

10 najlepszych praktyk dla Infrastructure-as-a-Code na przykładzie Terraforma

Wojciech Barczyński
VP of Engineering



Najlepsze praktyki

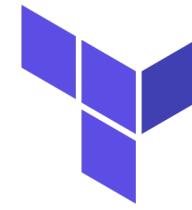
- Celem jest zaangażowanie najszerzej grupy inżynierów w współtworzenie infrastruktury;
- Początkowe kroki mogą decydować o trudności otworzenia zasobów chmurowych.

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.

The screenshot shows a section of the HashiCorp Terraform documentation for the AWS provider. At the top, there's a navigation bar with 'Providers' (selected), 'hashicorp', 'aws' (selected), 'Version 4.46.0', and a 'Latest Version' button. Below the navigation is the title 'aws' with a yellow location pin icon. To the right, under the heading 'Resource: aws_instance', it says: 'Provides an EC2 instance resource. This allow... deleted. Instances also support provisioning.' Below this, under 'Example Usage', is a code snippet for 'Basic example using AMI lookup':

```
data "aws_ami" "ubuntu" {
  most_recent = true

  filter {
    name    = "name"
    values = ["ubuntu/images/hvm-ssd/"]
  }

  filter {
```

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

- Zacznij od mono-repo,
• Env -> Komponent
- ```
tf.git/
|- production/
| |- service-a/
| |- service-b/
| \- service-c/
|
|- staging/
| |- service-a/
|
| \- ...
\- dev
```

## 2. Struktura projektu

```
tf.git/
| - modules/
| | - network
| \- 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 rigid infrastructure.

### 3. Wielkość stanu

Co to jest rigid infrastructure?

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

# 4. Jak uniknąć rigid infrastructure?

Jeśli nie za stanu to jak?

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

# 5. Naming conventions

- Podążaj za wskazówkami z [dokumentacji Terraformu](#).

# 6. Pinning wersji dla wszystkiego

- Providers – minimum version
- Exact:
  - Modules
  - Wersji Terraforma

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

 backend "s3" {}
}
```

## 7. Narzędzia

- terraform plan
  - daje nam możliwość sprawdzenia zmian przed ich zaaplikowaniem,
- Czyli możemy wiele weryfikacji przesunąć w „lewo”.

# 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](#):

- [Ostrzeż, że kasujemy lub odtwarzamy](#),
- [Etykiety na zasobach wymagane](#).

# 7. Narzędzia

- Warto zacząć od Continuous Integration;
- Kiedy demokratyzujemy dostęp i skalujemy,  
Continuous deployment jest niezbędne;
- Opcje:  
generic CI, [Atlantis](#) (open source), TF Cloud, i Spacelift.io

## 8. Demokratyzacja

To tylko kwestia czasu:

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

## 8. Demokratyzacja

1. Zaczynamy od jednej osoby w zespole,
2. Infra/... champion,
3. Opcjonalnie - przydzieleniem jeden osoby  
z zespołu devOps / platformy (tymczasowo).

## 8. Demokratyzacja

Będzie łatwiej:

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

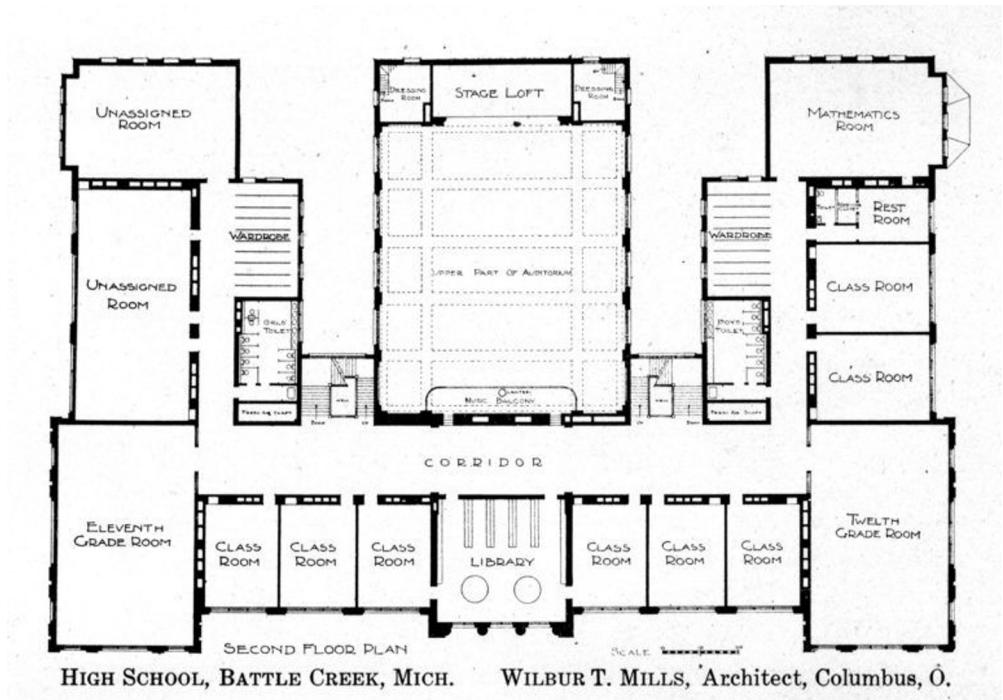
## 8. Demokratyzacja

4. Dobrze mieć CI.
5. Praca oparta o Pull Requesty,
6. W pierwszej iteracji,  
może być copy&paste planu w Pull Request,
7. Więcej wspólnych (małych) modułów.

# 8. Konwencje przed narzędziami

New repository:

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



## 9. Skalowanie

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

# 9. Skalowanie

Template-y dla nowych serwisów:

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

# 9. Skalowanie

Drift detection:

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

# 9. Skalowanie

Mono repo czy per komponent:

- To zależy od firmy.

# 10. Metryki

1. Pierwsze sukcesy łatwo odczuć,
2. Później trudniej mierzyć podstępy,
3. Co więc mierzyć?

# High perf teams

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



# 10. Metryki

Deployment frequency:

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

# 10. Metryki

Lead time:

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

# 10. Metryki

Obecnie używam:

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

# Podsumowanie

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

# Terraform and Kubernetes

Często padające pytanie:

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

# Pytania?

---



[https://www.flickr.com/photos/bruno\\_brujah/](https://www.flickr.com/photos/bruno_brujah/)



## Hiring

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

The image consists of two parts. The top part is a screenshot of the Spacelift website homepage. It features the Spacelift logo at the top left, followed by a navigation bar with links for 'Use Cases', 'Documentation', 'Pricing', and 'Blog'. On the right side of the navigation bar are 'Login', 'Book a demo' (in a white button), and 'Get started' (in a purple button). The main content area has a dark background with large, bold text: 'Collaborative Infrastructure' in light blue and 'For Modern Software Teams' in white. Below this text is a description: 'Spacelift is a sophisticated CI/CD platform for Terraform, CloudFormation, Pulumi, and Kubernetes'. At the bottom of this section is a purple 'Get started' button. The bottom part of the image is a screenshot of the Spacelift user interface. It shows a sidebar with icons for stacks, history, and workspace. A central card displays a stack update: 'Stacks • Stack for manual-gitlab-test changed • Update main.tf' with a status of 'FINISHED'. The update details show a commit hash '0.15.3' from 'spacelift-cr-gh/workspace', a file 'main', a timestamp '9 days ago', and a note 'State managed by Spacelift'. Below the card is a text input field 'Share your thoughts' and a 'Comment' button. A comment bubble from a user named 'tommy90' is visible, with the text 'Any thoughts?' and a timestamp '2 minutes ago'.

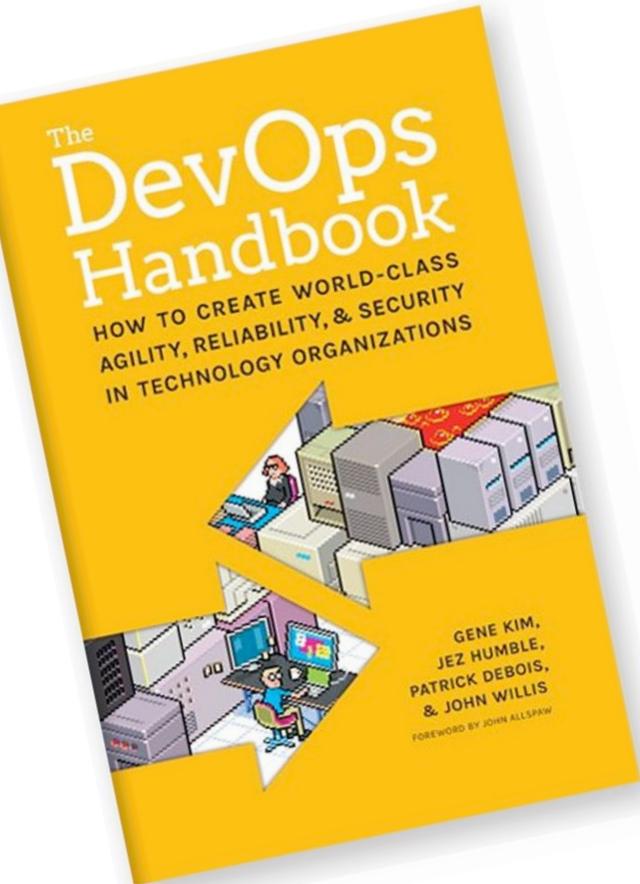
// Infrastructure-as-Code was only the beginning

Stacks • Stack for manual-gitlab-test changed • Update main.tf  
FINISHED  
0.15.3 spacelift-cr-gh/workspace main 9 days ago State managed by Spacelift  
History Add +0 Change ~3 Delete -0

tommy90 2 minutes ago  
Any thoughts?

# Backup

# Recommended read



1

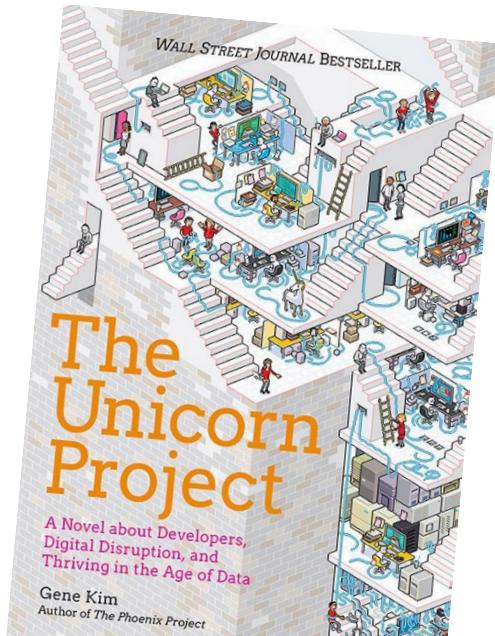
Concrete, do A, do B,  
because C  
(DevOps here as culture ☺ )

2

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

3

A story,  
not as straight forward  
as (1)



# High performance

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



# Rabbit holes everywhere...

Approach:

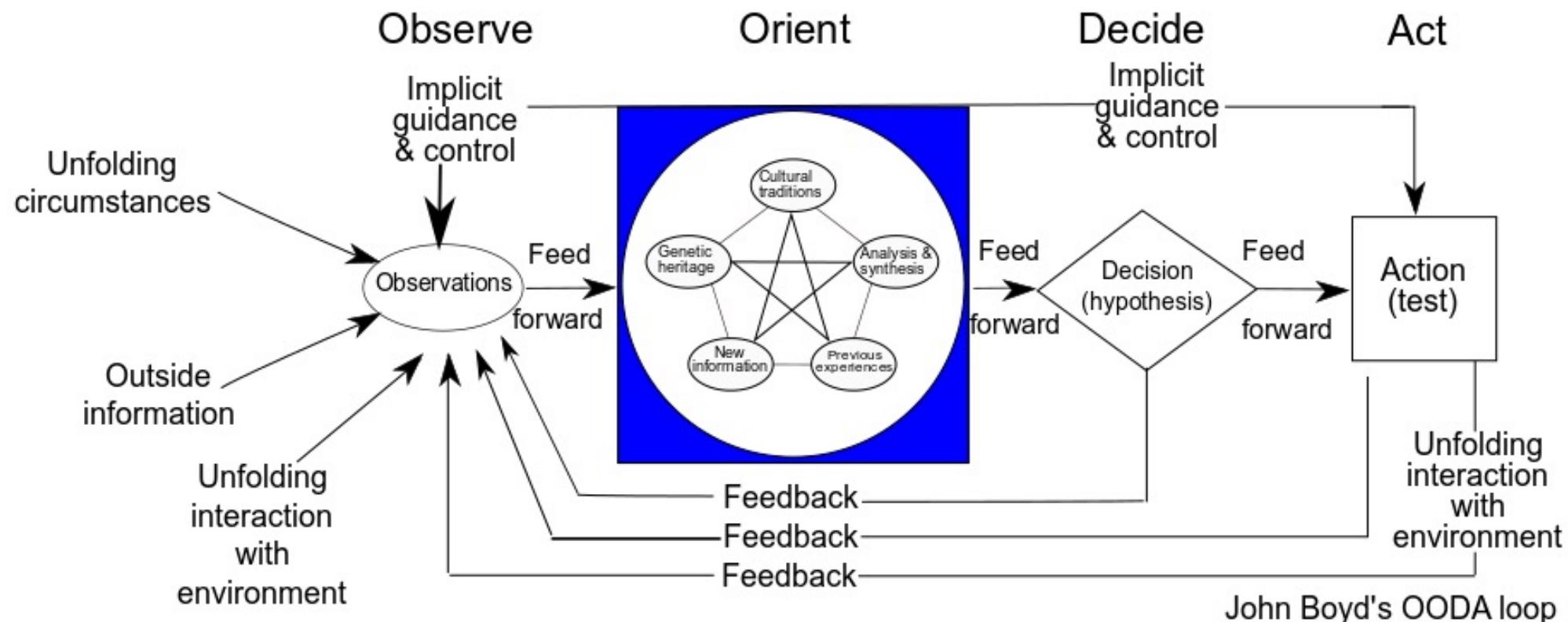
- The iteration, decision, deliver
- As soon as possible  
to get into the cycle Patch Patch Patch

Alternative take:

- Lean v1/v2\*

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

# OODA



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