ensō

Terraforming Azure

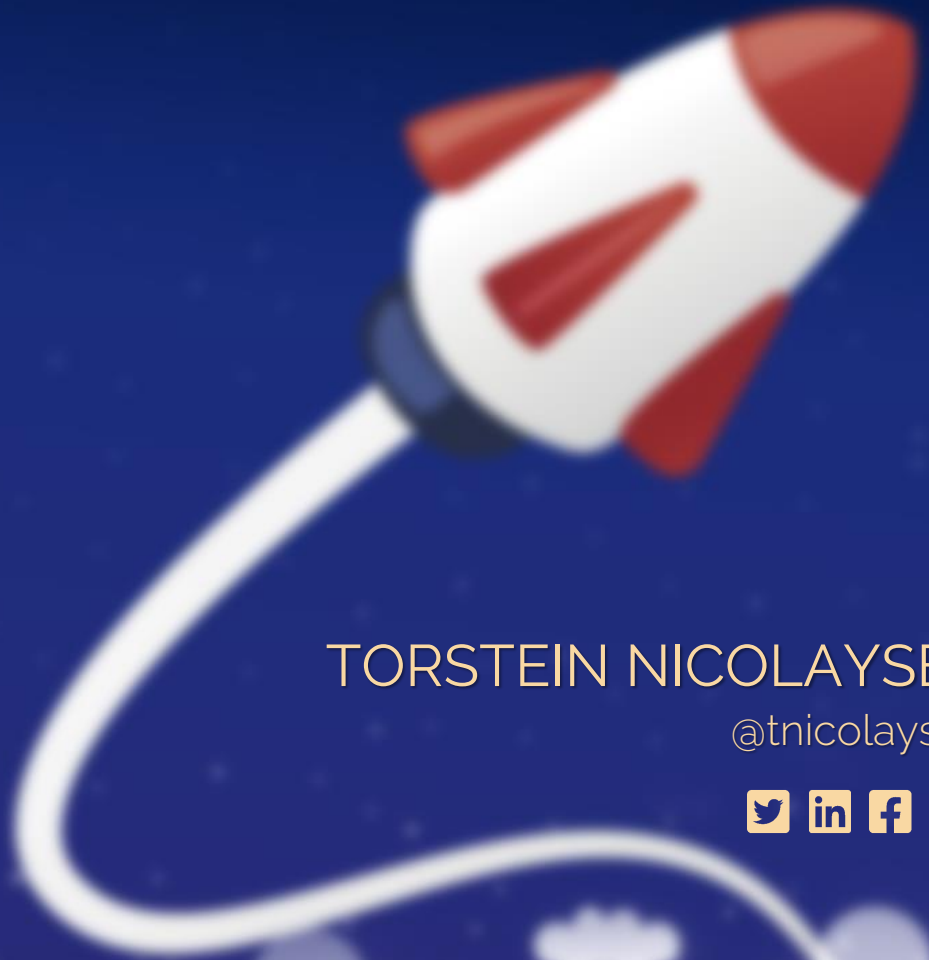
TORSTEIN NICOLAYSEN
@tnicolaysen

NDC OSLO
2019-06-19

Image by Bruno DE LIMA from Pixabay

Agenda

- Provide some context
- Azure Resource Manager (ARM)
- Terraform
- Experiences



TORSTEIN NICOLAYSEN
@tnicolaysen

DEVOPS

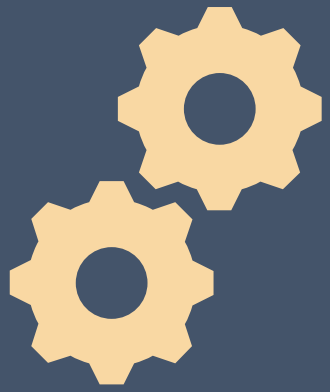
Deliver software faster
and more reliable

CULTURE

MEASUREMENT

SHARING

AUTOMATION



Infrastructure as Code

PROVISIONING TOOLS

SERVER TEMPLATING TOOLS

AD HOC SCRIPTS

CONFIGURATION MANAGEMENT TOOLS

Why DevOps and IAC?

- Reduce time to market
- Reduce defects and time to fix
- Documentation
- Transparency
- Increase autonomy
- Easier disaster recovery
- Log of changes (VCS)
- Competitive edge (for now)

2016 STATE OF DEVOPS REPORT

Organizations that use DevOps practices, such as IAC, deploy 200 times more frequently, recover from failures 24 times faster, and have lead times that are 2,555 times lower.

https://puppet.com/resources/whitepaper/2016-state-of-devops-report

IMPERATIVE → **HOW**

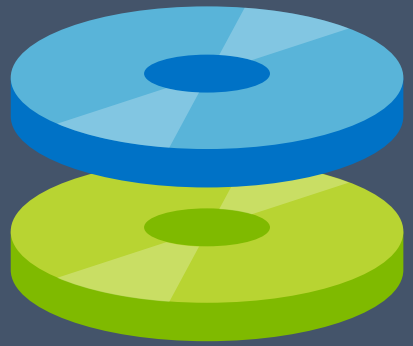
Describe step by step how to get what you want

DECLARATIVE → **WHAT**

Describe what you want, not how to do it

Resources in Azure

"A thing you can manage"



Azure

An application

APP SERVER


Virtual Machine


Virtual NIC


Virtual Network

Network sec. group


Storage Account

SECRETS


Key Vault


Storage Account
(for diagnostics and audit logs)

DATABASE

SQL Server


Storage Account
(for backup, diagnostics, audits etc.)

Azure Resource Manager

ARM

DEMO|1

Provision a storage account in Azure using ARM

ARM: The good

- Tried and tested
- Let's you manage all resource types in Azure (high coverage)
- Many examples to learn from
- Fairly robust
- Tooling support
- Can retrieve secrets from Key Vault when deploying*

ARM: The bad

- JSON is not optimal for ARM
  - No easy way to add comments
  - A lot of noise
  - Functions within strings
- Grows hard to maintain and read
- Inconsistent
- Hard to reuse components
- Limited support for configuring resources
  - E.g. no way to add a secret to a Key Vault

ARM: The ugly

- Bad error messages
- Resource API versioning is weird and lacking documentation
  - Changing API version changes the schema
- Simple things can be hard to achieve

ARM: The ugly

Requires hardcoded resource IDs to use secrets from Key Vault

```
"adminPassword": {
    "reference": {
      "keyVault": {
        "id": "/subscriptions/<subscription-id>
              /resourceGroups/<rg-name>
              /providers/Microsoft.KeyVault
              /vaults/<vault-name>"
      },
      "secretName": "vm-admin-password"
    }
  }
```

HashiCorp

Terraform

DEMO|2

The same thing we did with ARM

Terraform on a slide

- Created by HashiCorp
  - Vagrant, Consul, Vault, Nomad, Packer
- Open Source – written in Go
- Let's you manage any resource
- Supports many platforms – not only Azure
- Has enterprise options to provide a more complete experience
- Declarative templates in HashiCorp Configuration Language (HCL)

HCL

```
# An AMI
variable "ami" {
  description = "the AMI to use"
}

/* A multi
line comment. */
resource "aws_instance" "web" {
  ami               = var.ami
  count             = 2
  source_dest_check = false

  connection {
    user = "root"
  }
}
```

JSON

```
{
  "variable": {
    "ami": {
      "description": "the AMI to use"
    }
  },
  "resource": {
    "aws_instance": {
      "web": {
        "ami": "${var.ami}",
        "count": 2,
        "source_dest_check": false,
        "connection": {
          "user": "root"
        }
      }
    }
  }
}
```

HCL

```
# An AMI
variable "ami" {
  description = "the AMI to use"
}

/* A multi
line comment. */
resource "aws_instance" "web" {
  ami               = var.ami
  count             = 2
  source_dest_check = false

  connection {
    user = "root"
  }
}
```

JSON

```
{
  "variable": {
    "ami": {
      "description": "the AMI to use"
    }
  },
  "resource": {
    "aws_instance": {
      "web": {
        "ami": "${var.ami}",
        "count": 2,
        "source_dest_check": false,
        "connection": {
          "user": "root"
        }
      }
    }
  }
}
```

```
> terraform
Usage: terraform [-version] [-help] <command> [args]
```

Single file command line interface (CLI)

Cross platform

Easy to use

Good documentation in CLI

Providers

"Responsible for understanding API interactions and exposing resources"

125+ available for 1000+ of resources

`azurerm` is a provider for Azure

Spans across IaaS, PaaS, FaaS and SaaS

Modules

"Container for multiple resources that are used together"

Enable easy reuse

Can be used to create abstractions

Supports versioning

E.g. company standard Key Vault module

State

"All the data about your managed infrastructure"

https://www.terraform.io/docs/state/purpose.html

Contains data like:

- IP-addresses
- Connection strings
- Access keys
- Metadata

Needed to map your configuration to resources in e.g. Azure

Track dependencies

Performance and advanced usage

(State) Backends

"Determines how the state is loaded and how operations are performed"

Local

Remote

- Azure Storage
- Consul
- etcd
- Google Cloud Storage
- AWS S3
- Terraform Enterprise

Remote enables collaboration

Data sources

"Data sources allow data to be fetched or computed for use elsewhere in Terraform configuration"

Points to a resources anywhere

- Provider is under the hood

Can read data from it

- Connection strings
- IP
- Configuration

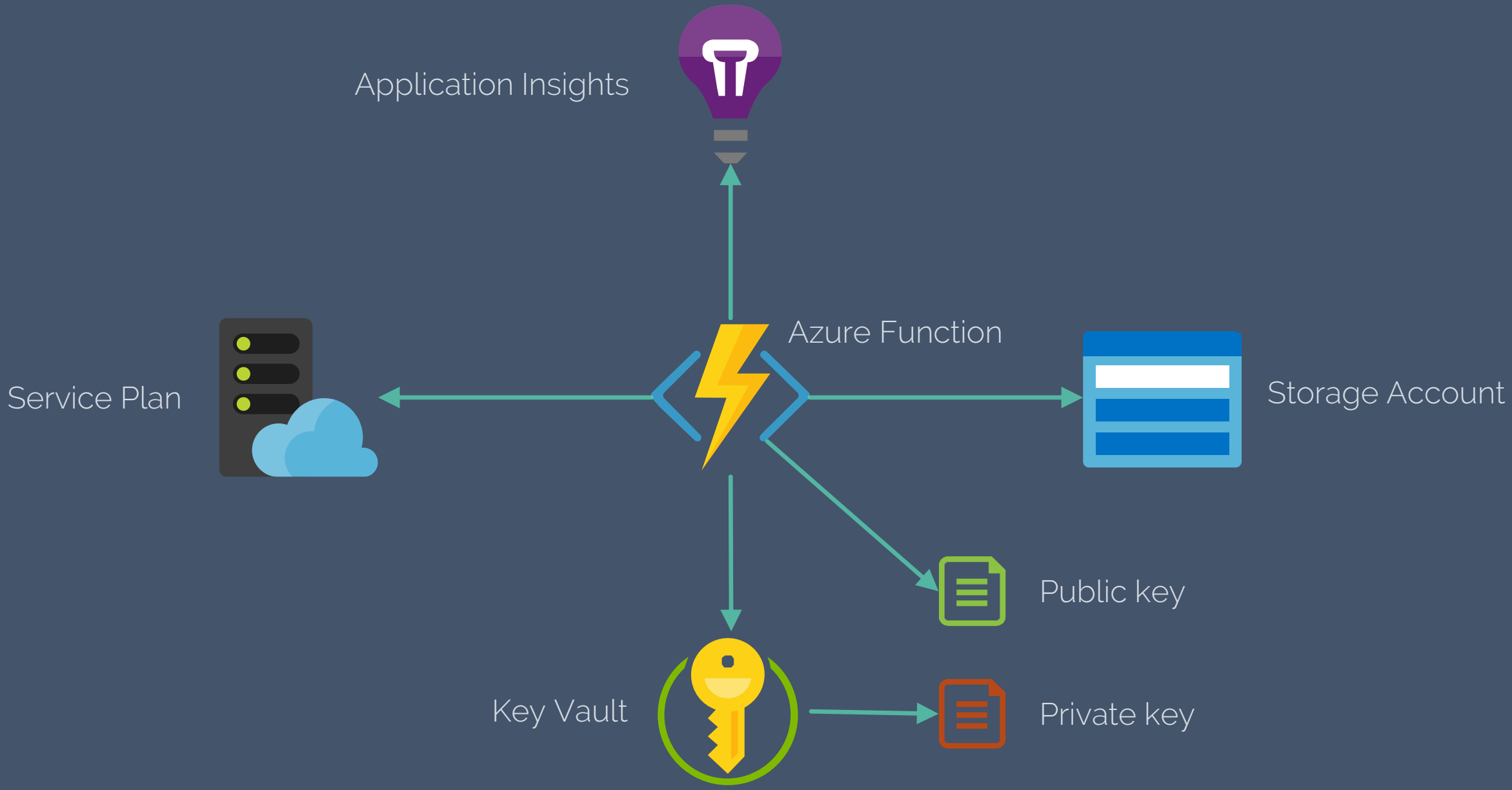
Can avoid hard-coding IPs, connection strings etc.

Can be used to sync settings after changes in a dependency

MICROSOFT  TERRAFORM

DEMO|3

Showing the power of Terraform


Application Insights
Service Plan
Azure Function
Storage Account
Public key
Key Vault
Private key

Terraform: The good

- Well documented
- Good tooling support
- Mature enough
- Scales well
- Good error messages
- Fast!
- Rarely need to resort to shell-scripts
- HCL

Terraform: The bad

- Multi-environment setup requires a extra effort
- State file must be secured
- New Azure APIs can lag behind
  - You can deploy an ARM template with Terraform
- No “all of nothing” deploys
- If something doesn’t work, you’re dependent on the HashiCorp and/or the community

Terraform : The ugly

- Many open issues on GitHub
- People like me tend to oversell it – it's not perfect!
- Might still require extra tooling to deal with large/complex infrastructures

New opportunities

- Easily change passwords
- Easily rotate TLS keys
- Create a new environment per pull-request and destroy the environment after it's merged
- Easily hook in other specialized tools to "complete" the provisioning
  - E.g. Puppet or a shell script
- Multi-cloud solutions
  - Combine Google Spanner, AWS Lambda and Azure Cognitive Services in one Terraform configuration
  - Extreme availability

Gotchas

- Read the docs – thoroughly
  - Azure Key Vault and “sub resources”
- Not designed to deploy code at the same time as you create infrastructure
- Multi-cloud requires extra effort
- It’s not an “abstraction” over the providers

A few of the things ARM can't do

- Work with other clouds and other types of resources
- Replicate the change password / private key scenario
- Remove an entire "environment" consistently in one command
- "Programming" constructs like loops and if-else
- Preview the impact of a deploy (i.e. changes)

Personal experience

- Simple to learn
- More productive with Terraform than with ARM
- Some gotchas
- Can be a hard sell-in
- Requires trial and error to find the organization and techniques that works in your organization

Continue learning

- Try it out!
- Terraform v. 0.12 was just released
  - Very new, so editor support is not ready at this time
  - HCL2 has if-else, for-loops, more types and other useful features
- Structuring
  - Many variations based on your needs and preferences
- Working with multiple environments
  - workspace is a command that might help you

Community extensions and tooling

- Atlantis
  - GitHub-based workflow and collaboration tool
- terraform-docs
  - Create nice documentation from modules
- Terrahelp
  - Encrypt state file before uploading it
- Terragrunt
  - Reduce boilerplate
- … and loads more

Useful resources

- Awesome Terraform – large collection of curated resources
  https://github.com/shuaibiyy/awesome-terraform
- Microsoft's official Terraform documentation
  https://docs.microsoft.com/en-us/azure/terraform/
- Links to various Azure-related Terraform resources and examples
  https://github.com/Azure/awesome-terraform
- Blogpost from HashiCorp with a list of talks on Terraform from Microsoft Ignite
  https://www.hashicorp.com/blog/hashicorp-at-microsoft-ignite
- Microsoft's official Terraform VM
  https://azuremarketplace.microsoft.com/en-us/marketplace/apps/azure-oss.terraform
- VS Code Extensions
  https://github.com/terraform-providers/terraform-provider-azurerm/blob/master/examples/
- Terraform Up and Running (Book from 2017)
  https://www.terraformupandrunning.com

enso

Slides and code: http://bit.do/tnic

Thanks!

TORSTEIN NICOLAYSEN

@tnicolaysen

tnicolaysen@gmail.com

NDC OSLO
2019-06-19

Image by Bruno DE LIMA from Pixabay