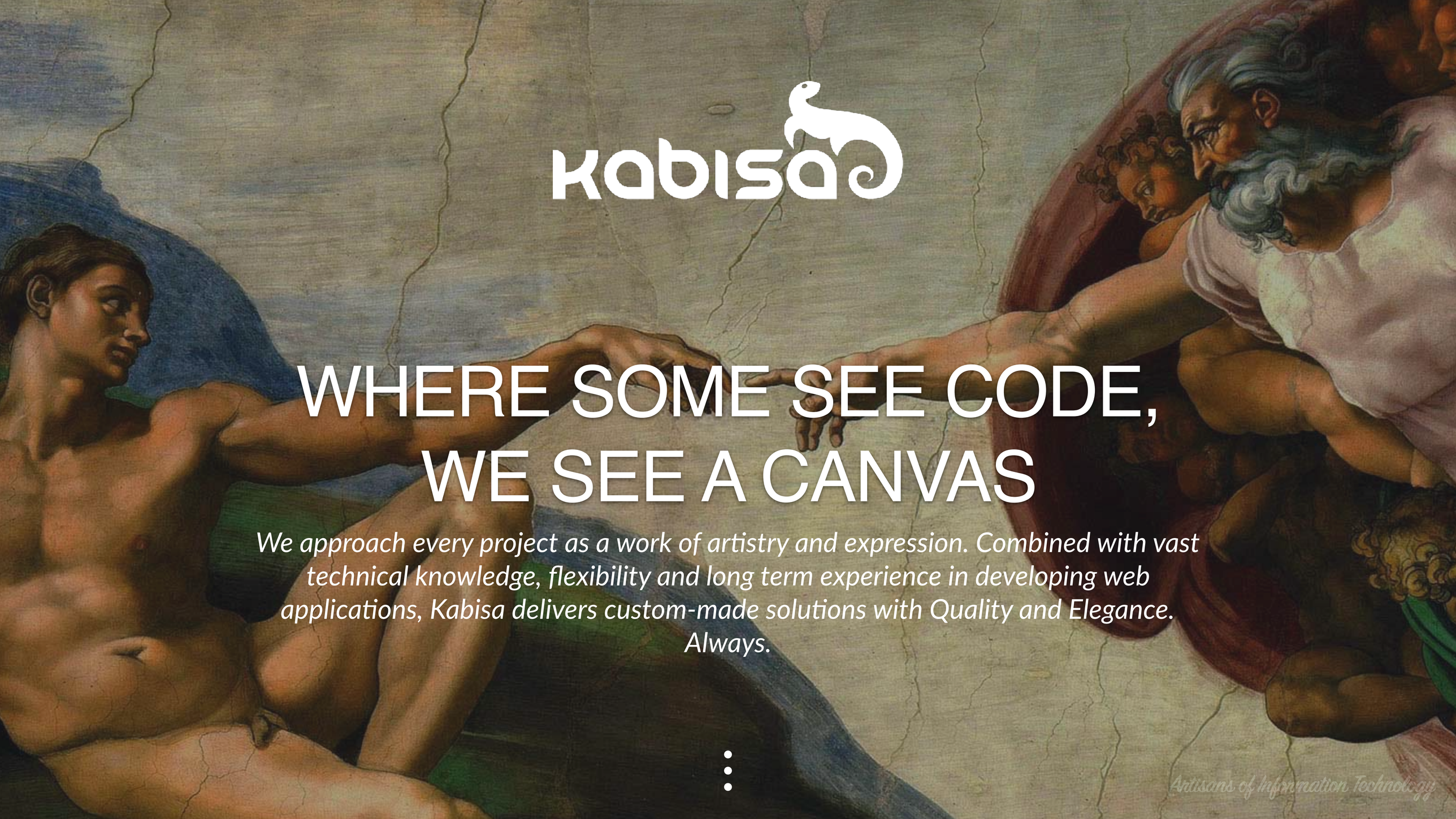

Kabisa
Artisans of Information Technology

Agenda

- About us
- About you
- Introduction into Infrastructure as Code (IaC), AWS and Terraform
- Getting started
- Pizza!
- Terraform
- The workshop
- Drinks!



WHERE SOME SEE CODE, WE SEE A CANVAS

We approach every project as a work of artistry and expression. Combined with vast technical knowledge, flexibility and long term experience in developing web applications, Kabisa delivers custom-made solutions with Quality and Elegance. Always.

Artisans of Information Technology

Kabisa

Kabisa is a software development agency, specialised in custom web and mobile app development & integration.

12+
years in business

300+
Successful projects

45
team members

3
Offices

Artisans of Information Technology

Kabisa

Kabisa is een maatwerk software ontwikkelaar die zich specialiseert op het gebied van web en mobiele app ontwikkeling & integratie.

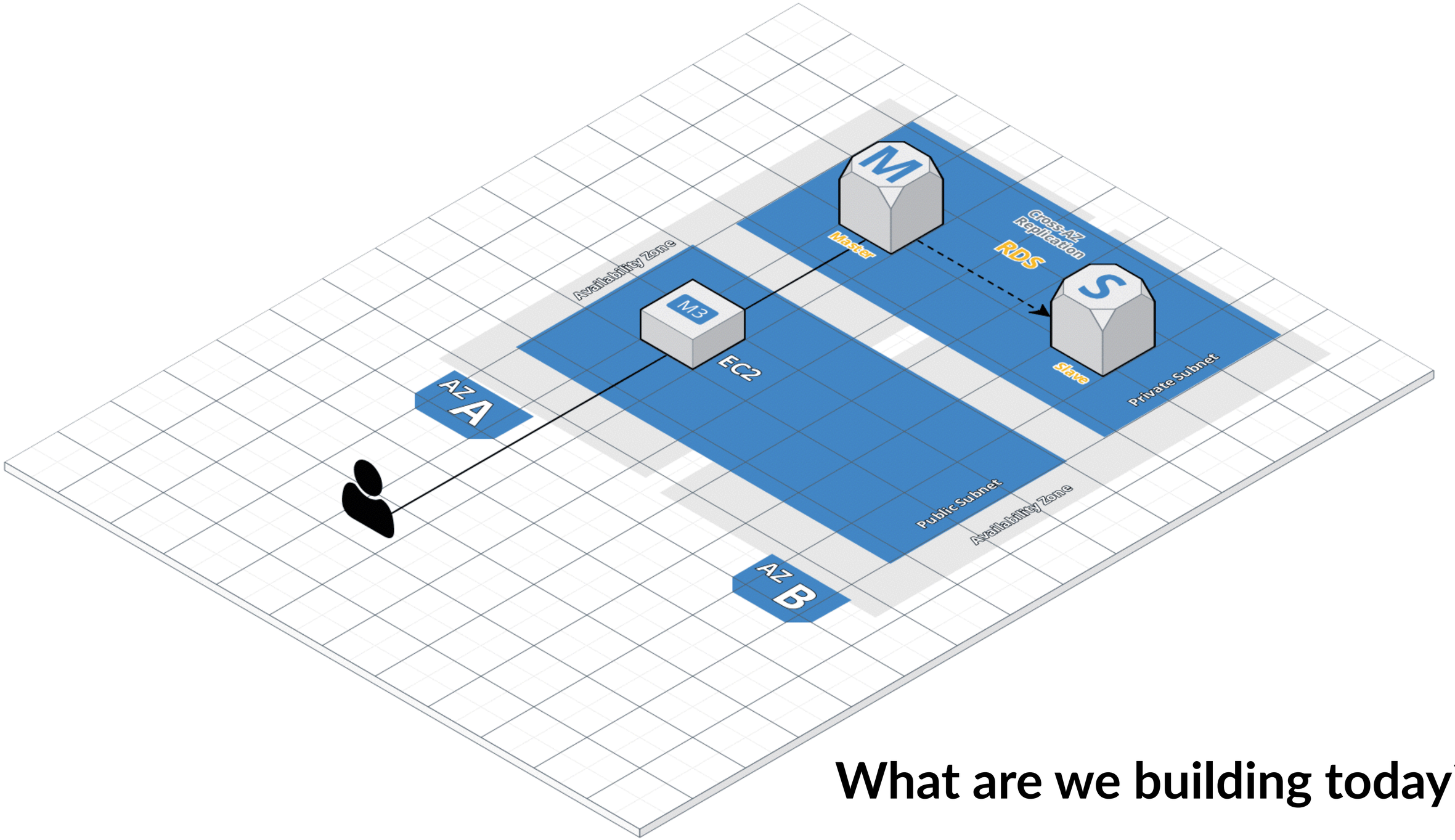
Luk van den Borne
DevOps Engineer

Andy Maes
DevOps Engineer

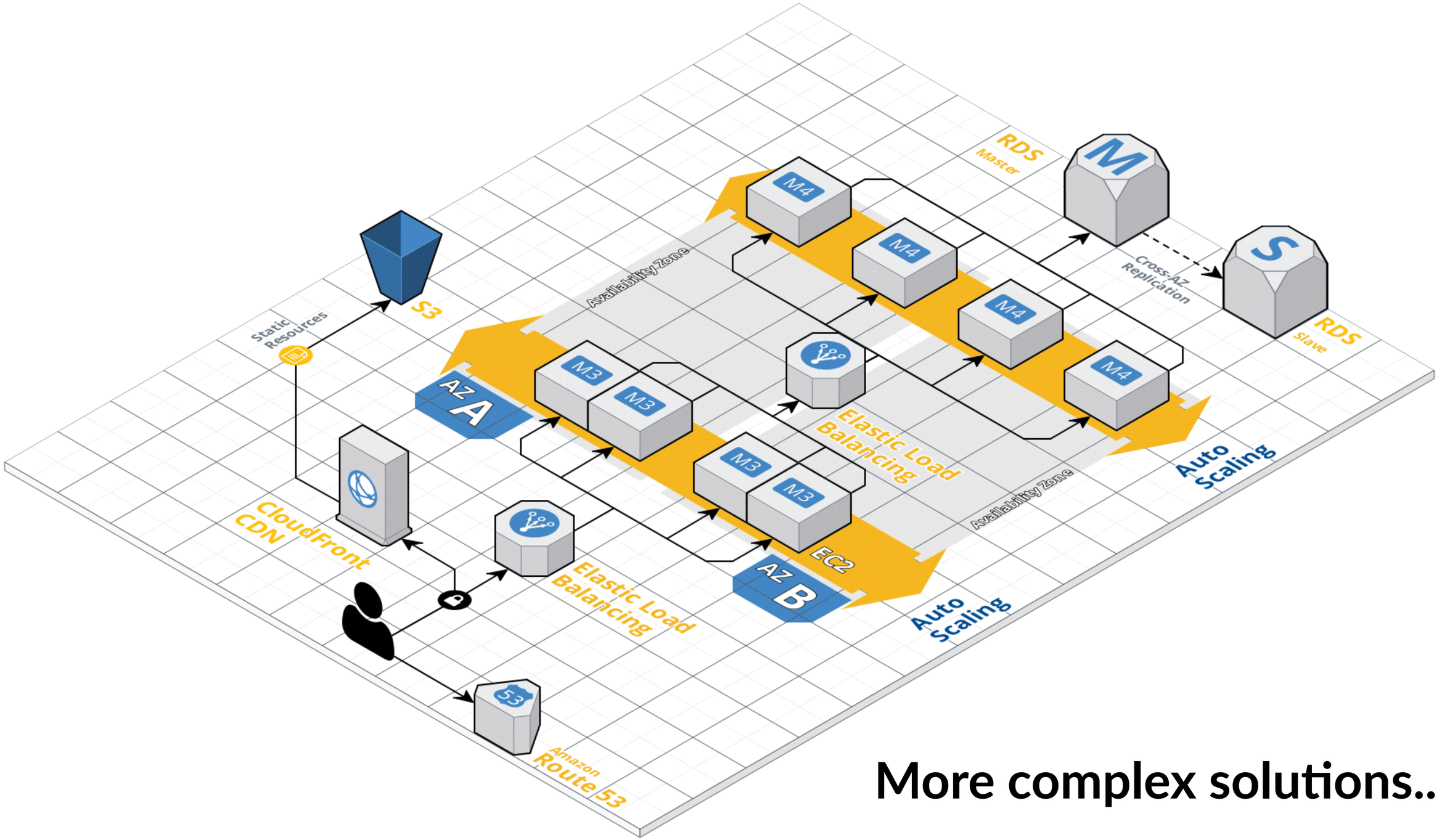
Joost Saanen
DevOps Engineer

Artisans of Information Technology


Who are you?



What are we building today?



More complex solutions..

“A tool for building, changing and versioning infrastructure safely and efficiently”


KEEP
CALM
AND
AUTOMATE
ALL THE THINGS

Terraform

- A way to manage AWS, and other public cloud platforms
- Declarative language
- Easy to read and write
- Open source!
- Free!

Terraform - Getting started

- Install AWS CLI
- Install Terraform
- Create an IAM user and use aws configure to insert the credentials into ~/.aws/credentials
- git clone git@github.com:kabisa/terraform-workshop.git

Terraform - Core Concepts

Core concepts:

- HashiCorp Configuration Language (HCL)
- Providers
- Resources
- Data
- Output
- Variables
- References

Terraform - HCL

- Declarative language

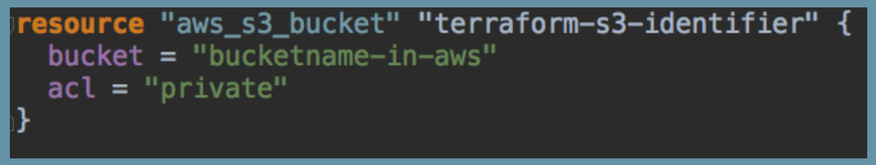
Terraform - providers

- AWS
- Google Cloud
- Azure
- OpenStack
- Many, many more!
- https://www.terraform.io/docs/providers/

```
provider "aws" {
  region                  = "eu-west-1"
  shared_credentials_file = "~/.aws/credentials"
  profile                 = "kabisa-demo-aws"
}
```

```
[kabisa-aws-demo]
aws_access_key_id = AKIAABCDEFERGHDGSDFG
aws_secret_access_key = sdfSDFhrtewgav23wSFSDfsdfsd/sdfSDFSDFSDF
```

Terraform - resources



- https://www.terraform.io/docs/providers/aws/r/s3_bucket.html
- Important sections:
  - Argument Reference
  - Attributes Reference

Terraform - Data

Terraform - output


```
output "cloudfront_domain" {
  value = "${aws_cloudfront_distribution.cf_distro.domain_name}"
}
```

```
output "ip" {
  value = "${aws_eip.elastic_ip.public_ip}"
}
```

Terraform - Variables

- https://www.terraform.io/docs/configuration/variables.html
- Data Types:
  - Scalar: “hello world”, true/false, 0/1/2
  - List: [“a”, “b”, “c”]
  - Map: { var1: “value1:”, var2: “value2” }

Terraform - Variables

- Variable definitions: variables.tf
- Variable values:
  - terraform.tfvars
  - -var-file=secrets.tfvars parameter
  - -var ‘var1=value'
  - Interactively via terraform

```
variable "azs" {
  type = "list"
  default = ["eu-west-3a", "eu-west-3b", "eu-west-3c"]
}

/* EC2 vars */
variable "ec2_type" {}
variable "ec2_ami" {
  description = "The ID of the AMI you want to install"
}
variable "ec2_keypair" {}
```

```
/* General vars */
shared_credentials_file = "~/.aws/credentials"
profile = "kabisa-demo-aws"

/* Customer vars */
customer = "kabisa"
domain = "kabisa.nl"

/* ec2 vars */
ec2_type = "m5.large"
```

Terraform - variables

Original

```
provider "aws" {
  region                  = "eu-west-1"
  shared_credentials_file = "~/.aws/credentials"
  profile                 = "kabisa-demo-aws"
}
```

w/ vars

```
# General AWS data
provider "aws" {
  region                  = "${var.region}"
  shared_credentials_file = "${var.shared_credentials_file}"
  profile                 = "${var.profile}"
}
```

Interactively

```
# General AWS data
provider "aws" {
  region     = "${var.region}"
  access_key = "${var.access_key}"
  secret_key = "${var.secret_key}"
}
```

```
// No default value or value through terraform.tfvars
// so they will be prompted interactively
variable "access_key" {}
variable "secret_key" {}
```

Terraform - References

- Variables: “${var.varname}”
- Resources: “${resource_type.resource_id.attribute}”
- Data: “${data.data_type.data_id.attribute.format}”

```
resource "aws_route53_zone" "kabisa-nl" {
  name = "kabisa.nl"
}

resource "aws_route53_record" "mx-kabisa-nl" {
  zone_id = "${aws_route53_zone.kabisa-nl.zone_id}"
  name = ""
  type = "MX"
  records = [
    "1 ASPMX.L.GOOGLE.COM.",
    "5 ALT1.ASPMX.L.GOOGLE.COM.",
    "5 ALT2.ASPMX.L.GOOGLE.COM.",
    "10 ASPMX2.GOOGLEMAIL.COM.",
    "10 ASPMX3.GOOGLEMAIL.COM.",
  ]
  ttl = "${var.route53_default_ttl}"
}
```

Terraform - Running terraform

- terraform init
- terraform plan
- terraform apply
- .tfstate files

Terraform - caveats

- Think about secrets management! The tfstate file basically contains everything!
- Carefully check the actions in terraform plan, especially destroy or destroy-and-create operations

Terraform - general remarks

- Keep the documentation and console at hand.
- Think about maintainability, readability and modularity.
- Build immutable infrastructure.

Terraform - workshop part I

- The goal: terraform an EC2 instance with a root volume
- It has to allow incoming SSH access from your current IP and http and https access from the outside world. It should also allow all outgoing traffic
- Make sure it has a reusable (elastic) ip address. This ip address should be outputted at the end of a terraform apply.
- You should be able to login to the node via SSH
- Make sure the code could be reused for other projects
- Free tier: t2.micro

Terraform - workshop part II

- The goal: terraform an RDS instance with a root volume
- The application we're going to host will be Wordpress
- Make sure the database service is redundant across 2 availability zones for the purpose of disaster recovery
- Allow only nodes with the security group you created in the previous part for the EC2 node
- Output the database name, username, password and endpoint
- Free tier eligible: db.t2.micro
- Challenge: instead of creating a single instance, create an Aurora cluster with 2 instances. This isn't covered under the Free Tier

Terraform - workshop part III

- Use the user data in EC2 to bootstrap a Wordpress application to the point that it's ready to be installed out of the box.
- The easiest way to run WP is using apache and mod_php in the default apache DocumentRoot (/var/www/html/)
- Tip: WP can be downloaded via wget https://wordpress.org/latest.tar.gz
- The application in the tar is located in the "wordpress" directory. To strip the first level, you can use --strip 1 in the tar command.
- Don't forget to chown the application to the default webserver user (www-data for Debian/Ubuntu)
- Note that changing the user data will destroy-and-create your ec2 instance. Once again: build immutable infra!

Terraform - cool stuff

- Of course you can use Terraform with many more resources and providers
- You can create reusable TF-modules, includable from a git source
- There's a fairly large repository with pre-made TF-modules: https://registry.terraform.io/

Terraform - caveats and challenges

- Secrets management:
  - remote state storage
  - Use AWS Vault instead of plain-text credentials: https://github.com/99designs/aws-vault
- Some resources have to be created in us-east-1, e.g. TLS certificates for CloudFront
- (Further) provisioning of the infrastructure, e.g. SSH keys, nginx config
- Converting hand-crafted infrastructure into Terraform'ed IaC

Terraform - Remote State


```
terraform {
  backend "s3" {
    region         = "eu-west-1"
    bucket         = "kabisa-tfstates"
    dynamodb_table = "kabisa-tflocks"
    key            = "customer/production/terraform.tfstate"
    encrypt        = true
    profile        = "kabisa-profile"
  }
}
```

Terraform - Different provider


```
provider "aws" {
  alias = "us-east-1"
  region = "us-east-1"
  shared_credentials_file = "${var.shared_credentials_file}"
  profile                 = "${var.profile}"
}
```

```
resource "aws_acm_certificate" "kabisa-nl" {
  domain_name = "${var.domain}"
  validation_method = "DNS"
  subject_alternative_names = ["www.${var.domain}"]
  provider = "aws.us-east-1"
}
```