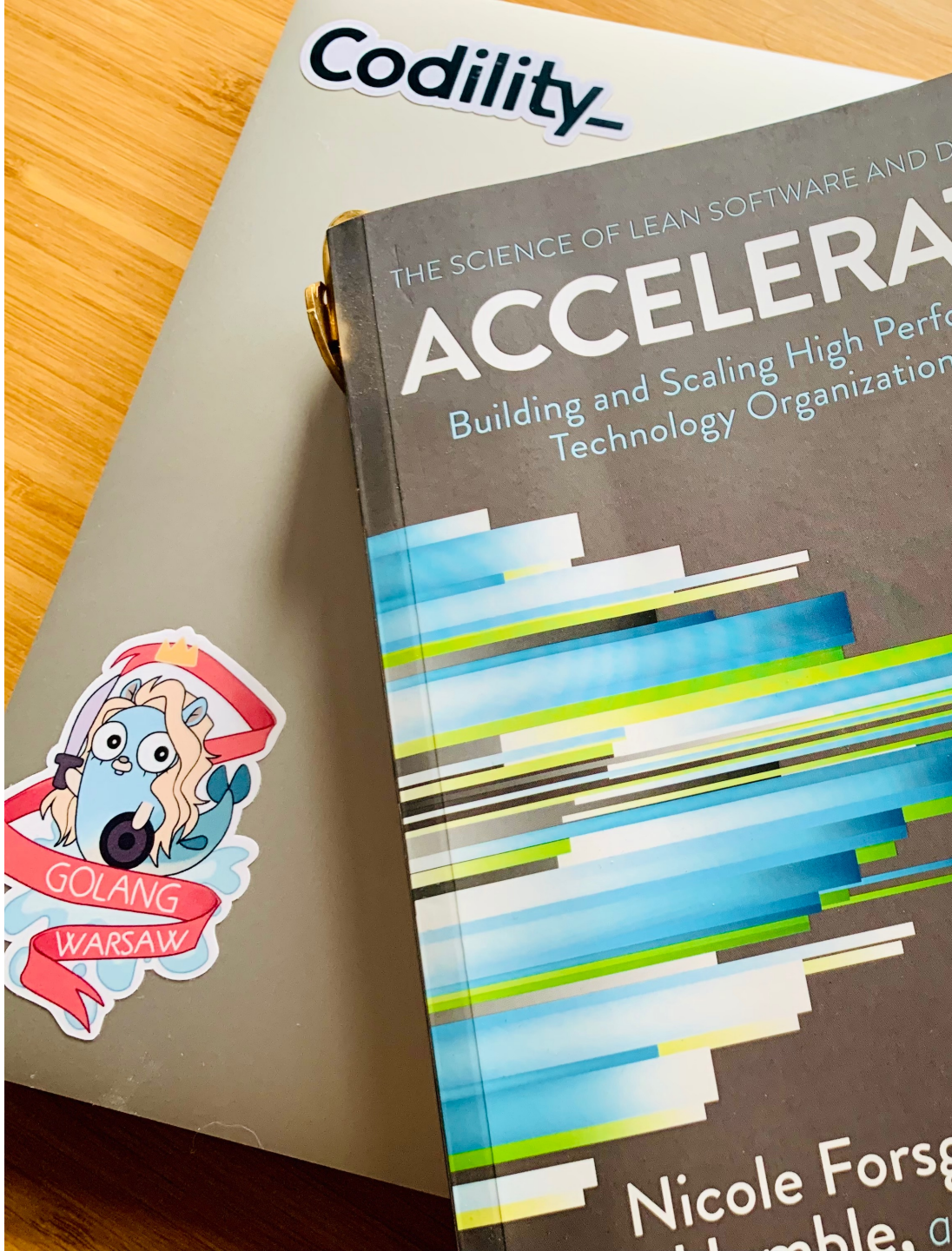


10. Metryki

1.

High performance

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



Questions?

https://www.flickr.com/photos/bruno_brujah/

spacelift

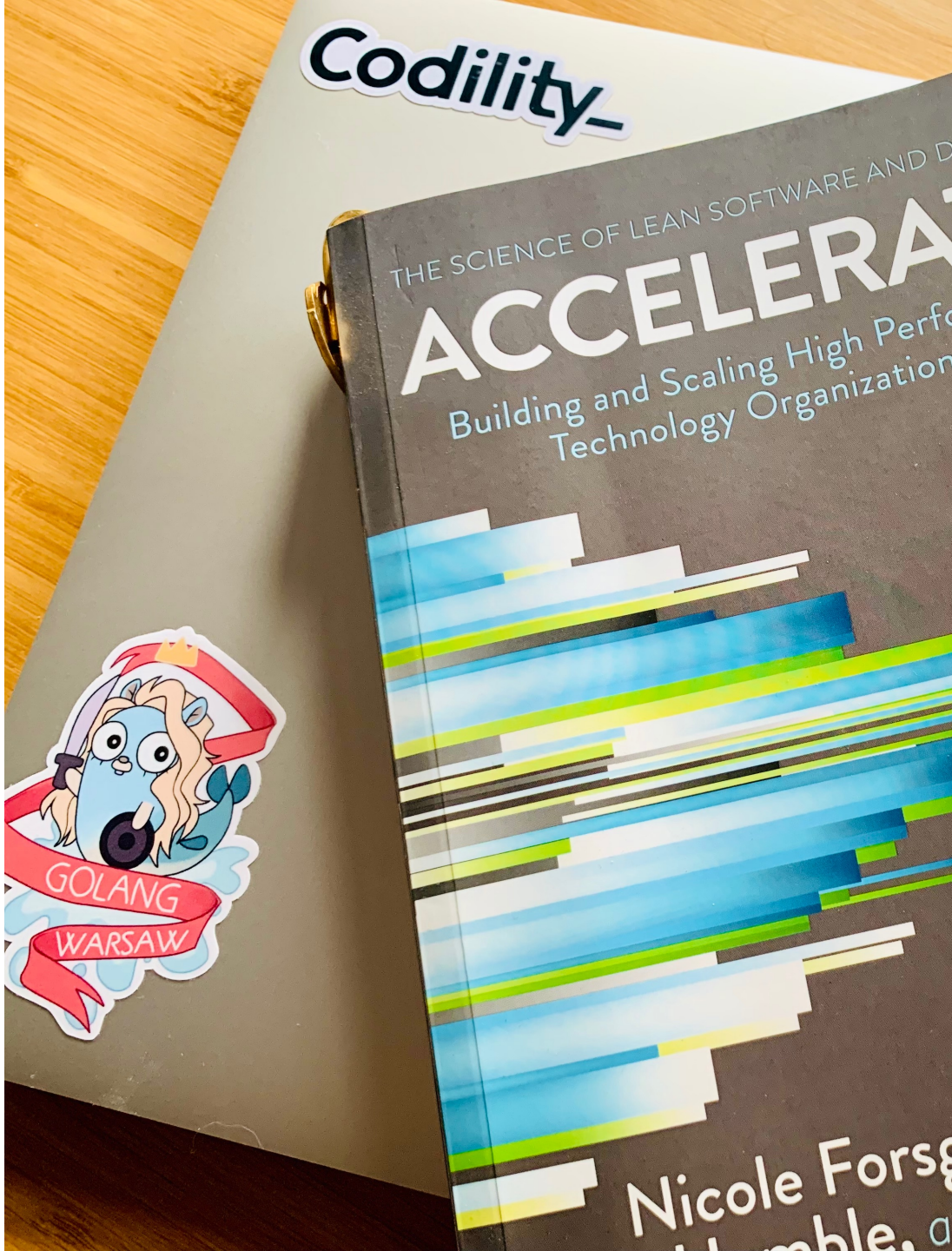
Hiring

Golang* developers passionate about system engineering, DevOps culture, or building tools.

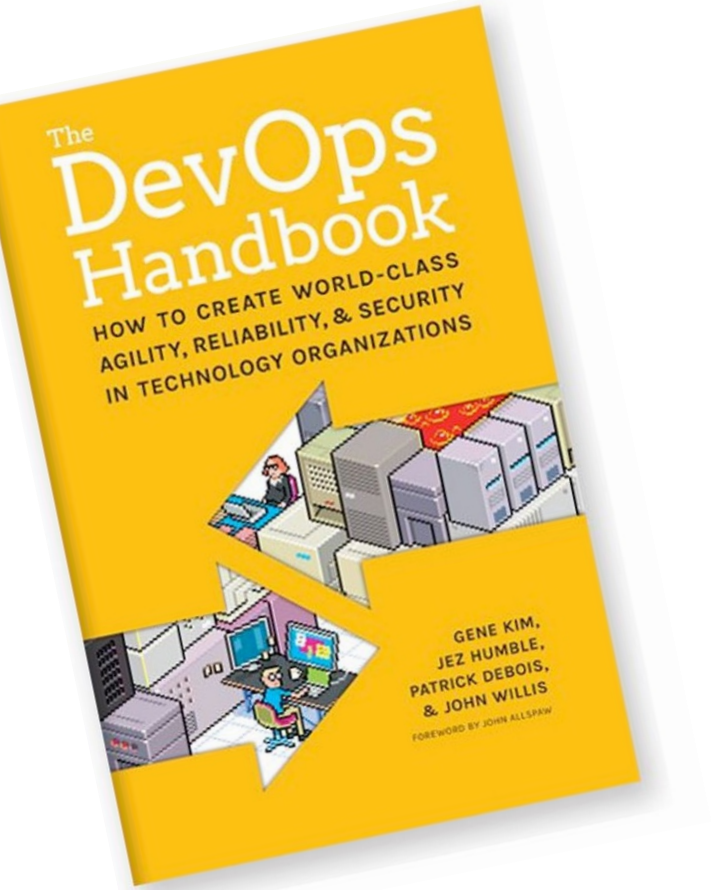
Backup

High performance

- Commercial success
- ...
- Satisfaction of the employees

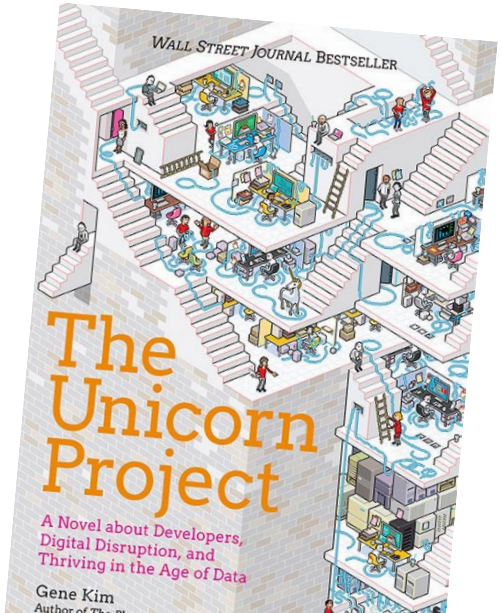


Recommended read



1 Concrete, do A, do B,
because C
(DevOps here as culture, not an admin ☺)

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



3 A story,
not as straight forward
as (1)

The way of Kubernetes

https://www.flickr.com/photos/bruno_brujah/

- In iterations
- Learn-as-you-go

Rabbit holes everywhere…

Approach:

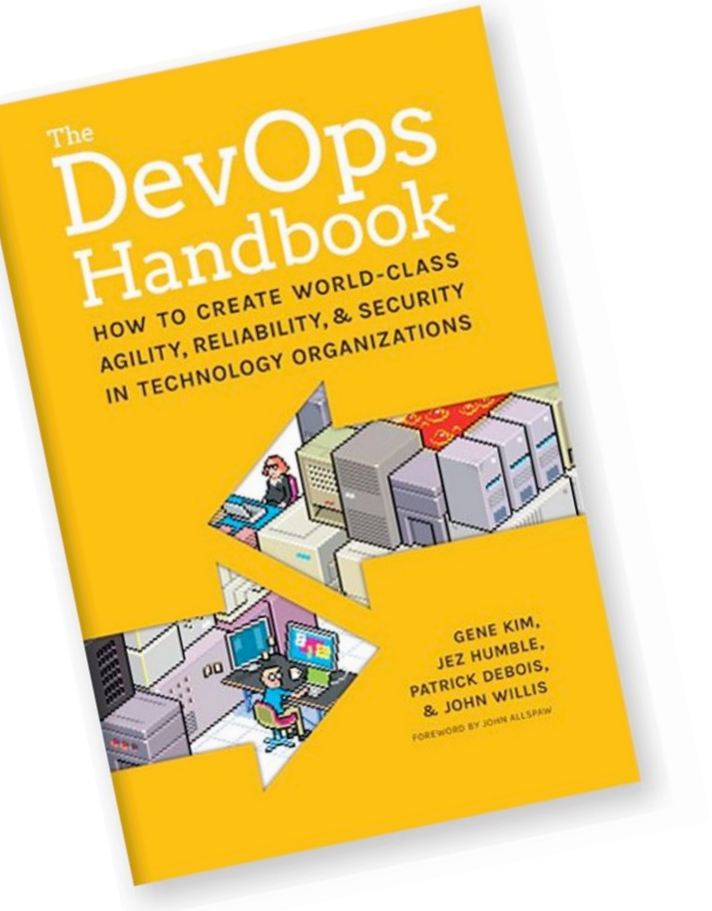
- The iteration, decision, deliver
- As soon as possible
  to get into the cycle Patch Patch Patch

Alternative take:

- Lean v1/v2*

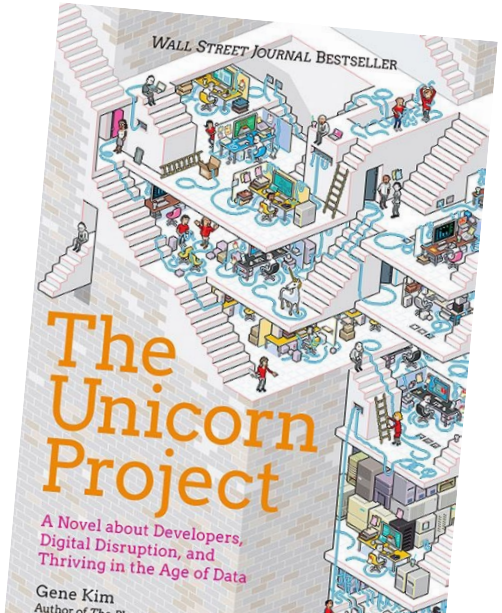
* https://katemats.com/blog/lean-software-development-build-v1s-and-v2s

Recommended read



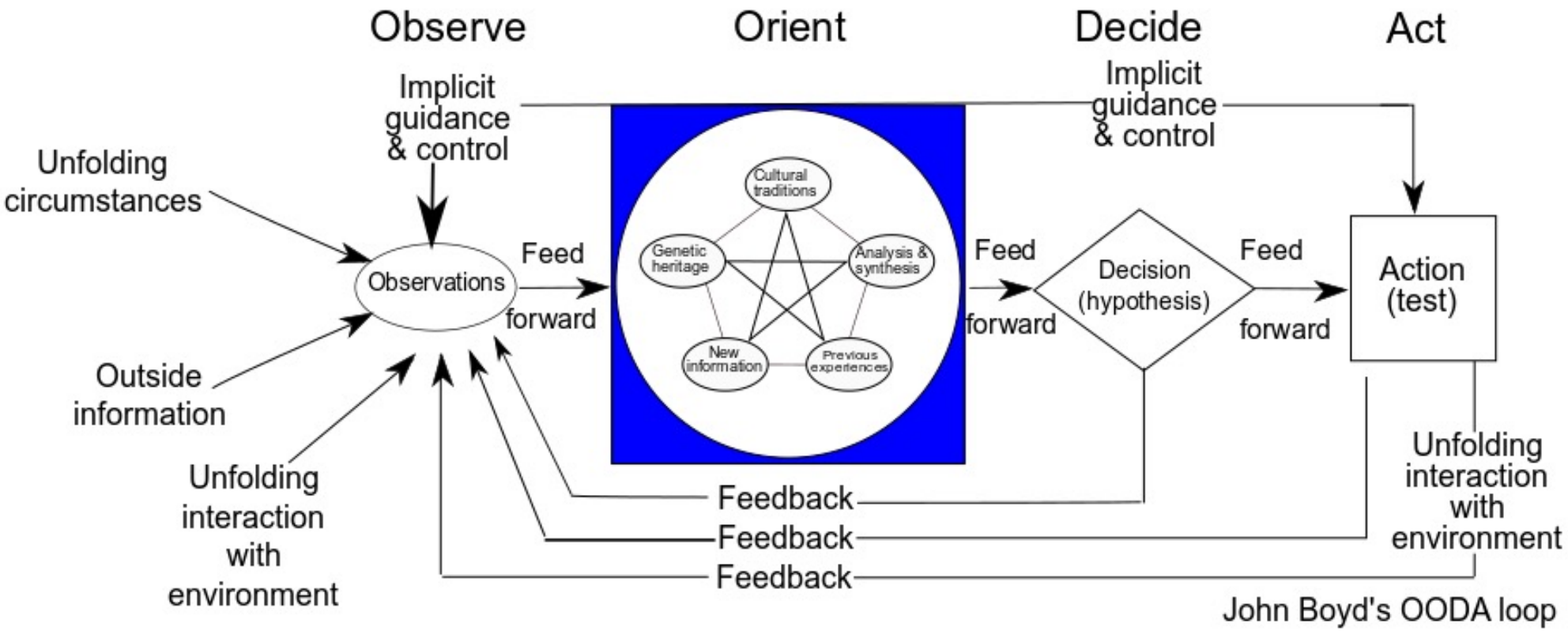
1 Concrete, do A, do B,
because C
(DevOps here as culture, not an admin ☺)

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



3 A story,
not as straight forward
as (1)

OODA



https://upload.wikimedia.org/wikipedia/commons/3/3a/OODA.Boyd.svg