

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

Wojciech Barczyński
VP of Engineering



Najlepsze praktyki

- Ostatecznym celem jest zaangażowanie najszerzej grupy inżynierów;
- Początkowe kroki mogą decydować o trudności otworzenia zasobów chmurowych,
- 10 praktyk.

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?

1. terraform plan – co się zmieni;
2. terraform apply – wykonanie planu;
3. stan – plik opisujący jak wyglądają zasoby chmurowe.

Dlaczego Terraform?

- Doskonała dokumentacja;
- Bogaty ecosystem;
- Łatwo wspólnie pracować oraz śledzić zmiany.

The screenshot shows a screenshot of the HashiCorp Terraform provider documentation for AWS. At the top, there's a navigation bar with 'Providers' (selected), 'hashicorp', 'aws', 'Version 4.46.0', and a 'Latest Version' button. Below the navigation is a search bar with the text 'aws'. A sidebar on the left lists various AWS services and providers. The main content area is titled 'Resource: aws_instance' and describes it as providing an EC2 instance resource. It includes sections for 'Example Usage' and 'Basic example using AMI lookup', showing code snippets for Terraform configurations.

Providers / hashicorp / aws / Version 4.46.0 ▾ Latest Version

aws

AWS DOCUMENTATION

Filter

aws provider

- > Guides
- > ACM (Certificate Manager)
- > ACM PCA (Certificate Manager Private Certificate Authority)
- > AMP (Managed Prometheus)
- > API Gateway
- > API Gateway V2
- > Account Management
- > Amplify
- > App Mesh
- > App Runner
- > AppConfig

Resource: aws_instance

Provides an EC2 instance resource. This allow deleted. Instances also support provisioning.

Example Usage

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 najłatwiejszym 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.

Przeniesienie zależności do runtime.

# 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: confest & 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.

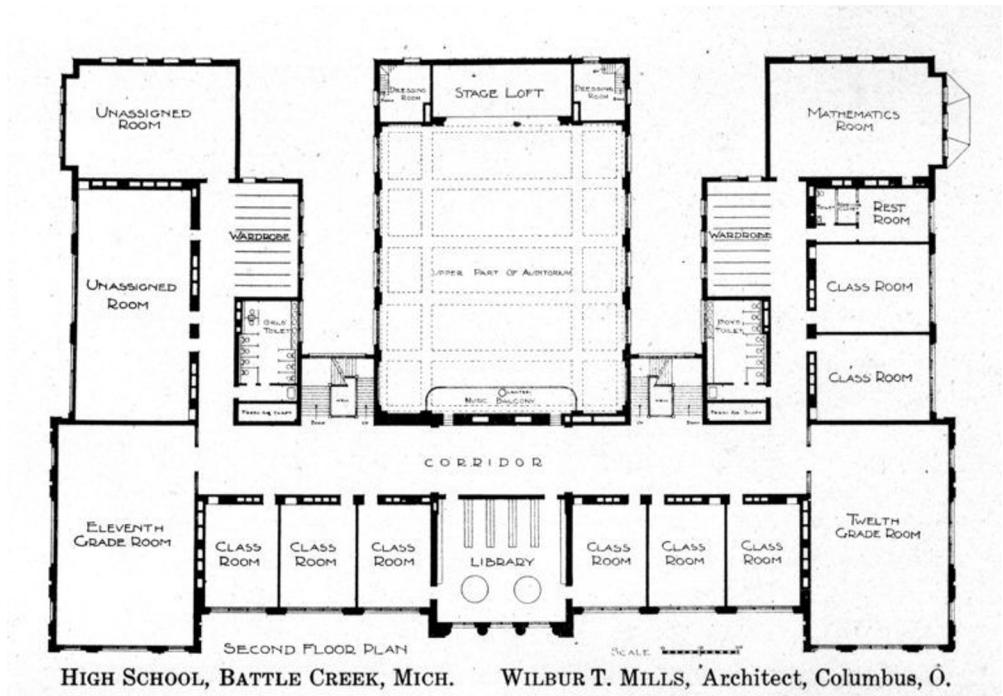
## 8. Demokratyzacja

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,
- Więcej wspólnych modułów,
- 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

Wspólne moduły:

- Wymuszanie aktualizacji modułów,
- Zarządzanie modułami przez prywatne repozytorium.

# 9. Skalowanie

Mono repozytorium 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, CDK-TF, 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 to '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 light-colored text. The title 'Collaborative Infrastructure For Modern Software Teams' is displayed in large, bold, blue and white font. Below the title is a subtitle: '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 'Stacks' sidebar with icons for 'Stacks', 'Workspaces', and 'Jobs'. A specific stack named 'Stack for manual-gitlab-test changed' is selected, showing a 'Update main.tf' history entry. The entry details include a commit hash (0.15.3), workspace (spacelift-cr-gh/workspace), branch (main), commit ID (88da430), and timestamp (9 days ago). The status is 'State managed by Spacelift'. Below the history is a 'Share your thoughts' input field and a 'Comment' section with a message from 'tommy90' (2 minutes ago) asking 'Any thoughts?'. There are also 'Add comment' and 'Edit' buttons.

// Infrastructure-as-Code was only the beginning

Stacks • Stack for manual-gitlab-test changed • Update main.tf

← Update main.tf FINISHED

0.15.3 spacelift-cr-gh/workspace main 88da430 9 days ago State managed by Spacelift

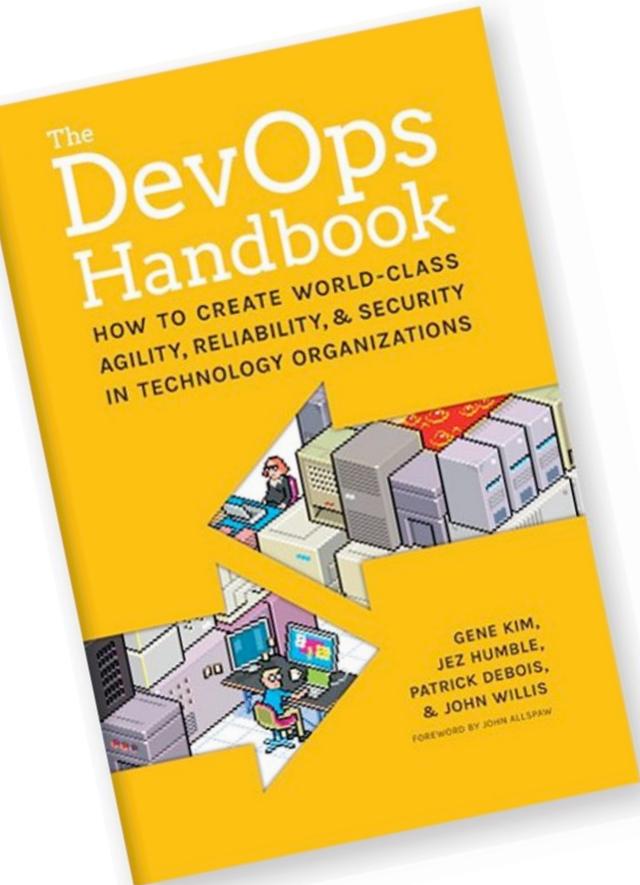
History Add +0 Change ~3 Delete -0

tommy90 2 minutes ago Any thoughts?

Add comment

# Backup

# Recommended read



1

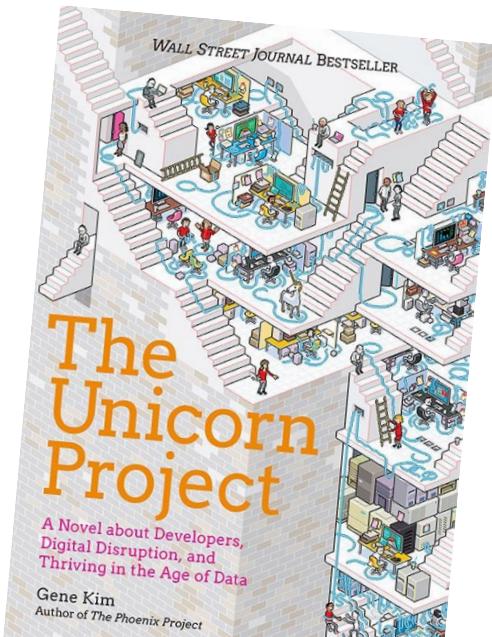
Concrete, do A, do B,  
because C  
(DevOps here as culture ☺ )

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:

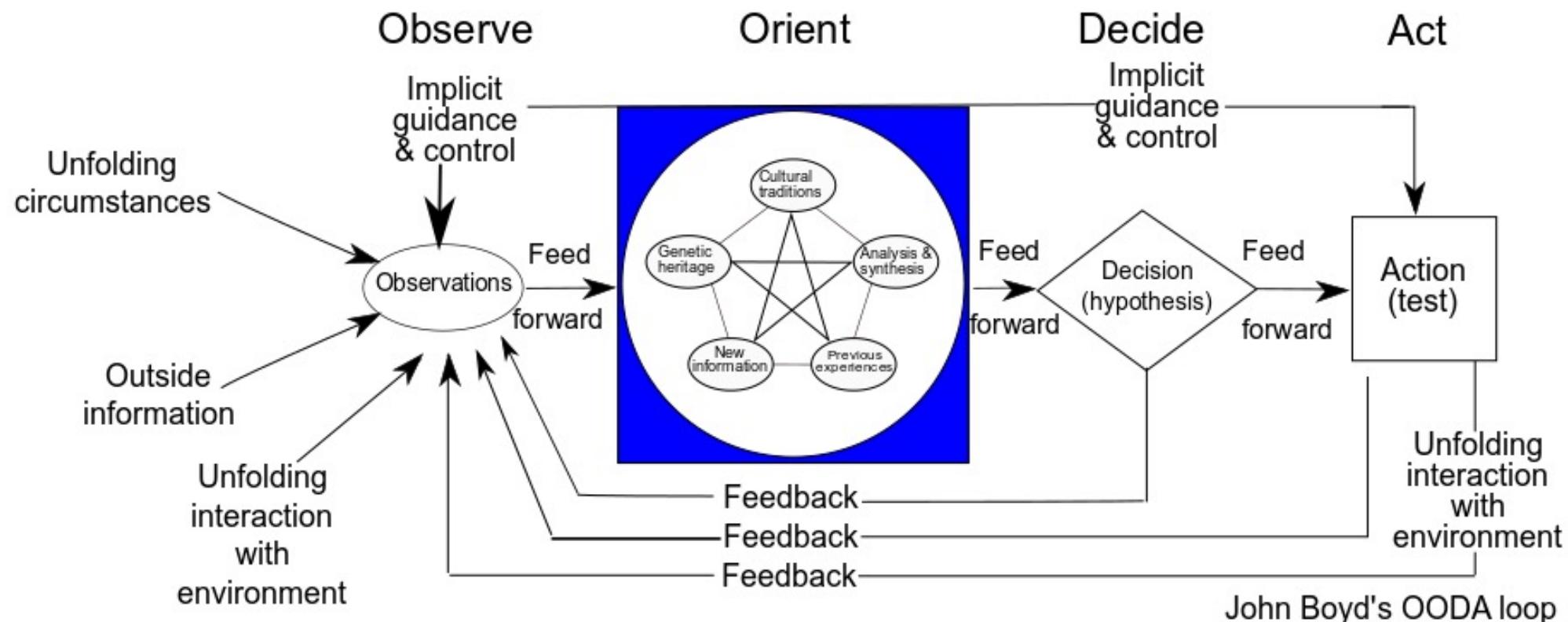
- The iteration, decision, deliver
- As soon as possible  
to get into the cycle Patch Patch Patch

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>