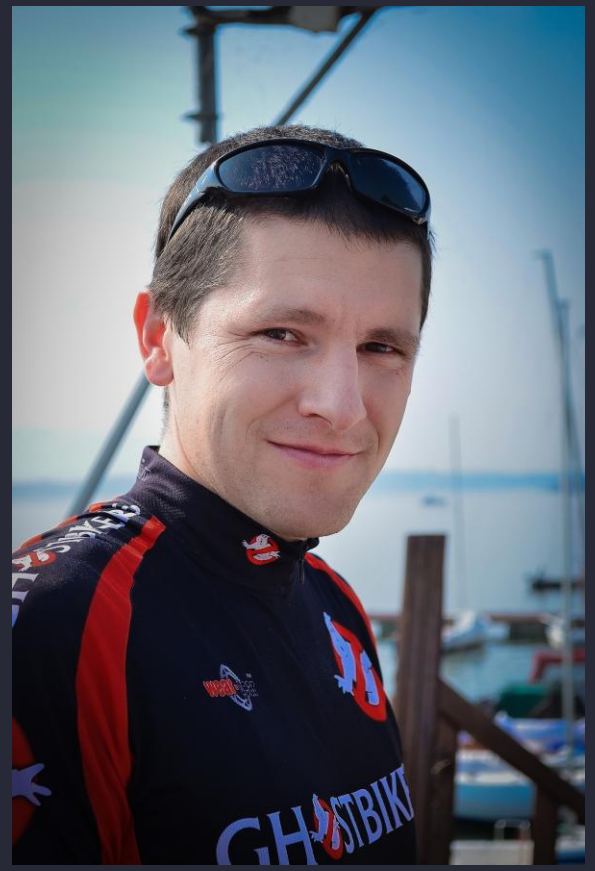
Spring Boot is Just the Beginning – Microservices' Journey to the Cloud

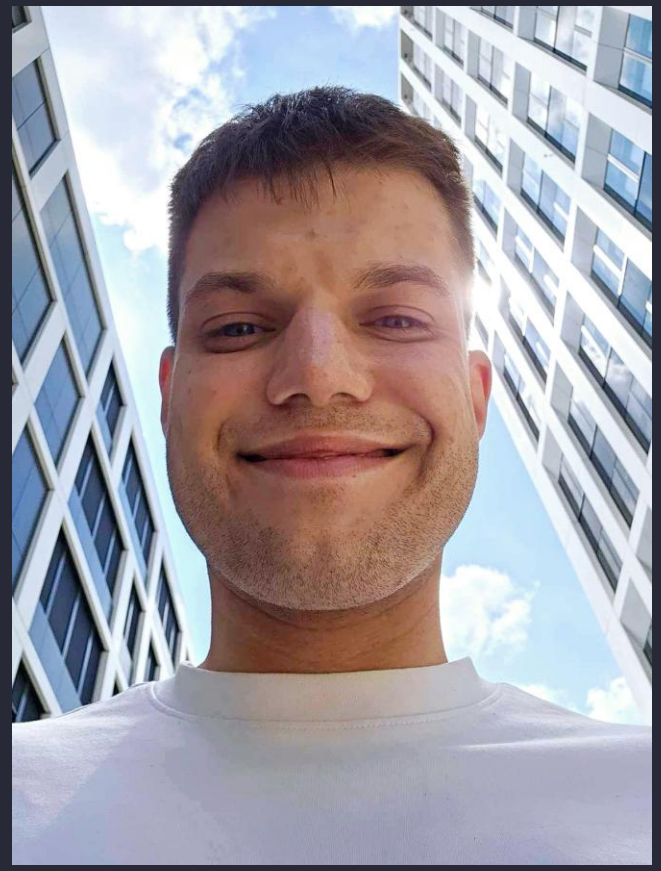
Tech Talk 2025, Piotr Strojek & Damian Cywiński

Capgemini

Who are we?

- **Solution Architects at Capgemini**
- Designing and **delivering Cloud solutions for clients in Automotive sector,** currently working for BMW
- Specialized in:
  - Cloud migrations
  - Cloud-native solutions
  - Serverless
  - DevOps
  - Java





Who are YOU?

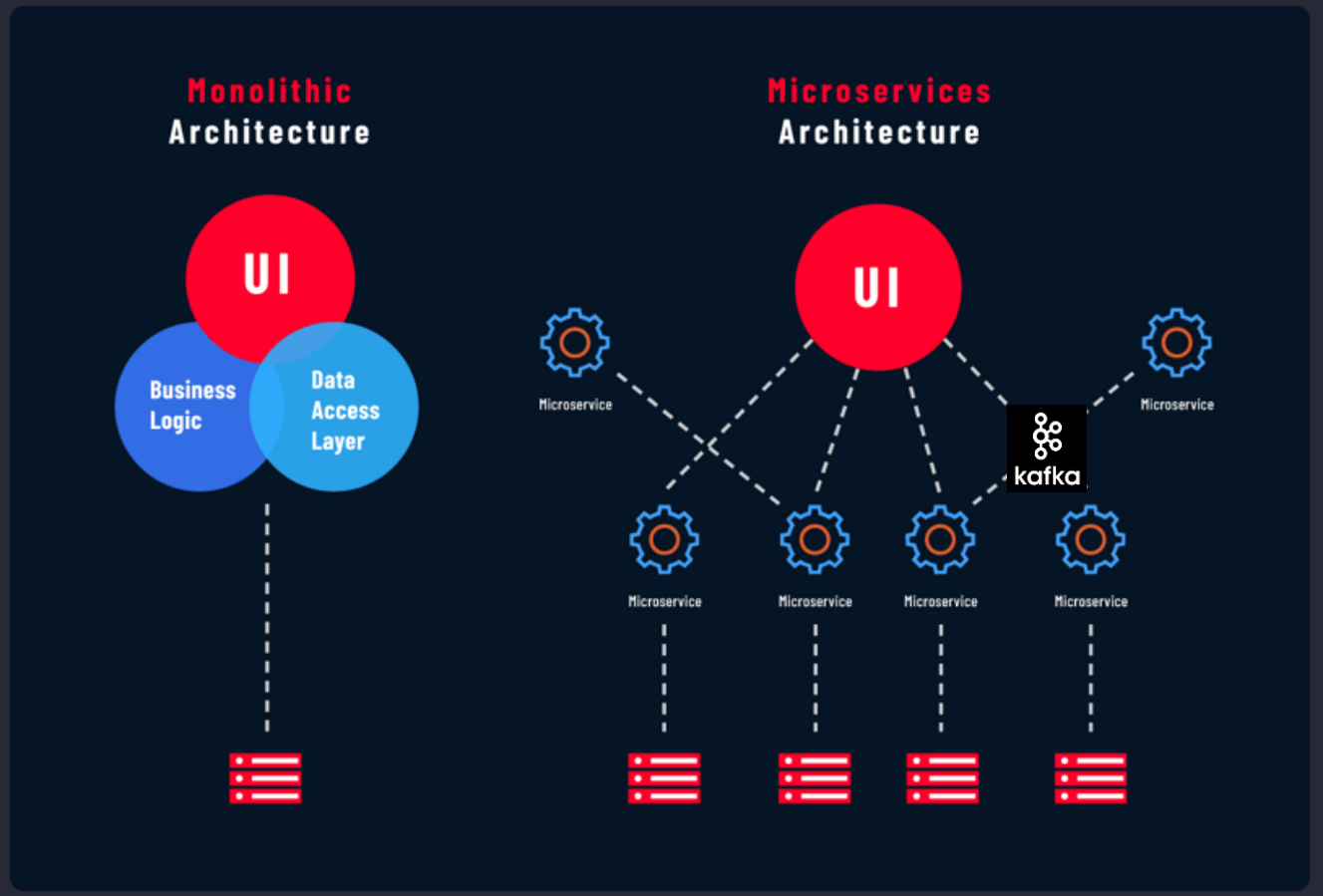
What will we discuss today?

SCOPE

1. Developers & apps these days…
2. How to deploy Spring Boot application to AWS…
3. …but is it enough?
4. Components of the modern Cloud applications…

Apps these days…

MONOLITH VS MICROSERVICES

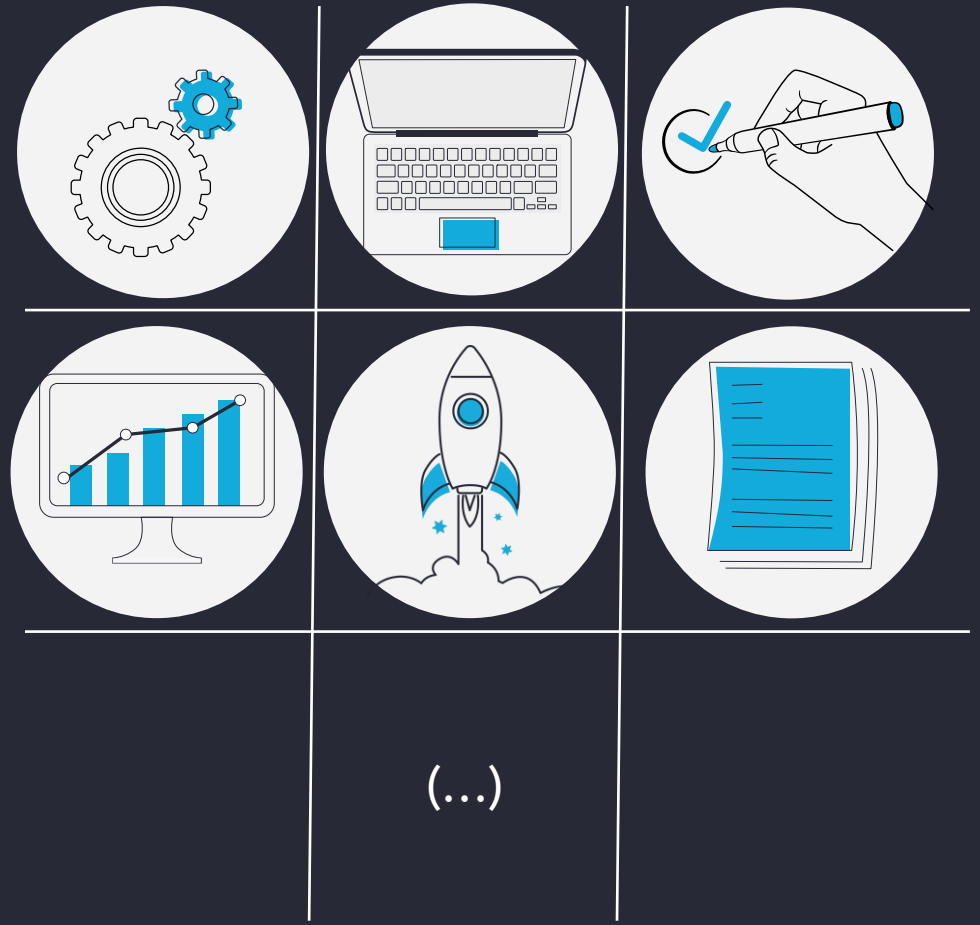


Source: SparkFabrik

Developers these days…

COMPETENCES

Classic competences

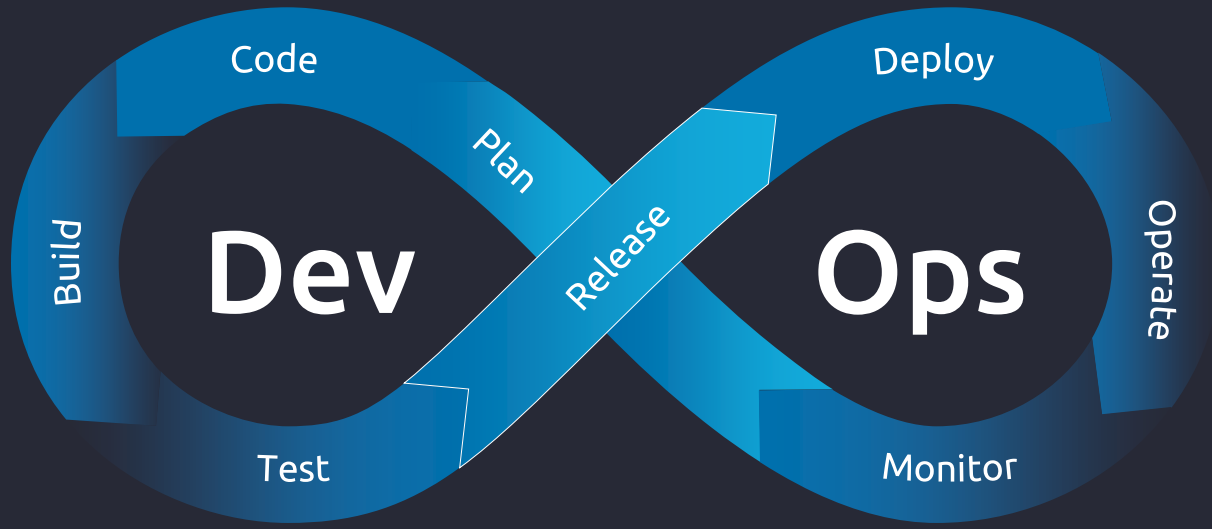


Modern (Cloud) competences…

Dev(Ops) these days…

COMPETENCES

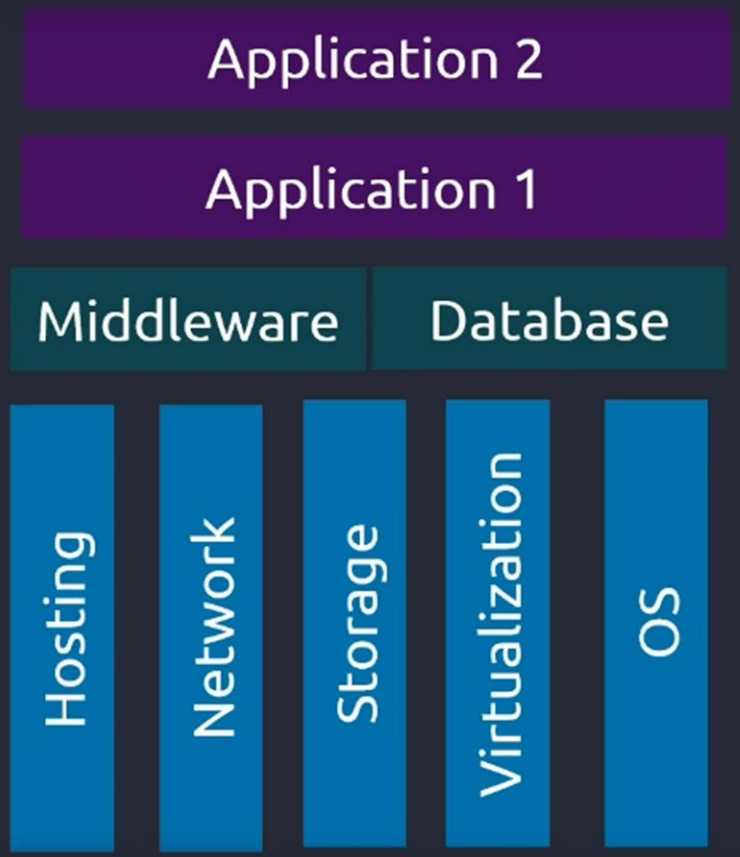
- **Dev (Developer):**
  - Focuses on writing and maintaining application code (features, bug fixes).
- **DevOps (Developer + Operations):**
  - Bridges development and operations.
  - Works across the full lifecycle of the app, not just the code.
- **DevOps Skills:**
  - CI/CD pipelines (Jenkins, GitHub Actions)
  - Infrastructure as Code (Terraform, Ansible)
  - Cloud platforms (AWS, Azure, GCP)
  - Containerization (Docker)
  - Container orchestration (Kubernetes, ECS Fargate)
  - Version control (Git)
  - Monitoring & observability (Grafana, Kibana, CloudWatch)
  - Networking fundamentals
  - Automations, Security



Developers these days...

DIFFERENT TYPES OF PROJECTS

TEAM D



TEAM A

TEAM B

TEAM C

Developers these days…

COMPARING SITUATION

Area	~2015	~2025
Depth of Knowledge	Core Java and one framework, relational DBs, SQL, testing…	Expected to know a **wider tech stack**, including DevOps basics, Python, asynchronous systems basics (Kafka, etc.), NoSQL databases…
Learning Curve	Companies often offered more ramp-up time	Expected to be **productive quickly**, often self-driven learning
Tooling	Fewer tools, simpler stack	Complex ecosystem (cloud, containers, observability, etc.) — **tool fatigue is real**
Entry barrier	Easier — fewer techs to learn	**Higher** — juniors often need knowledge of **full development lifecycle**, frameworks, tools
Competition	Fewer bootcamps, less global competition	**Global remote competition**, bootcamp grads, AI-assisted developers
Keeping up	Slower evolution of tools	Faster — **monthly changes in tools/libraries/cloud platforms**
AI	Not present	**AI tools (e.g., Copilot)** create pressure to learn **meta-skills** (problem-solving, code review, prompt engineering)
DevOps/CI-CD	Not expected	**Knowledge of CI/CD tools** (Jenkins, GitHub Actions), Docker, Cloud & IaC basics…
Soft Skills	Less emphasized	Strong focus on **communication, collaboration**, agile mindset, **delivering value**
Mindset	Focus on mastering Java	Focus on **adaptability, continuous learning**, and tooling around Java

Developers these days...

GenAI

Developers these days…

Lo/No-Code, Serverless

We are still programming, but the scope of the technologies is much wider!

New (exciting) trends are here, and more is yet to come…

What should we do?

Developers these days…

QUESTIONS…

Should I even be interested in learning all these new technologies?

Can’t I just stay Java developer? (Code + Docker and that’s it!)

Do I have not only to be a Developer, but also a DevOps & Cloud expert?

Components of the modern *Cloud Applications*

Components of the modern Cloud applications…

CLOUD

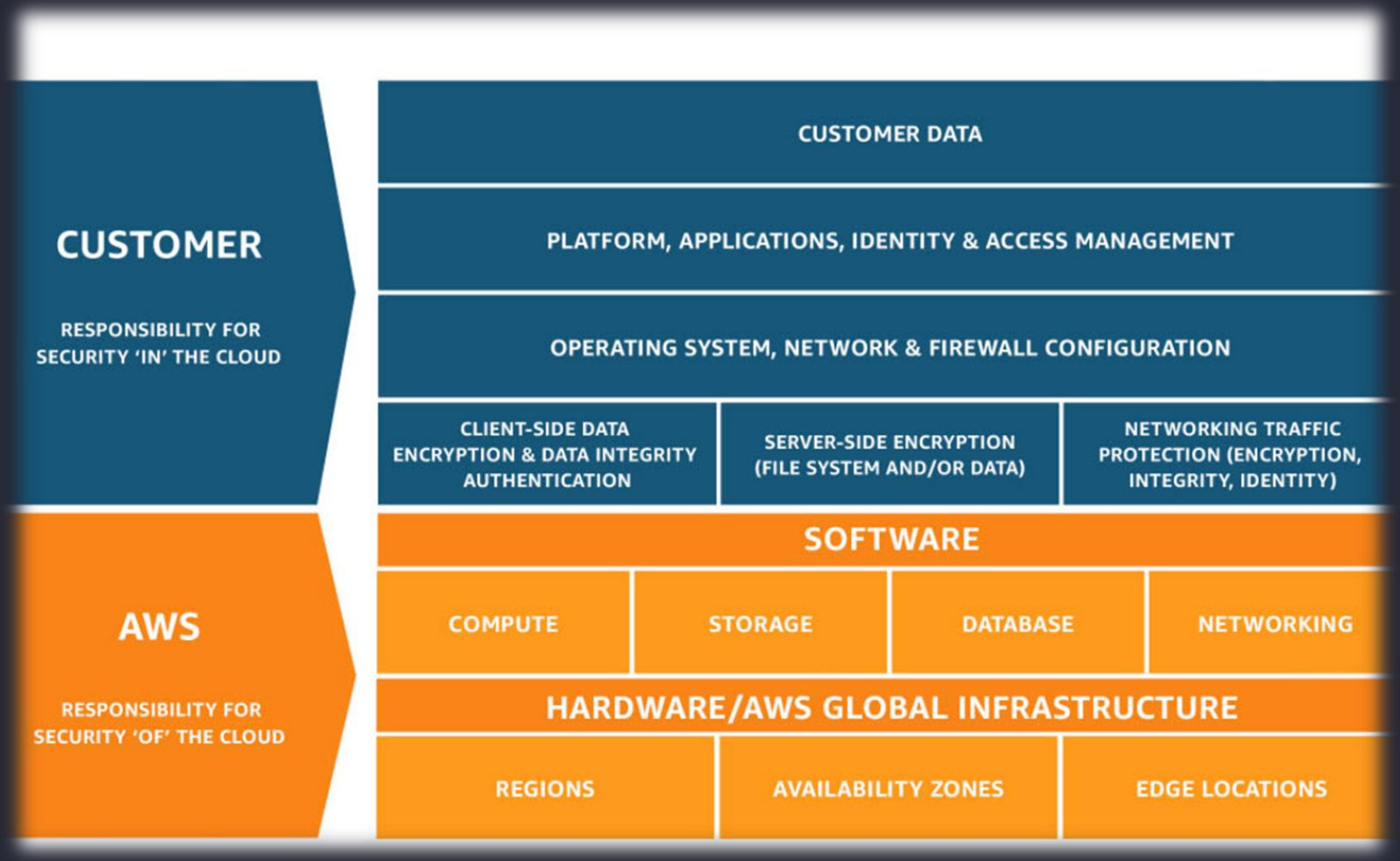
- **Ready-to-use infrastructure**
  - No upfront costs
  - **Pay-as-you-use**
  - No custom servers on-premise
- **Native services** addressing most of the IT use cases
  - Data storage
  - App hosting
  - Big Data, Data Analysis, Messaging…
- **Built-in features**
  - Encryption
  - Virutal networking
  - Security
  - Monitoring & Alerting
- **High Availability & Scalability**

Google Cloud

Azure

Components of the modern Cloud applications…

SHARED RESPONSIBILITY MODEL



Components of the modern Cloud applications…

APPLICATION

- **Standard case – Distributed microservices architecture…**
  - REST API
  - GraphQL
  - Event-Driven-Architecture, Kafka…
- **Dockerized Spring Boot Application deployed to the Cloud…**
  - Virtual Machine
  - Dedicated Service / Platform?
- **Serverless…**
  - We just provide the code…
  - **Acutally, there are servers – we just don’t care**

Components of the modern Cloud applications…

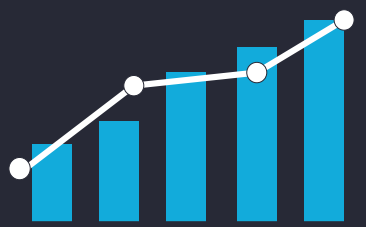
DATA

- Databases
  - SQL, NoSQL…
- Credentials, TLS certificates, encryption…
- Standard application configuration (parameters)
- Different kinds of objects…
  - Photo uploads
  - Documents, etc.

Components of the modern Cloud applications…

MONITORING & ALERTING

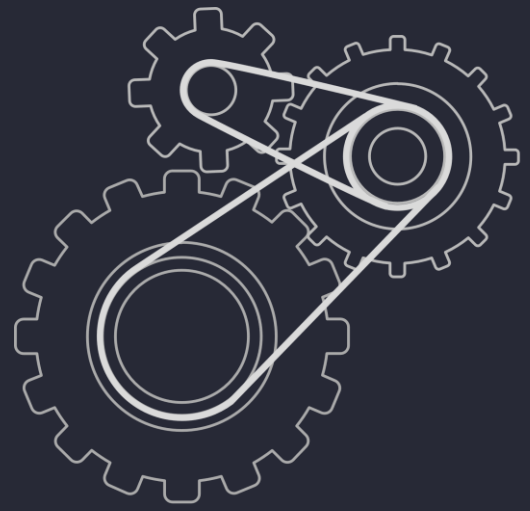
- **Application level monitoring**
  - Prometheus metrics & alerts
  - Grafana dashboards
  - Kibana logs
- **Cloud infrastructure monitoring**
  - AWS CloudWatch metrics & alerts
  - AWS CloudWatch Dashboards
  - AWS CloudWatch logs
- Automated reactions to alerting state
  - Autoscaling
    - „If CPU usage > 70% -> add additional virtual machine”
    - „If disk space too low in DB -> increase storage”
  - Integration with a ticketing system



Components of the modern Cloud applications…

INFRASTRUCTURE

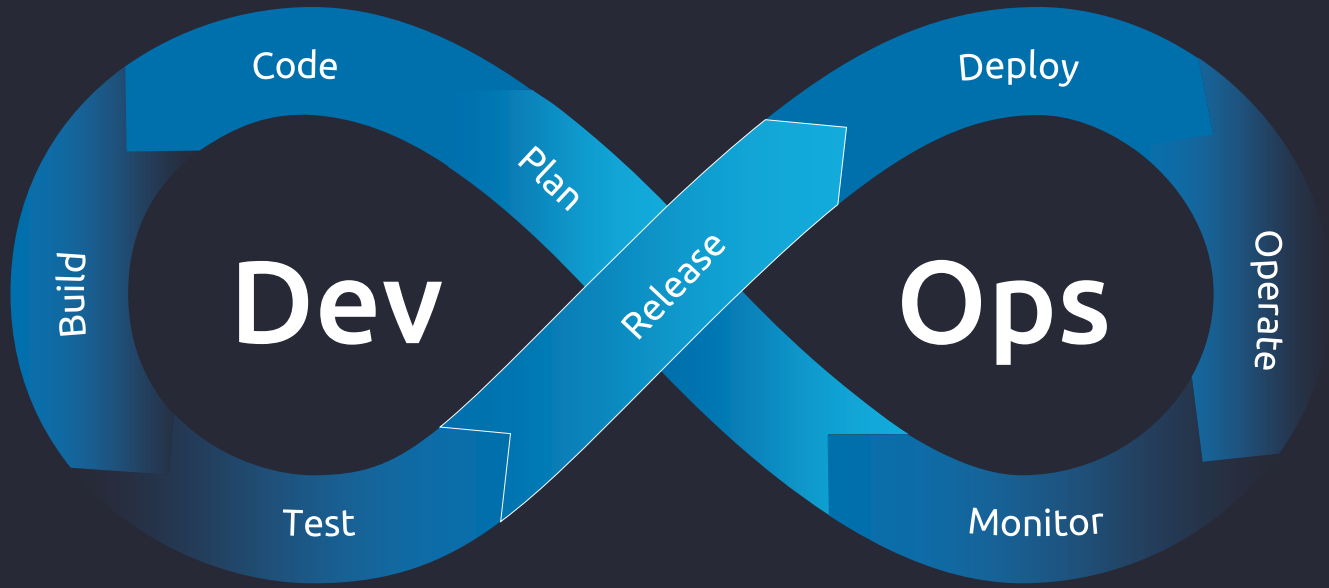
- So many resources in the Cloud…
  - Virtual machines, database services, networking, storage, compute…
- **Infrastructure as a Code (Terraform, CDK…)**
  - Write once
  - Deploy to multiple environments
  - Under control!
    - Code reviews
    - Testing
    - Versioning



Components of the modern Cloud applications…

CICD

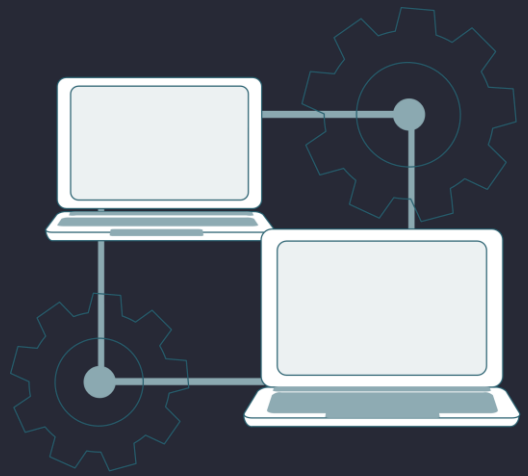
- Pipelines:
  - Builds
  - Tests
  - Deployments
- Jenkins, GitHub Actions, GitLab…
- Quality assurance
  - DAST
  - SAST
  - Dependencies checks



Components of the modern Cloud applications…

AUTOMATIONS

- Using CICD / serverless solutions (Lambda)
  - Automated reactions to the situation in our Cloud environment…
    - Raise ticket when someone removes encryption key accidently
    - Raise ticket if somebody tries to change some access policy
  - Automated renewal of the TLS certificates…
  - Cost optimization…
    - Delete test environment for weekends
  - Automatically extend Confluence page after deployment
    - Who? When? What?
    - Tracking…
  - Integration of multiple APIs / ready-to-use actions
    - Or rest can be written in Bash / Python…



Components of the modern Cloud applications…

ADDITIONAL FACTORS

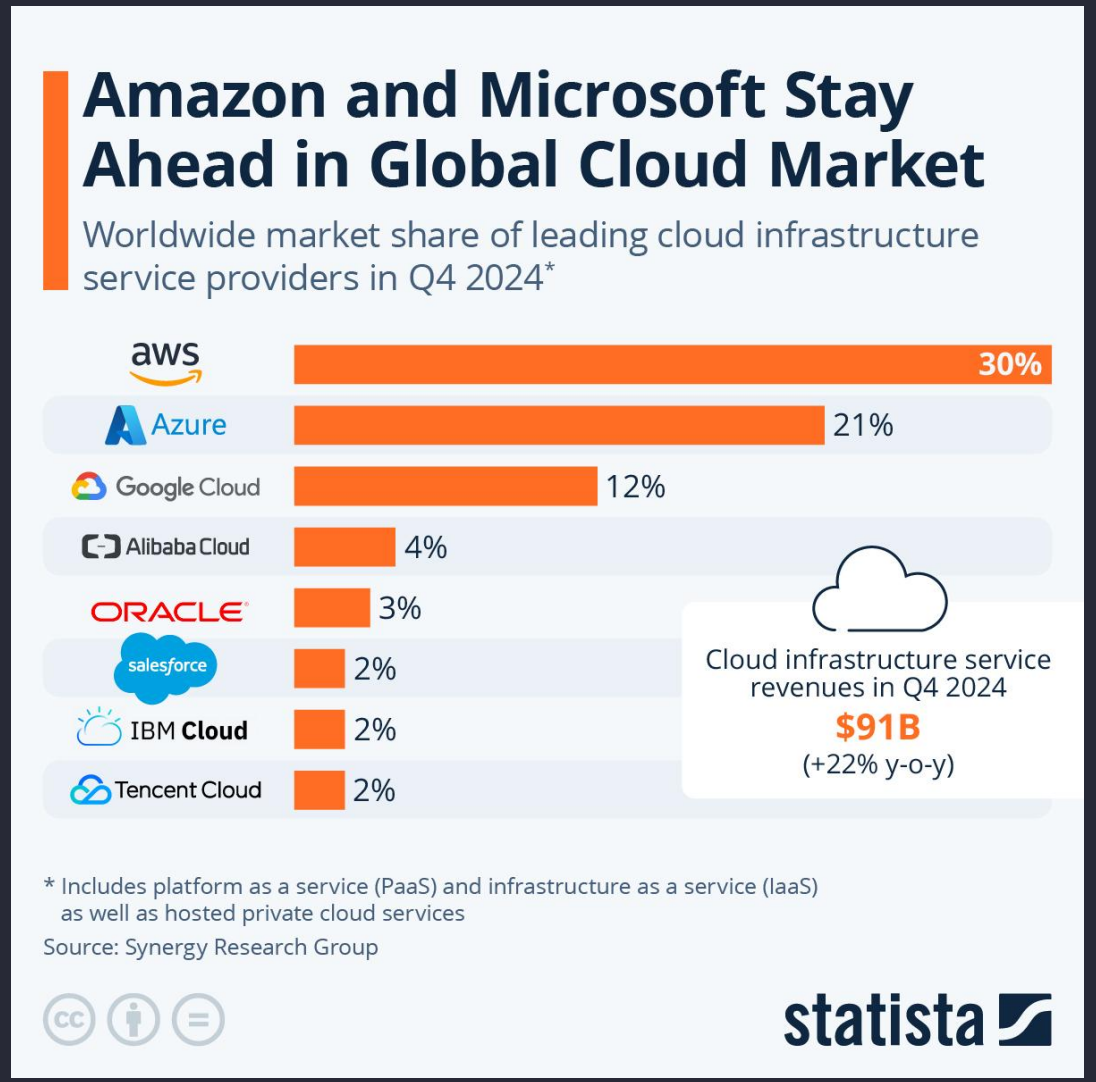
- FinOps
  - Infrastructure cost tracking, analysis / reports…
  - FinOps-by-design, rightsizing resources, autoscaling…
- Security
  - Continuous Monitoring & Logging…
  - Overall awareness, OWASP, encryption, secure communication, networking…
  - Access control, least-privilage…
  - Disaster recovery & resiliency

But okay… I thought we are going to talk about Spring Boot application deployment to AWS…

AWS Recap

Basics

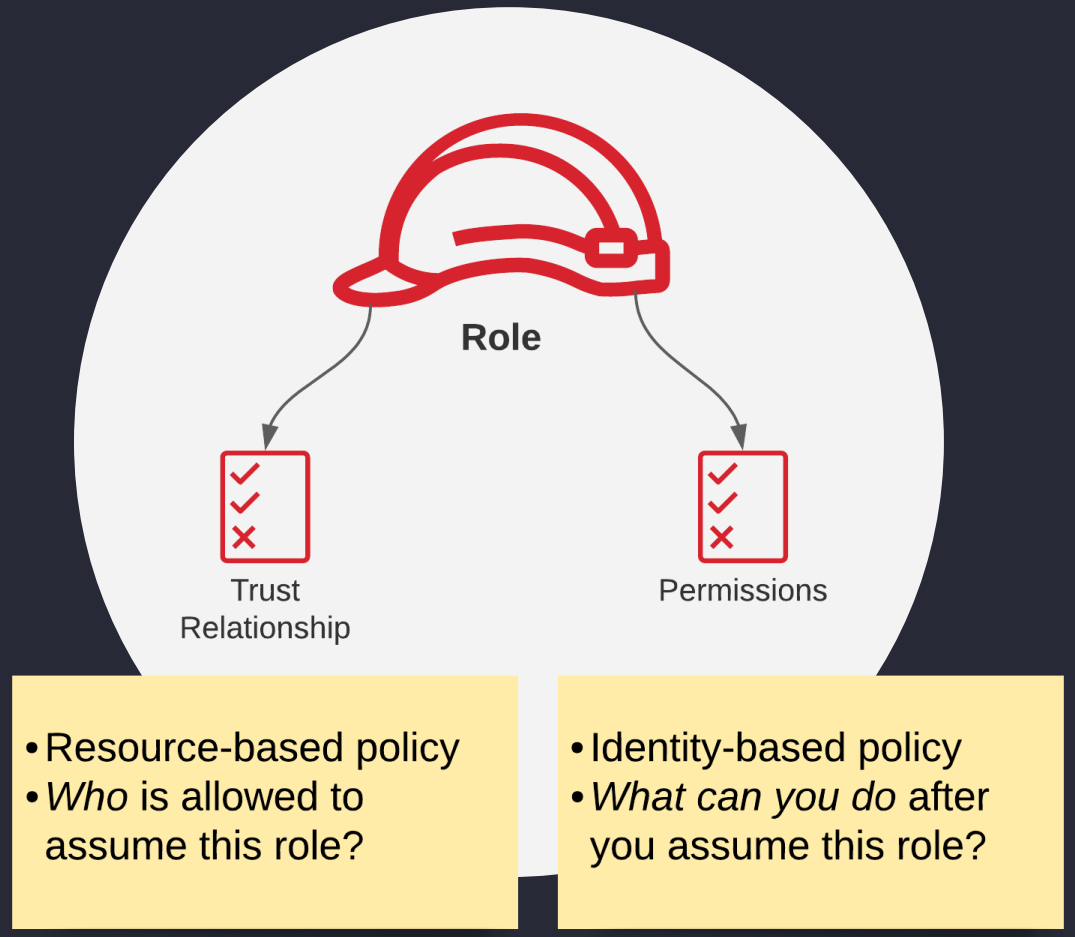
- Most popular Cloud provider
- Great support
- Most companies already have partnership with AWS
- All standard Cloud benefits
  - Dedicated services
  - Globally available
  - High availability
  - AWS support



AWS Recap

IAM – Identity Access Management

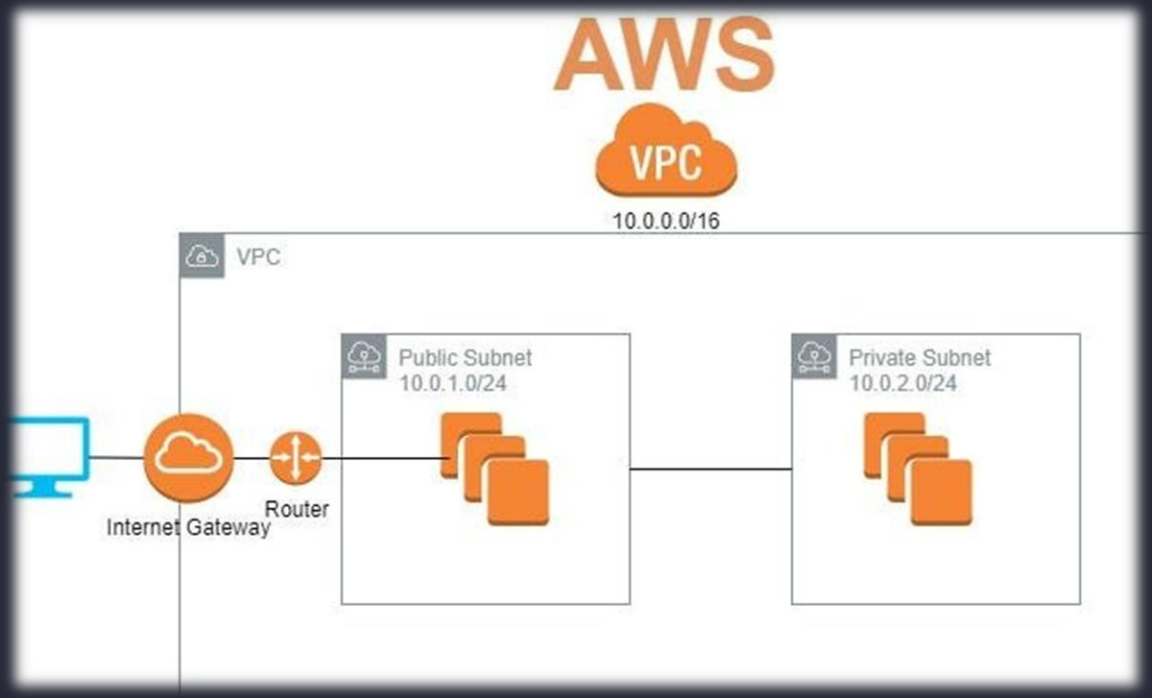
- Users
- Groups
- Policies
- Roles



AWS Recap

VPC – Virtual Private Cloud

- Separation of AWS resources on the **network level**
- We can create **virtual network** containing:
  - Private / public subnets
  - Internet Gateway
  - Route tables
  - (…)
- Secure access
  - Security groups – attached to all network resources (enabling inbound / outbound traffic)
  - ACL (Access Control List)



AWS Recap

EC2 – Virtual Machine

- A virtual machine on which we can run any process / script / application, e.g.:
  - Java application
  - MySQL database
  - Script removing outdated entries from our database
- Maintenance overhead

AWS Recap

Lambda

- A serverless solution – you provide code (Python, Node.js, Java, etc.), and AWS runs it!
- Max execution time 15 minutes
- Great for:
  - Event-driven applications
  - Consuming REST API requests
  - Consuming events, SQS messages, Kinesis data…
  - Reacting to alerts in AWS (internal state change & reaction)
- Scales automatically
  - No maintenance overhead

AWS Recap

CloudWatch

- Centralized monitoring in AWS
  - Logging
  - Alerting
  - Dashboards
  - Metrics
- Most AWS services push standard, ready-to-use metrics to Cloud Watch

AWS Recap

Secrets Manager & Parameter Store

- Secrets Manager keeps our credentials
- Parameter Store keeps our standard app configuration

AWS Recap

RDS & DynamoDB

- Managed AWS database services
- RDS – Relational Database Service (e.g. PostgreSQL, MySQL)
  - Traditional, relational database
  - Fixed schema
  - Vertical (scale up instance size)
- DynamoDB NoSQL Database Service
  - Serverless, pay-per-request (or provisioned throughput)
  - Scales automatically (horizontal scale out with partitions)
  - Massive read/write throughput with low latency
  - Simple key-value or document access patterns
  - **Not good for strongly relational data**
- Cool built-in features
  - Automated backups
  - Encryption
  - High availability, Multi-AZ

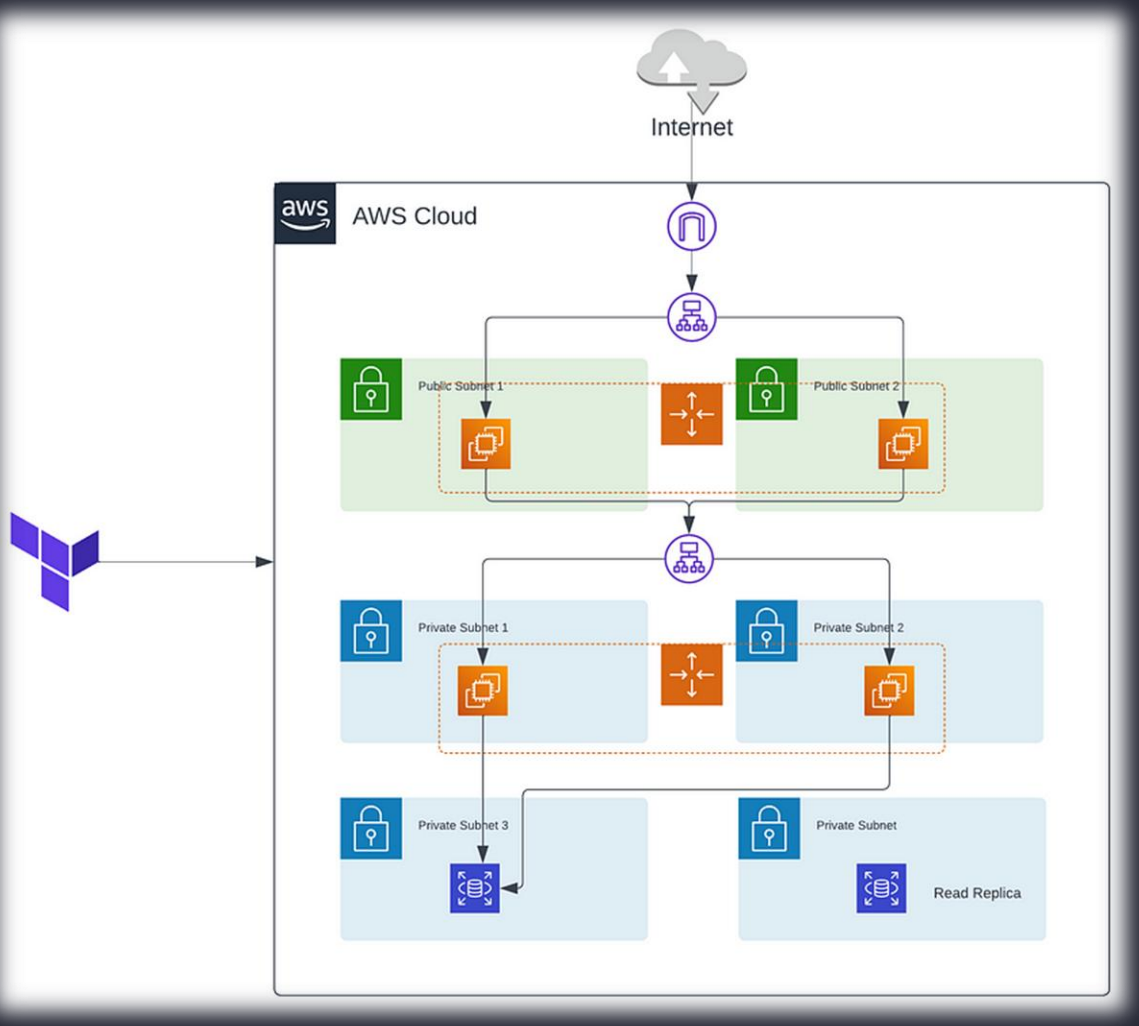
AWS Recap

Load Balancer

- **Evenly distributes incoming network traffic** across multiple targets (servers, instances)
- If one target is down, it will be removed and autoscaling can automatically initialize a new server to **maintain high availability**

How to deploy application to Cloud?

STANDARD ARCHITECTURE



How to deploy application to Cloud?

WHERE CAN I DEPLOY MY APPLICATION?

- AWS offers multiple services & platforms…

Service / Platform	Best Fit Use Case	Notes
Amazon EC2	Full control needed over environment or legacy systems	High flexibility; **high management overhead**
Amazon EKS	**Kubernetes-based** microservices requiring portability and full control	Great for teams already using Kubernetes; more complex to set up
AWS ECS (Fargate)	Containerized microservices with **minimal infrastructure management**	Serverless containers; good for small to medium-scale container-based apps
AWS Lambda	Event-driven microservices, serverless workloads, quick prototyping	Pay-per-request; no server management; limited to 15 mins execution time
Elastic Beanstalk	Simplified deployment of web apps and services (Java, .NET, Python, etc.), you push raw code	PaaS abstraction; good for fast deployment of monoliths or microservices with less ops, small apps deployed asap
App Runner	Deploy containerized applications directly from source or image repo	Simple CI/CD; abstracts infrastructure; great for smaller apps or MVPs
API Gateway + Lambda	Backend APIs with fine-grained traffic control and integrations	Ideal for microservices architecture with REST/HTTP APIs

How to deploy application to Cloud?

ECS (Elastic Container Service) Fargate

- AWS-native platform
  - You provide **Docker image** and basic setup (memory, networking, autoscaling policies, etc.)
- Serverless
  - **AWS provides and maintains the infrastructure**
- Easier to setup comparing to EKS (Elastic Kubernetes Service)
- Basically, if we have a **dockerized application image,** we can simply **host it in the ECS Fargate**
  - Fargate will spawn multiple tasks in paralel, each task hosting the same image of our application
  - When traffic increases – Fargate creates more tasks
  - When traffic goes down – Fargate removes obsolete tasks
- Rolling update deployment
  - Fargate launches new, waits for them to become healthy, and only then gradually stops the old tasks

How to deploy application to Cloud?

ECR – Elastic Container Registry

- Store for our application Docker images
- ECS Fargate consumes images out of it to run them in the tasks

How to deploy application to Cloud?

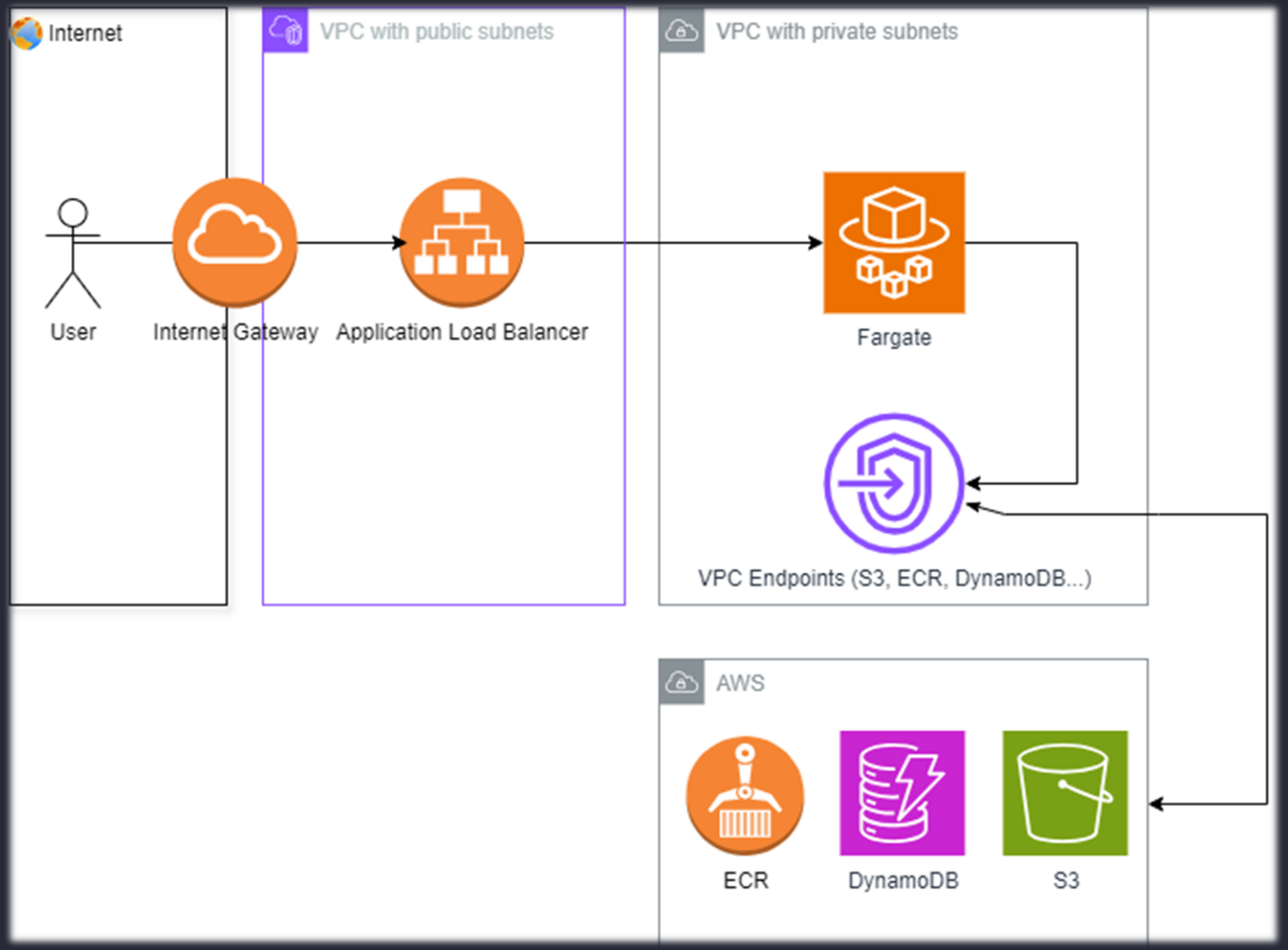
OUR APPLICATION CONCEPT

- Backend application (microservice) that allows collecting multiple measurements for devices, e.g.: CPU usage, free memory, etc.
- Java, Spring Boot application
- Integration with AWS DynamoDB, Parameter Store, Secrets Manager via **AWS Java SDK**

```
{
    "measurements": [
        {
            "timestamp": 1710147050538,
            "value": 80.5,
            "type": "cpu"
        },
        {
            "timestamp": 1710147126802,
            "value": 510.19,
            "type": "memory"
        }
    ]
}
```


How to deploy application to Cloud?

OUR APPLICATION CONCEPT



Let’s take a look at the reference application!

Bonus!
Reference app repository & step-by-step instruction on how to setup everything we demonstrated during this lecture!

Summary

FINAL WORDS

- It takes ~15 minutes to **setup this infrastructure in AWS using Terraform & CICD**
  - **Another 15 minutes** to build, test & **deploy dockerized image to the ECS Fargate**
- The Spring Boot application has one REST endpoint, but it could be much bigger – there would be no difference at the core of the infrastructure setup.
- Eventually, we have **a full working setup:**
  - CICD
  - Monitoring & Alerting infrastructure
  - Dockerized app
  - Terraform IaC
- Before going to PROD, we would like to:
  - Set TLS certificates at every path (load balancer, Kibana, etc.)
  - Implement alerts
  - Improve autoscaling
  - Configure Dependabot / SonarQube
  - (…)

Thank you! Questions?

About Capgemini

Capgemini is a global business and technology transformation partner, helping organizations to accelerate their dual transition to a digital and sustainable world, while creating tangible impact for enterprises and society. It is a responsible and diverse group of 340,000 team members in more than 50 countries. With its strong over 55-year heritage, Capgemini is trusted by its clients to unlock the value of technology to address the entire breadth of their business needs. It delivers end-to-end services and solutions leveraging strengths from strategy and design to engineering, all fueled by its market leading capabilities in AI, generative AI, cloud and data, combined with its deep industry expertise and partner ecosystem. The Group reported 2024 global revenues of €22.1 billion.

Get the future you want | www.capgemini.com

Capgemini