Chair for Decentralized Information Systems and Data Management
TUM School of Computation, Information and Technology
Technical University of Munich

# Cloud Information Systems

## Exercise 12

20th January 2025

Prof. Dr. Viktor Leis, M.Sc. Till Steinert

# 1. Announcement: Guest Lecture This Week

- Guests: Check24
- Wednesday *22.01.2025*, 12:30 in HS1
- Content of the Talk is **relevant for the exam**

# 2. Recap: Containers (Docker)

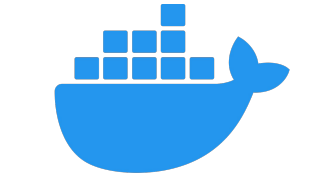
- lightweight form of virtualization
  → rapid creation, short lifetime, replication

- built on top of operating system (Linux)

- provides isolation between processes

- allow packaging software artifacts, including their required libraries

→ allows for consistent development environment




Link

# 2. Recap: Containers (Kubernetes)

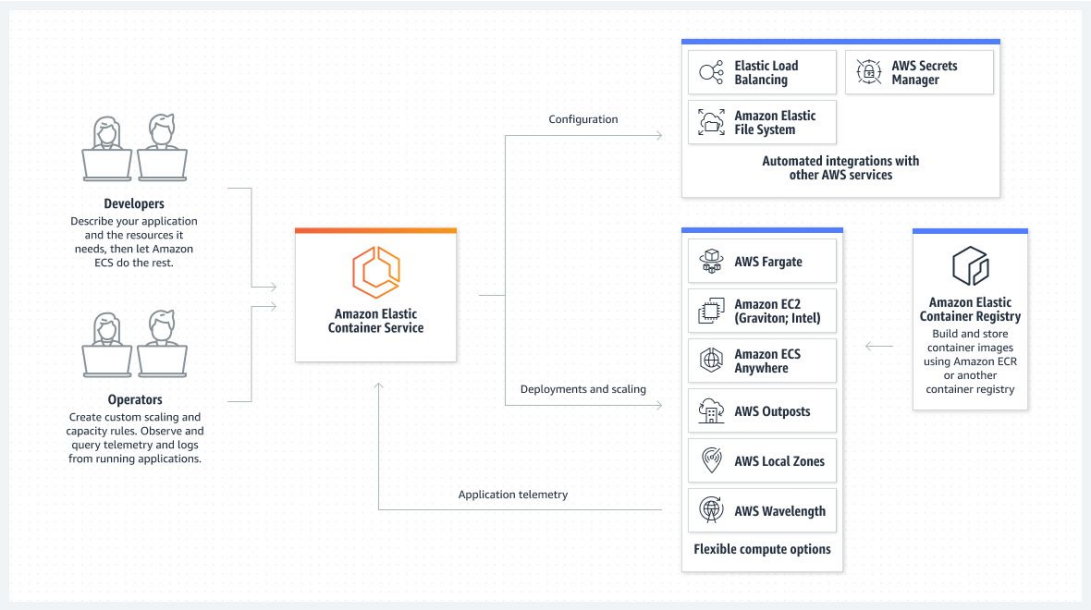
- (open-source) platform for managing containerized workloads and services
- provides framework to run distributed systems resiliently by handling tasks such as scaling, failover, and deployment patterns
- a pod is the smallest deployable unit in Kubernetes
  - can contain one or more containers

kubernetes

Link

# 2. Recap: Amazon Elastic Container Service (ECS)

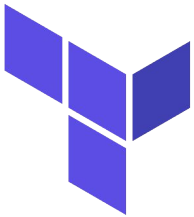
- proprietary container management service that handles infrastructure tasks
- integrated with AWS ecosystem (and therefore, also tied to AWS)
- can run containers on EC2 or Fargate




Guide Documentation

# 2. Recap: Terraform

- open-source infrastructure as code (IaC) tool by HashiCorp
- allows users to define and provision infrastructure resources across various cloud providers and services
- declare resources with Terraform language (HCL)

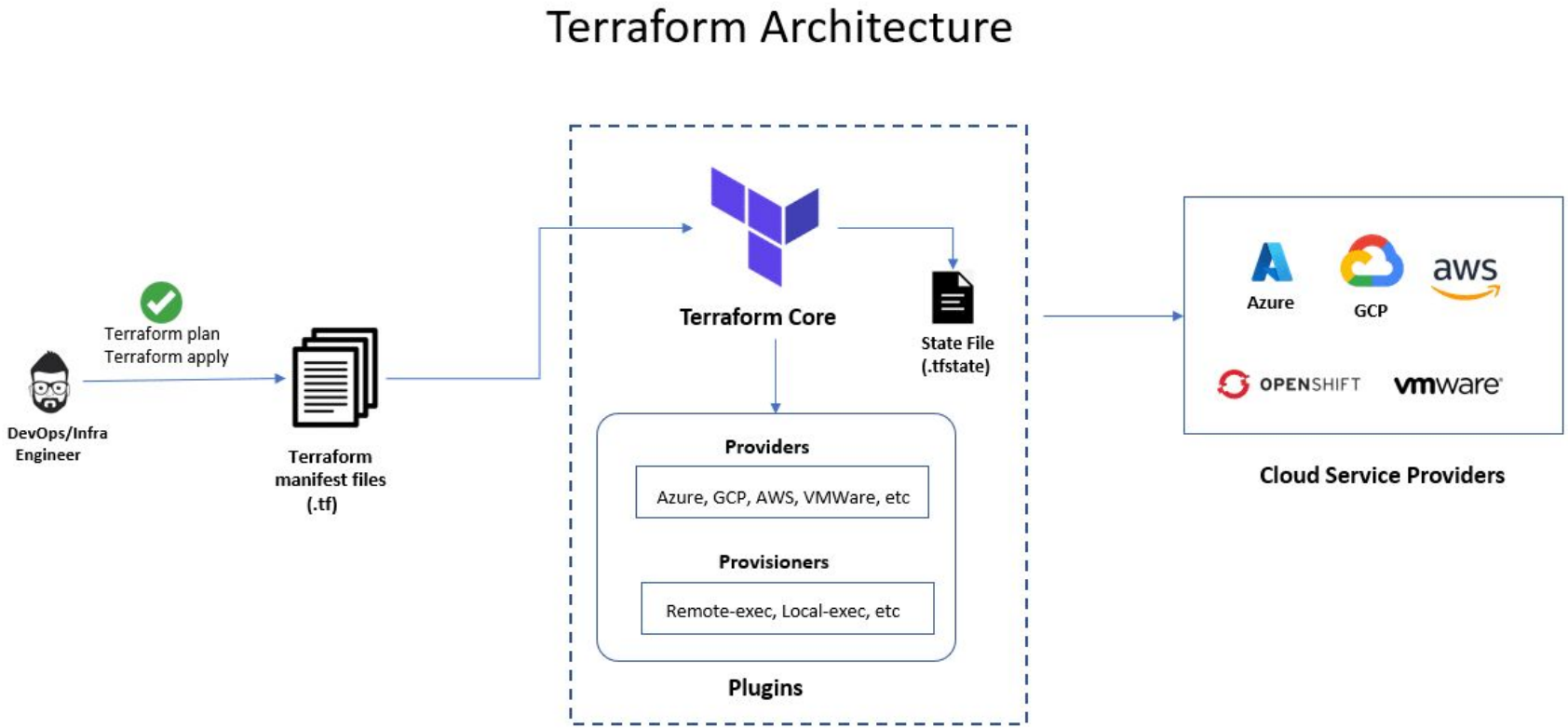


HashiCorp Terraform

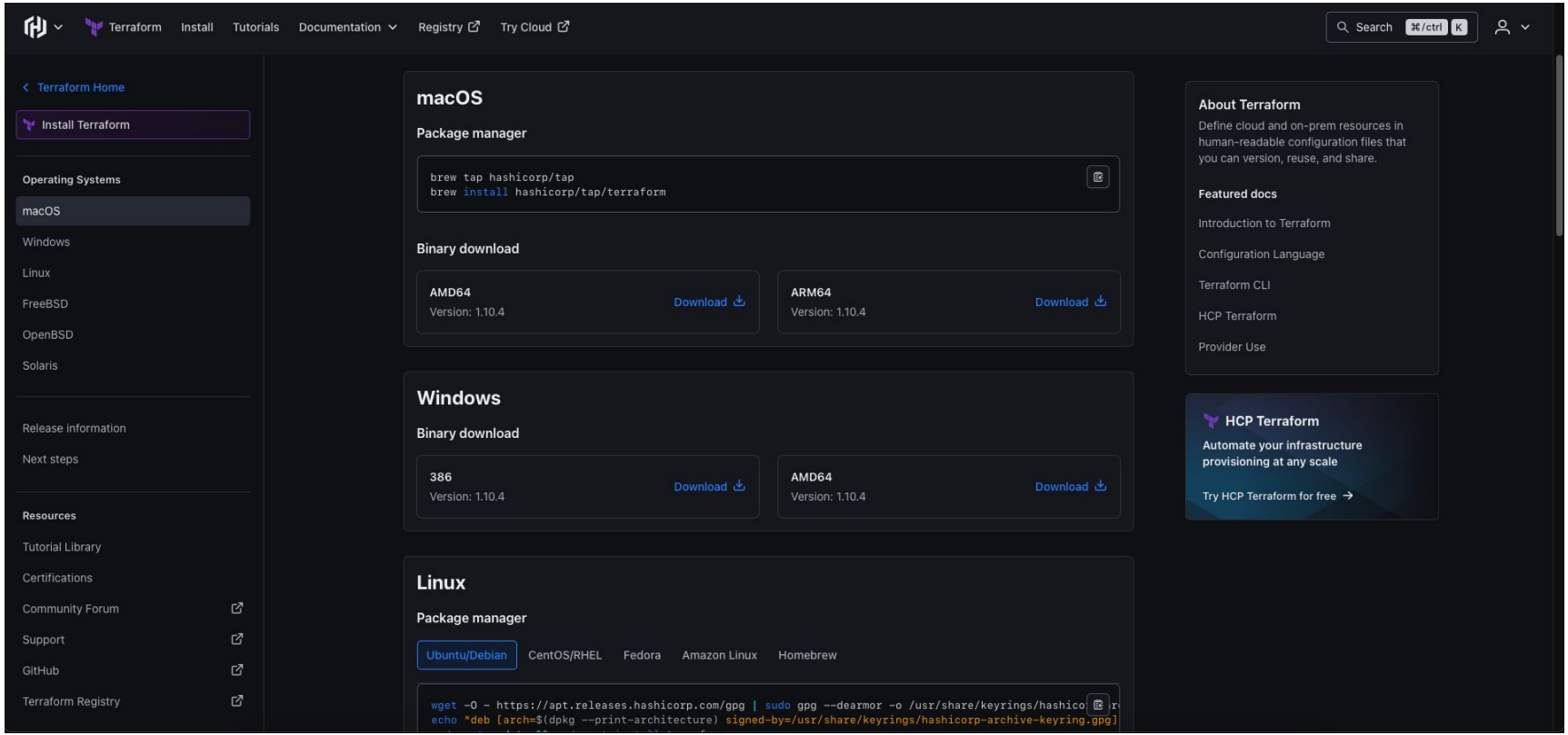
Link

# 3. Live Demo

# Terraform Architecture



# 1. Terraform Installation



# 2. Basic Configuration File

main.tf > ...

```
1  terraform {
2    required_providers {
3      aws = {
4        source  = "hashicorp/aws"
5        version = "~> 5.0"
6      }
7    }
8
9    required_version = ">= 0.12"
10 }
11
12 provider "aws" {
13     region = "us-east-1"
14 }
```

**Terraform Block:** Specify Providers needed for the Infrastructure

**Provider Block:** Setup Provider (e.g., region, credentials, etc.)

# 3. Terraform Init

```
tillsteinert@Tills-Mac-mini terraform % terraform init

Initializing the backend...

Initializing provider plugins...
- Finding hashicorp/aws versions matching "~> 5.0"...
- Installing hashicorp/aws v5.84.0...
- Installed hashicorp/aws v5.84.0 (signed by HashiCorp)

Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
tillsteinert@Tills-Mac-mini terraform % ls -a
.                       ..                      .terraform              .terraform.lock.hcl     main.tf
```

Cache for downloaded Providers

Lock Provider Versions

# 3. Terraform CLI

- Important Commands
  - terraform init: Initialize Terraform in current working directory
  - terraform plan [--out=<file>]: Creates an execution plan
  - terraform apply: Apply generated plan to setup infrastructure
  - terraform destroy: Destroy all Object created by Terraform

# 4. Questions