From Dev to Prod

Robert Clements

MSDS Program

University of San Francisco

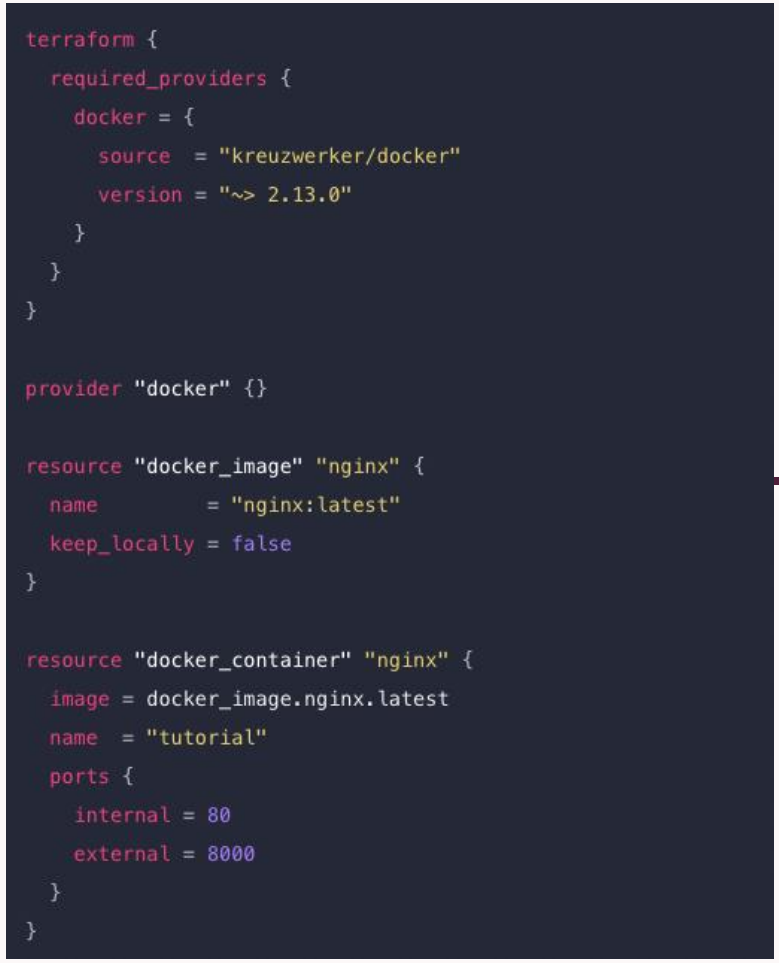
What to Expect

- Goal: to learn about infrastructure-as-code and CI/CD/CT.

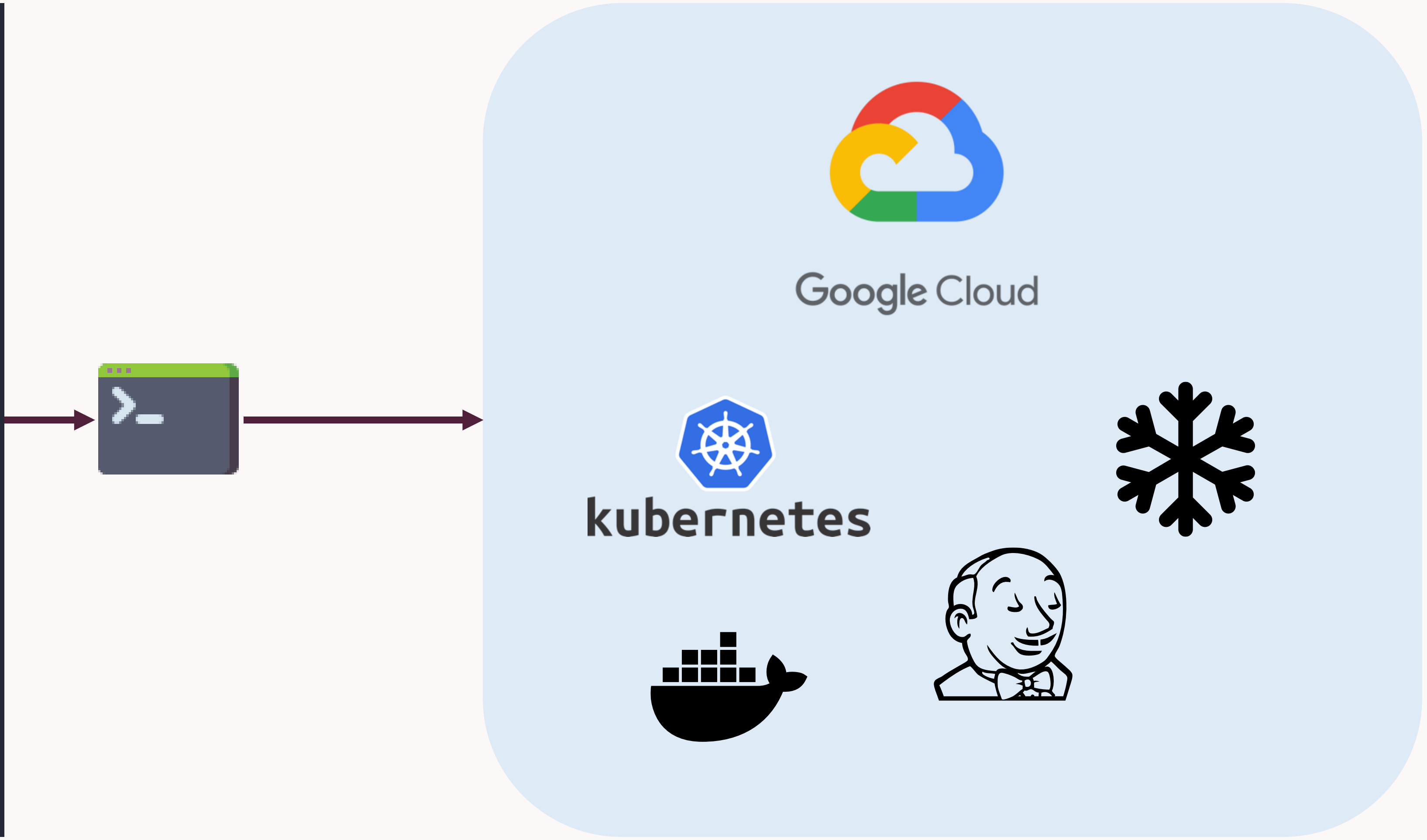
- How: we will learn the benefits of using an infrastructure-as-code tool and practice with Terraform in a simple tutorial lab. We will also cover CI/CD/CT and then apply it to our projects.

Infrastructure as Code

Infrastructure-as-Code (IaC)



```
terraform {
  required_providers {
    docker = {
      source  = "kreuzwerker/docker"
      version = "~> 2.13.0"
    }
  }
}

provider "docker" {}

resource "docker_image" "nginx" {
  name         = "nginx:latest"
  keep_locally = false
}

resource "docker_container" "nginx" {
  image = docker_image.nginx.latest
  name  = "tutorial"
  ports {
    internal = 80
    external = 8000
  }
}
```

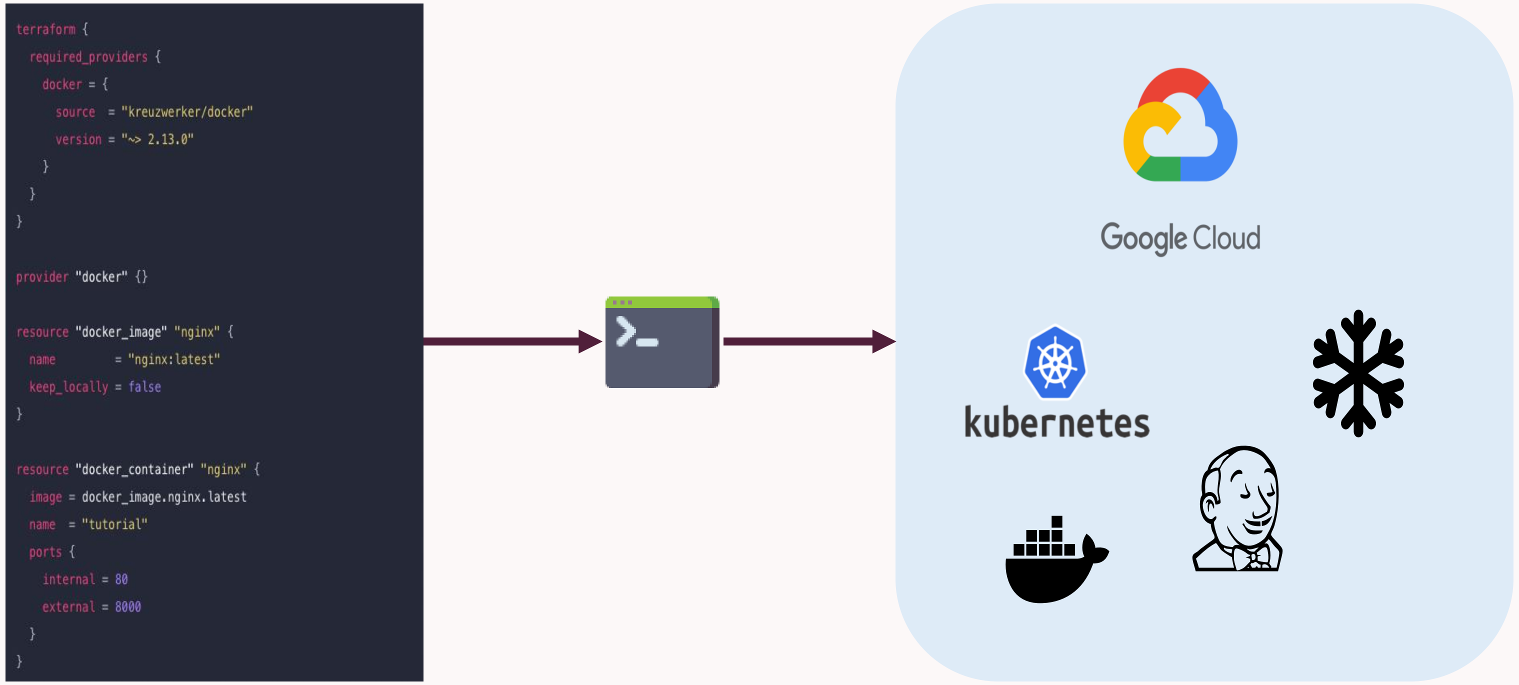


Benefits of Infrastructure-as-Code (IaC)

- Versioning
- Reusability and collaboration
- Provider-agnostic

Terraform Workflow

- ID infrastructure needed
- Create configuration files
- Install any plugins (providers) needed for managing the infrastructure
- Review infrastructure
- Apply changes



```
terraform {
  required_providers {
    docker = {
      source  = "kreuzwerker/docker"
      version = "~> 2.13.0"
    }
  }
}

provider "docker" {}

resource "docker_image" "nginx" {
  name         = "nginx:latest"
  keep_locally = false
}

resource "docker_container" "nginx" {
  image = docker_image.nginx.latest
  name  = "tutorial"
  ports {
    internal = 80
    external = 8000
  }
}
```

Terraform Workflow

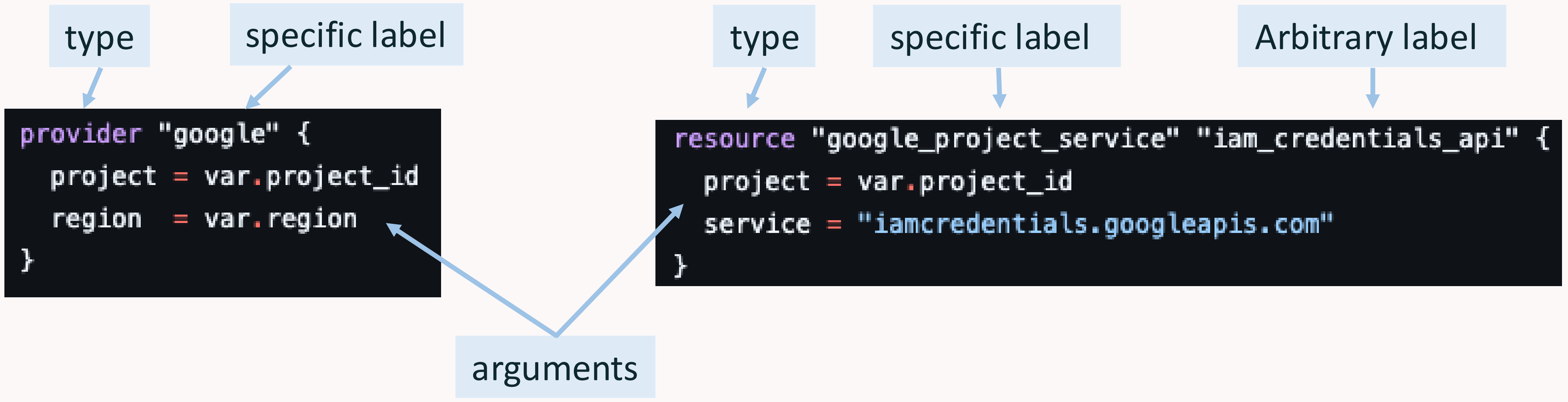
- Write and Init – write your Terraform configuration and initialize Terraform using `terraform init`
- Plan – run `terraform plan` to see the execution plan while iterating on your configuration
- Apply – run `terraform apply` to look at the final execution plan and provision the infrastructure
- Destroy – run `terraform destroy` to stop resources

Terraform HCL Syntax

- Arguments – like variables or attributes

```
project_id = ”my-project-id”
```

- Blocks -



```
provider "google" {
  project = var.project_id
  region  = var.region
}
```

```
resource "google_project_service" "iam_credentials_api" {
  project = var.project_id
  service = "iamcredentials.googleapis.com"
}
```

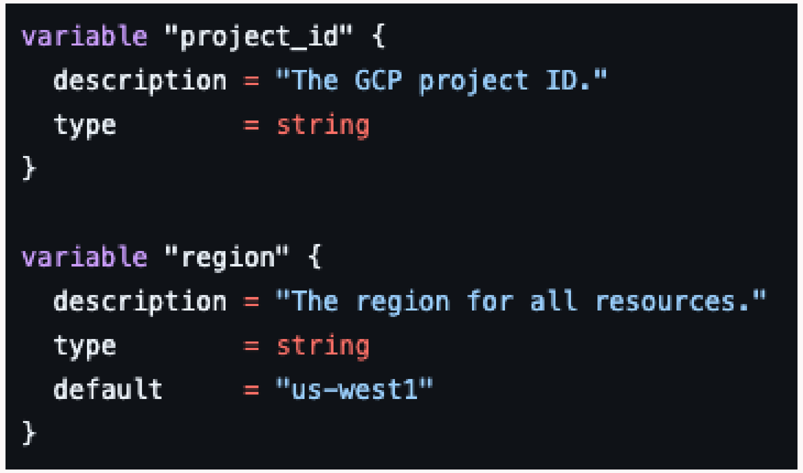
Blocks Types

- provider – GCP, AWS, etc.
- terraform – terraform version, global settings
- variable – for declaring variables
- resource – the resources your infrastructure consists of: buckets, databases, compute, etc.
- module – calls a child module, i.e. a set of resources used together
- output – prints/returns a value to the user or a parent module
- data – for reading data from a data source for use in Terraform

Terraform Variables

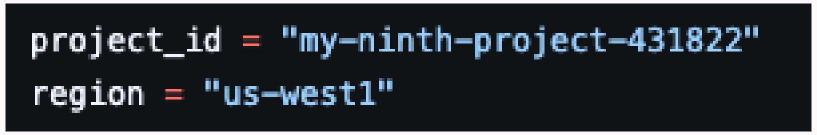
Variables – for anything you don't want hard-coded

- variables.tf – for declaring variables with variable blocks



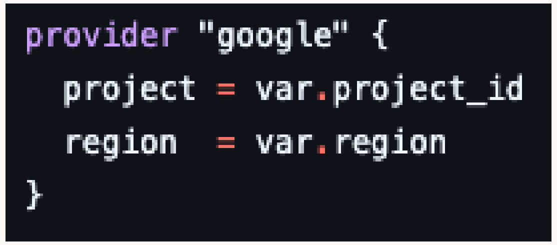
```
variable "project_id" {
  description = "The GCP project ID."
  type        = string
}

variable "region" {
  description = "The region for all resources."
  type        = string
  default     = "us-west1"
}
```

- terraform.tfvars – for defining/setting the variables declared in variables.tf



```
project_id = "my-ninth-project-431822"
region = "us-west1"
```

- Reference variables in your tf configuration files



```
provider "google" {
  project = var.project_id
  region  = var.region
}
```

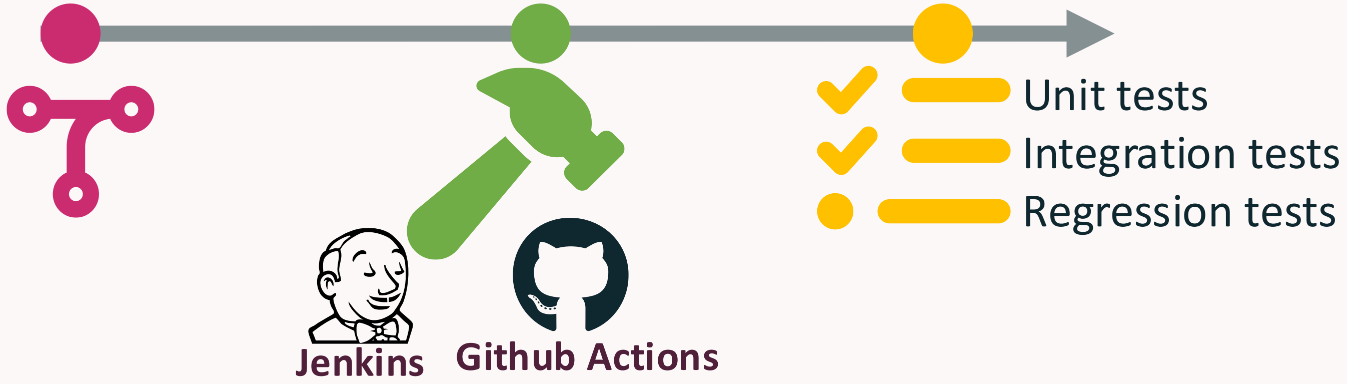
Terraform Modules

- Collection of .tf files in the same directory
- Root module – defined by the main working directory .tf files
- Child modules – modules called by other modules, usually the root module, using the module block type
- Modules can be local or remote
- Modules can be called multiple times

Continuous Integration/Continuous Delivery

Continuous Integration/Continuous Delivery

- DevOps – for speeding up deployment of software applications using automation
- Build, test, and deploy
- Integration:
  - **Merge branches**
  - **Kick off build**
  - **Kick off tests**
- Delivery:
  - Deployment



CI/CD for ML Systems (coming soon)

What opportunities are there for automation?

When is automation really necessary?

What is continuous training (CT)?