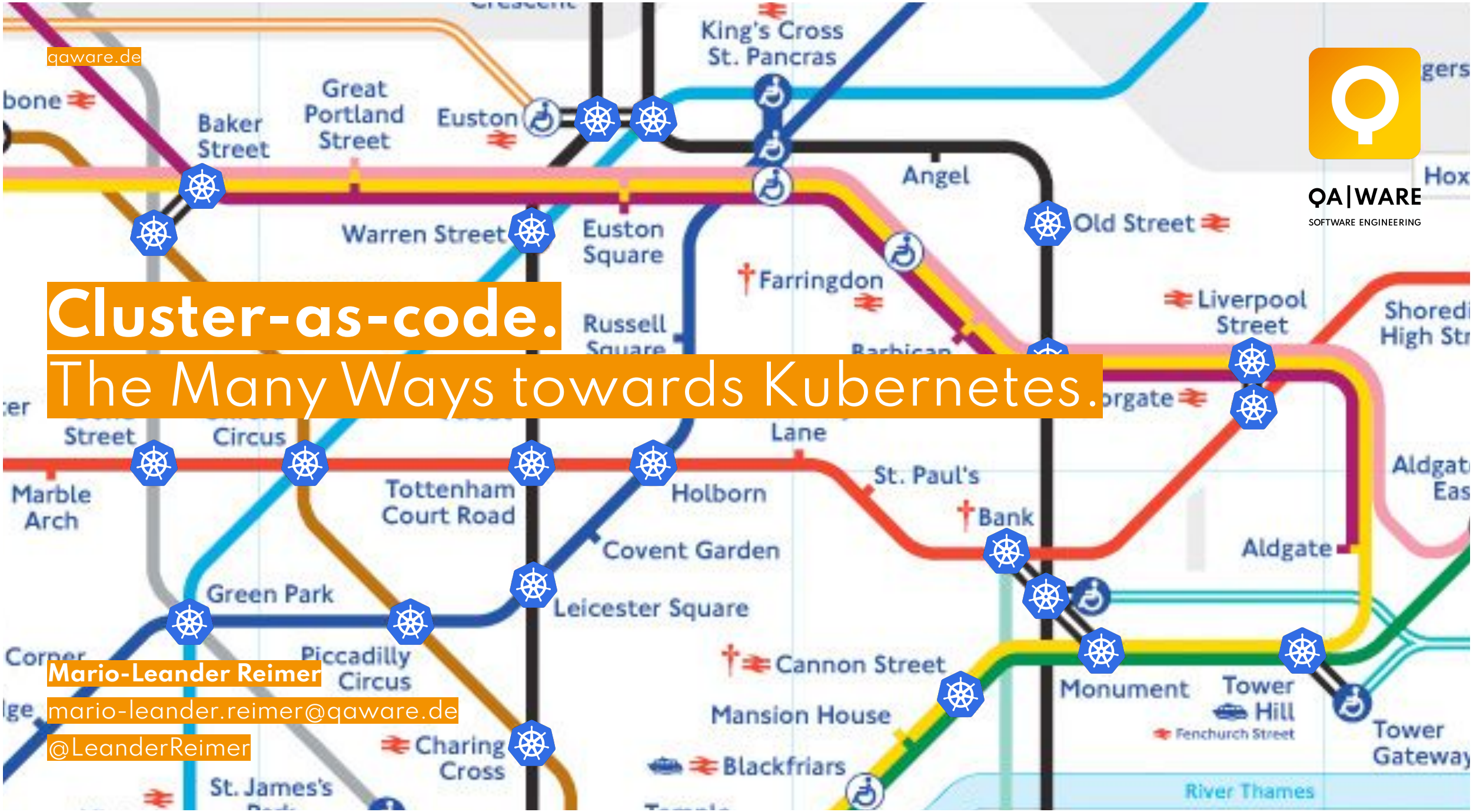


Cluster-as-code.

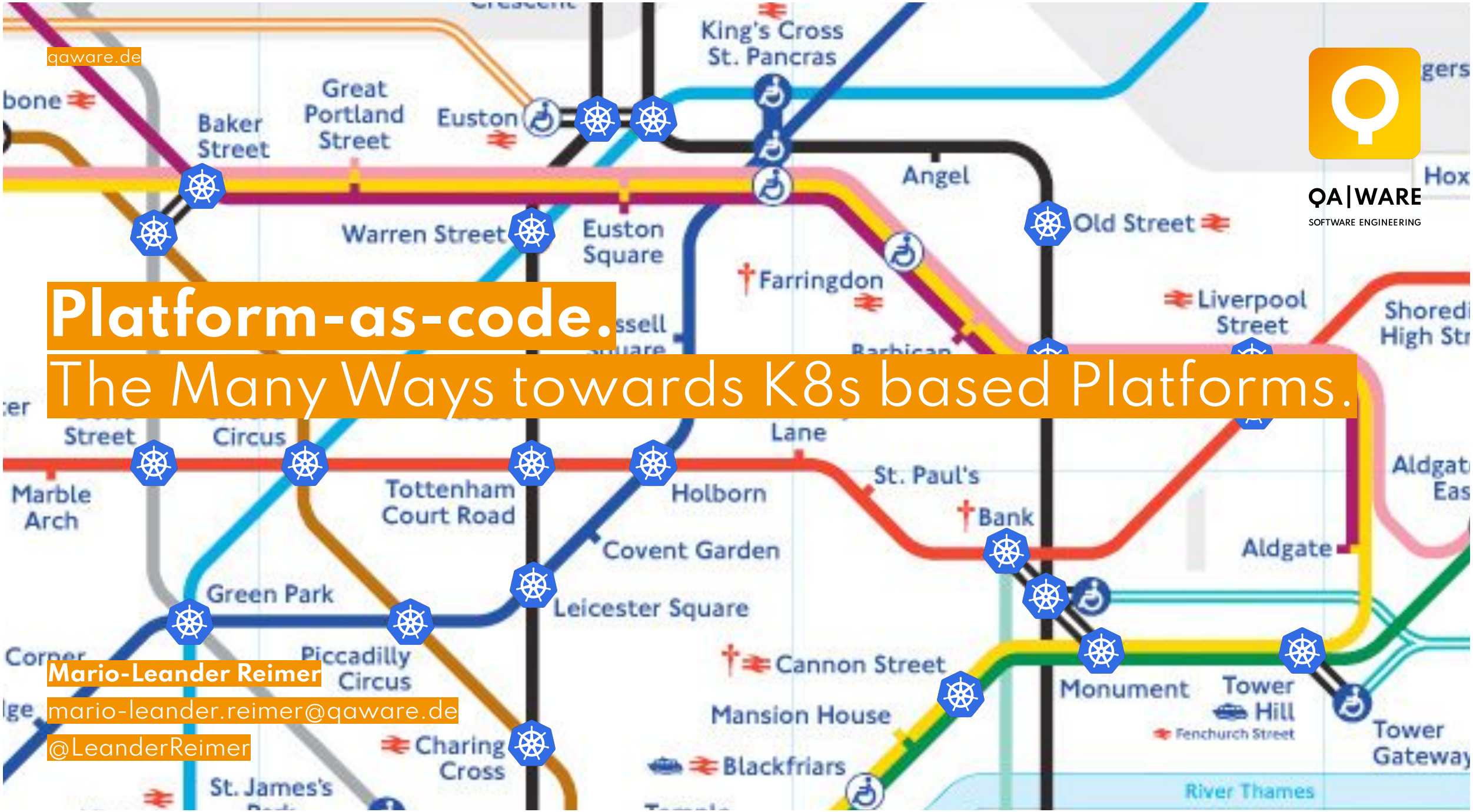
The Many Ways towards Kubernetes.

Mario-Leander Reimer
mario-leander.reimer@qaware.de
@LeanderReimer

10:04 PM - 27. November 2017

5:38 PM - 24. Februar 2019

QA|WARE

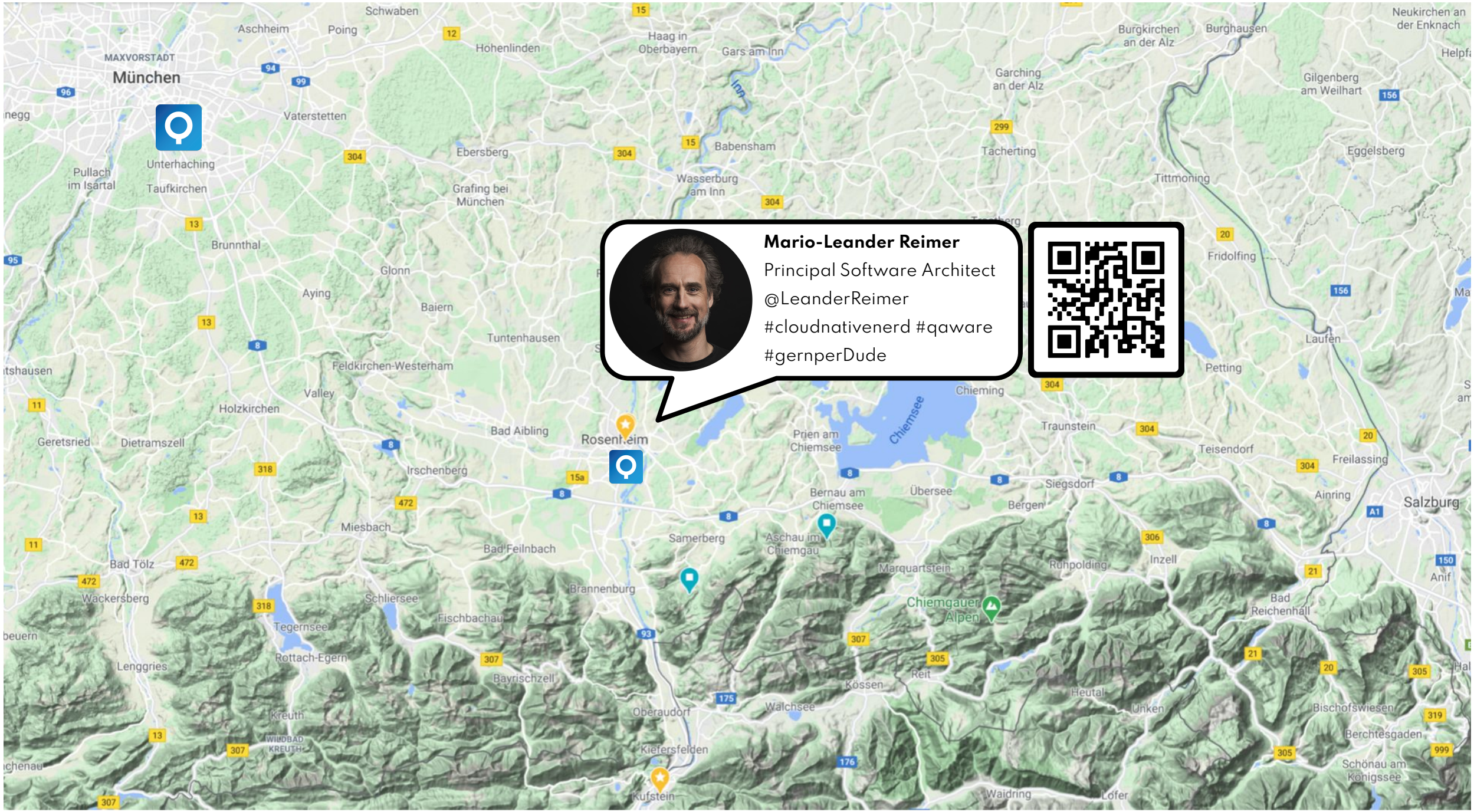


Platform-as-code.

The Many Ways towards K8s based Platforms.

Mario-Leander Reimer
mario-leander.reimer@qaware.de
@LeanderReimer

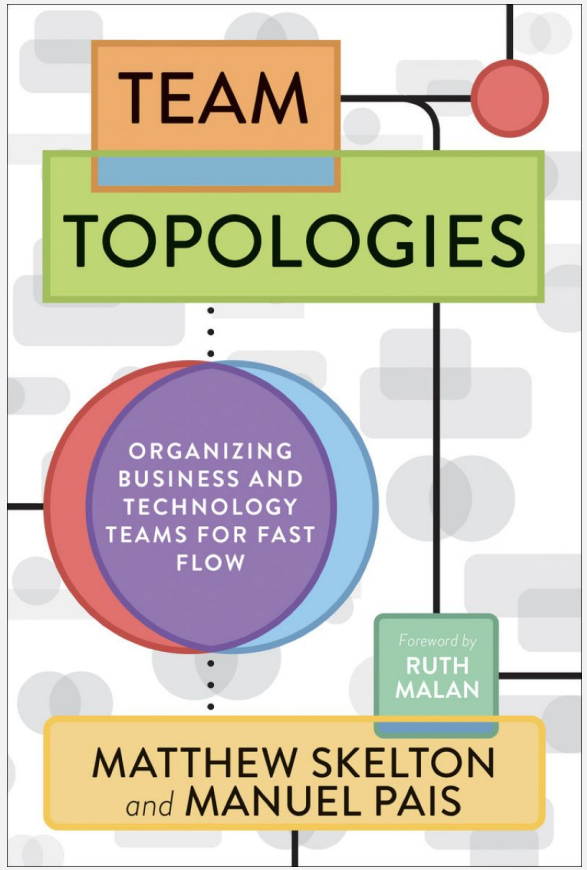
QA|WARE
SOFTWARE ENGINEERING


Mario-Leander Reimer
Principal Software Architect
@LeanderReimer
#cloudnativenerd #qaware
#gernperDude
München
Rosenheim
Unterhaching
Chiemsee
Salzburg

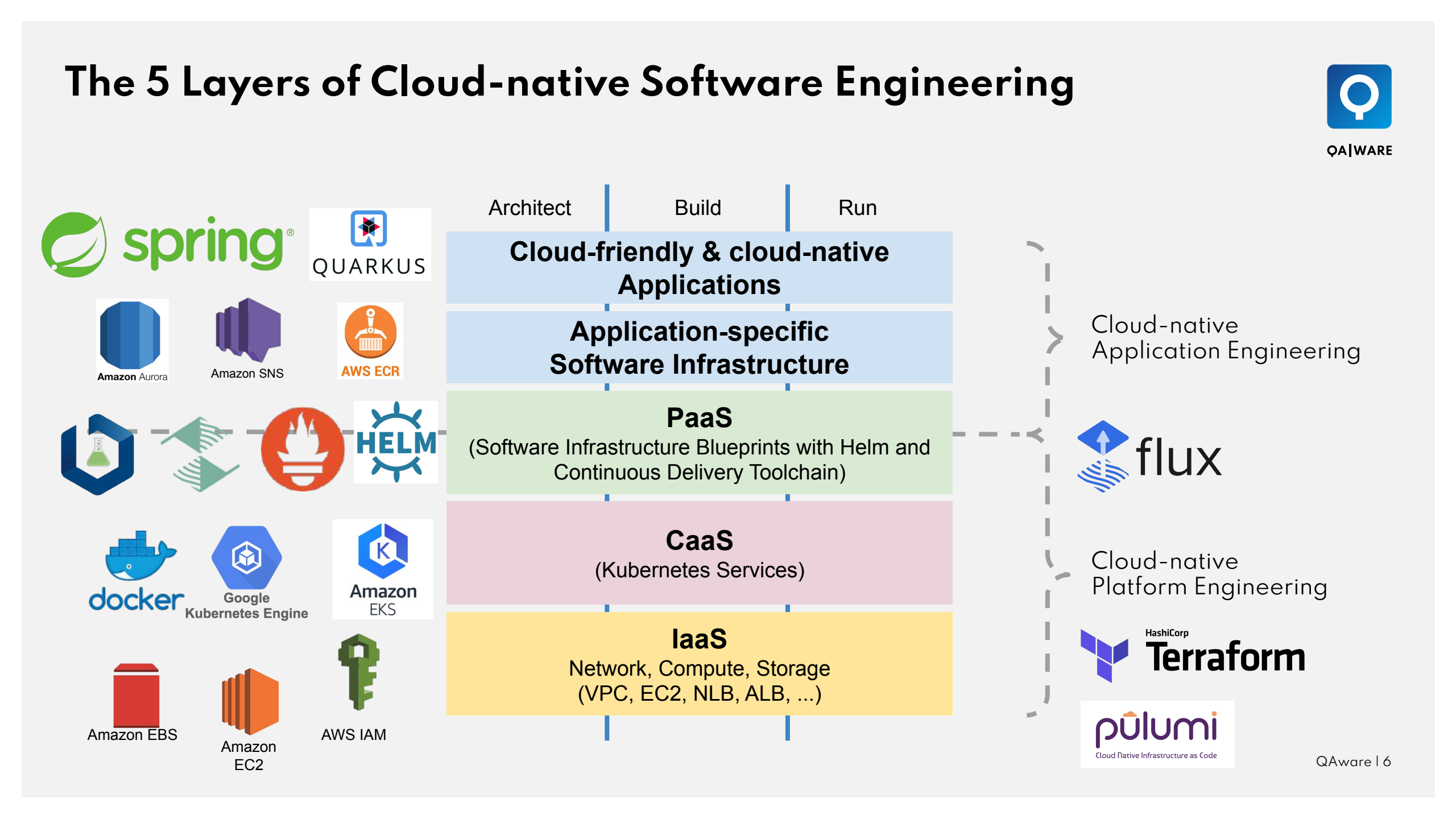
Too much cognitive load easily is a bottleneck for fast flow and high productivity for many DevOps teams.

QA|WARE

- **Intrinsic Cognitive Load**
  Relates to fundamental aspects and knowledge in the problem space (e.g. used languages, APIs, frameworks)
- **Extraneous Cognitive Load**
  Relates to the environment (e.g. console command, deployment, configuration)
- **Germane Cognitive Load**
  Relates to specific aspects of the business domain (aka. „value added“ thinking)



The 5 Layers of Cloud-native Software Engineering



A Platform team and its engineers are a key enabler for high productivity of stream-aligned DevOps teams.

QA|WARE

- Responsible to build and operation a platform to enable and support the teams in their day to day development work.
- The platform aims to hide the inherent complexity to reduce the cognitive load for the other teams.
  - Standardization
  - Self-Service
- Fully automated software delivery is the goal!

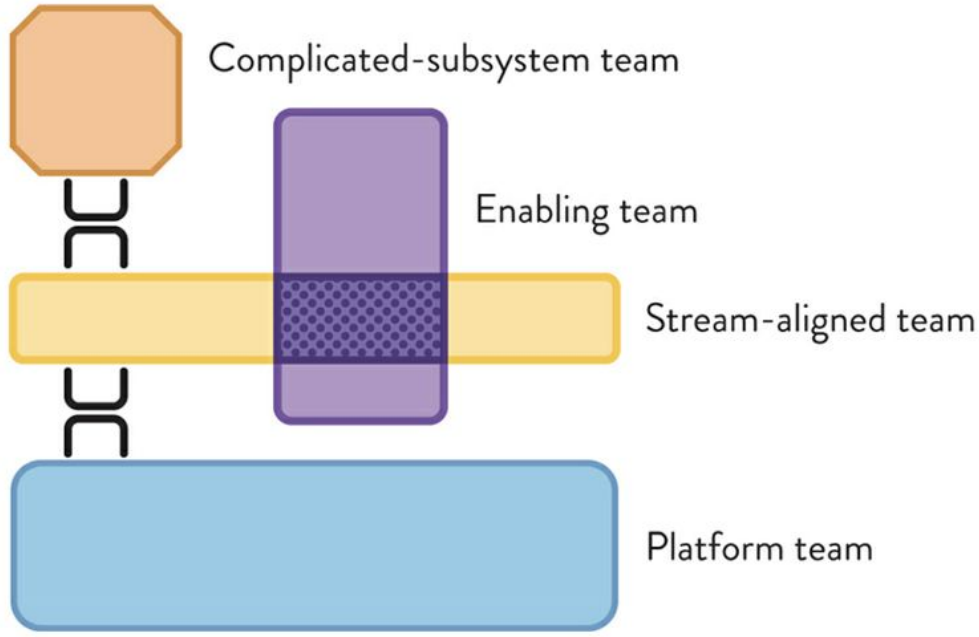
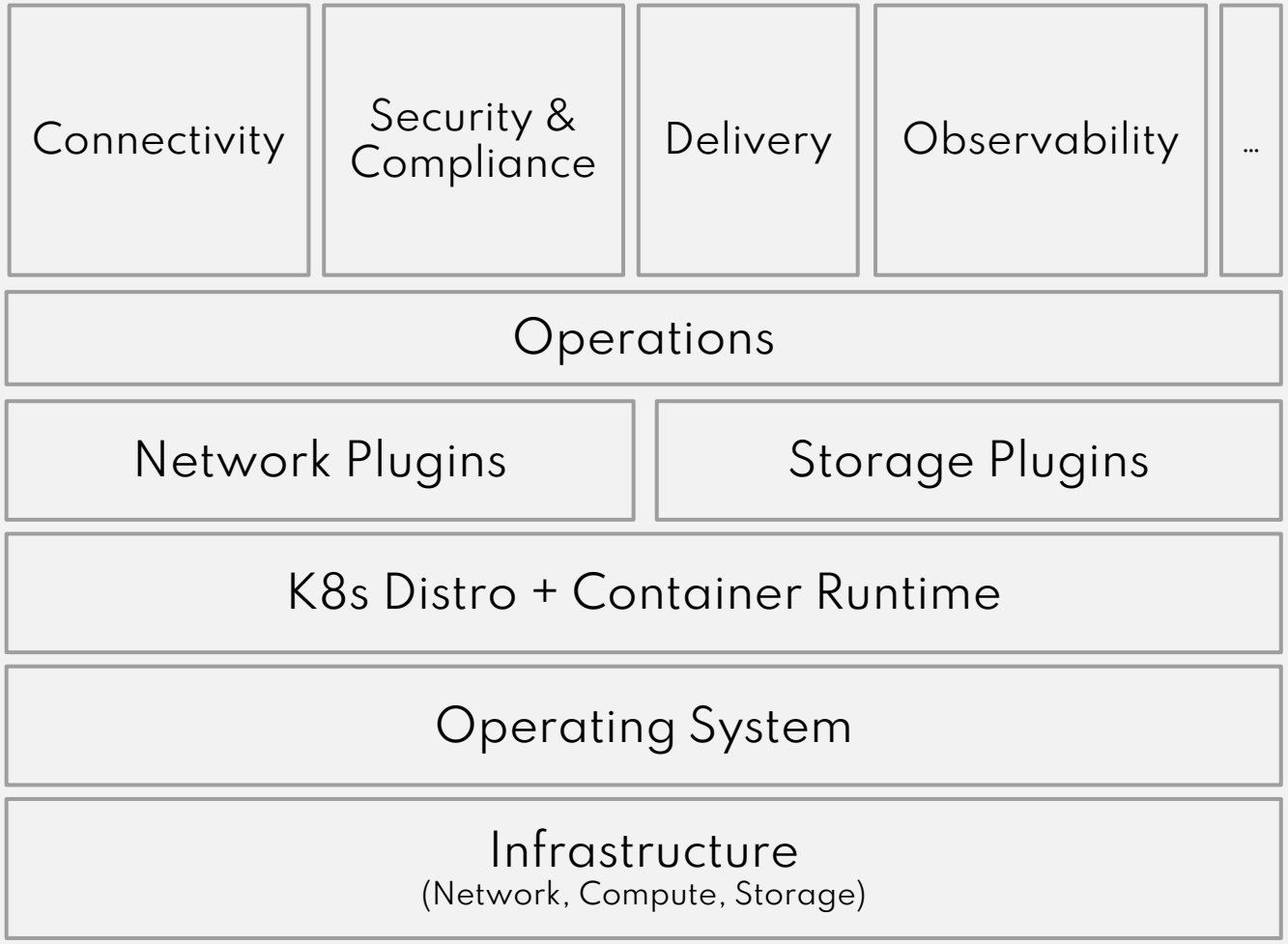


Figure 5.1: The Four Fundamental Team Topologies

https://hennyportman.wordpress.com/2020/05/25/review-team-topologies/

Layers and Components of a Kubernetes based Platform



Flux, Flagger, ArgoCD, Tekton, Traefik, LinkerD, Certificate Manager, Sealed Secrets, Kyverno, Prometheus, Loki, Temp, etc.

KureD, Velero, Karpenter, Quotas, Capsule, Metrics Server, Cluster Autoscaler, Dashboard

Cilium, Weave, Calico, CSI, Longhorn, Rook etc.

CRI-O, ContainerD or Docker

Ubuntu, Fedora CoreOS, Flatcar, Amazon Linux, etc.

Physical / Virtual / Private Cloud/ Public Cloud

GitOps allows to easily build and run a Kubernetes based Platform at scale.

- Single source of truth for the platform and application configurations
- Resource-transparent: if you can store the definition as YAML, it works
- Accountability and visibility of changes
- Clear history of changes
- Easy to build processes for approving configuration changes
- Easy to revert configuration changes
- Potential Disaster Recovery tool
- Several powerful tools available, e.g. Flux or ArgoCD

qaware/cloud-native-explab

QA|WARE

Use the right CLIs tool for the job!

gcloud

flux

eksctl

The Terraform Way

- Well-known infrastructure as code tool
- Build, change, and version cloud and on-prem resources safely and efficiently
- Uses HCL as configuration language
- Typical workflow: Write, Plan, Apply
- Static analysis, checking and linting available
- Some libraries for integration testing exist
- 1000+ providers available
  - AWS, GCP, Azure, Kubernetes, Helm,
  - github.com/fluxcd/terraform-provider-flux

```
# EKS cluster setup
module "eks" {
  source       = "terraform-aws-modules/eks/aws"
  cluster_name = local.cluster_name

  vpc_id = module.vpc.vpc_id
  subnet_ids      = module.vpc.private_subnets

  cluster_endpoint_private_access = true
  cluster_endpoint_public_access  = true

  tags = {
    Environment = "development"
    GithubRepo  = "cloud-native-explab"
    GithubOrg   = "qaware"
  }

  eks_managed_node_groups = {
    default_node_group = {
      create_launch_template = false
      launch_template_name   = ""

      disk_size = 50
      instance_type = "t2.medium"

      asg_desired_capacity          = 3
    }
  }
}
```

Declarative Infrastructure as Code is the predominant approach. So what's wrong with it?

- Nothing? Well, it depends!
- Declarative approaches like Terraform are initially really easy to use.
- However, you still have to learn a new tool and syntax, including the associated ecosystem.
- Modern engineering practices (clean code and architecture, TDD) are not well established.
- Usually, almost no flow control constructs, like loops, conditionals, if-else.
- No support for dynamic sources, like CMDBs.
- Modelling environments can get messy if done wrong and lead to a lot of duplication.

```
module "vpc" {
  source = "../../modules/some-other-tf-source-code"
}

resource "aws_instance" "web" {
  count = format("%.1s",var.instance_type) == "t" ? 1 : 0
}

%{ if <CONDITION> }<TRUEVAL>%{ else }<FALSEVAL>%{ endif }

dynamic "tag" {
  for_each = {
     for key, value in var.custom_tags
     key => upper(value)
     if key != “Name”
  }
  content {
    key   = tag.key
    value = tag.value
  }
}
```

Imperative Tools like Pulumi or Amazon CDK enable modern cloud infrastructure engineering for software developers and SREs.

- Tame overall complexity. Use your favourite language!
- No breach between application development and DevOps engineering.
- One consistent approach to Infrastructure as Code and cloud engineering for Docker, many cloud providers and Kubernetes.
- Easy to apply well-known clean code and general engineering practices to infrastructure code: automation, modularity, testing, and CI/CD.
- Many alternatives:
  - Pulumi (https://github.com/pulumi/pulumi)
  - Amazon CDK (https://github.com/aws/aws-cdk)
  - cdk8s (https://github.com/cdk8s-team/cdk8s)
  - cdktf (https://developer.hashicorp.com/terraform/cdktf)

Pulumi - Cloud Engineering for Everyone.
Modern Infrastructure as Code for Developers and SREs

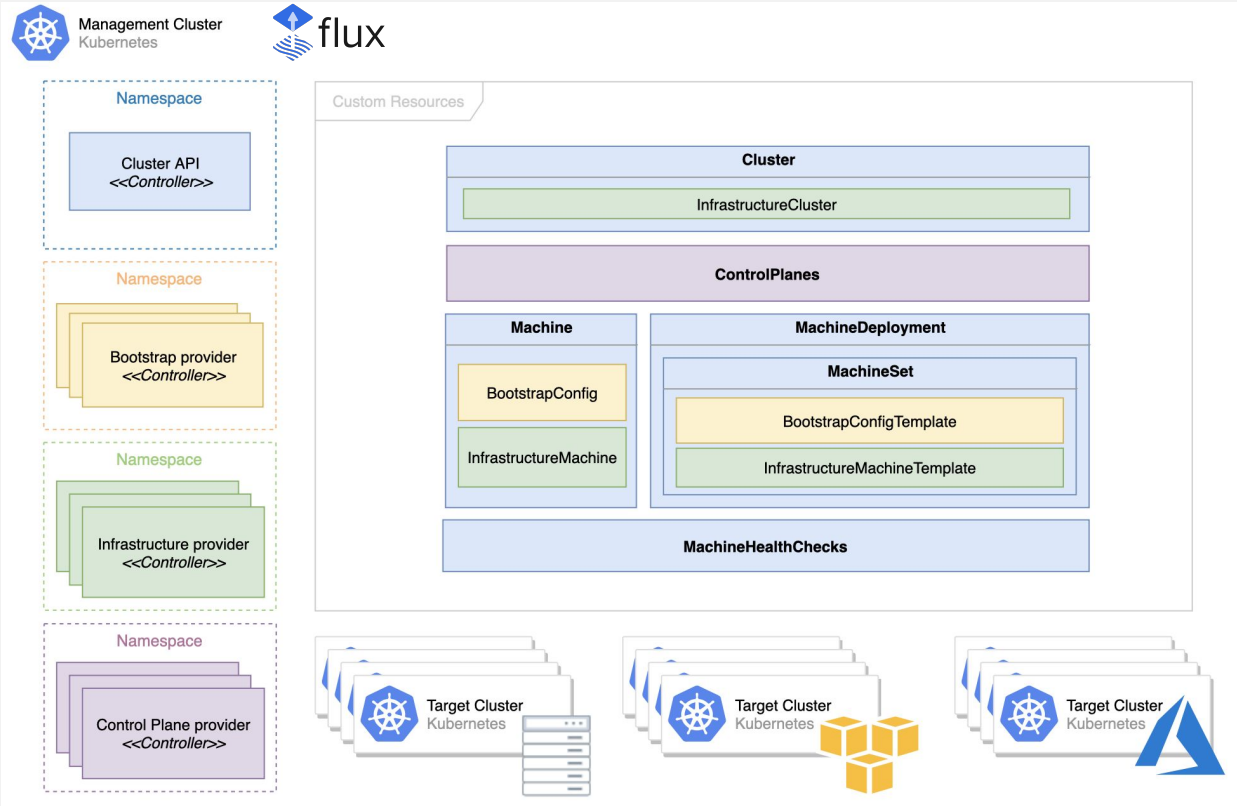
- Rich programmable cloud interfaces with abstractions and reusable packages.
- Apply engineering practices to infrastructure code: modularity, testing, and CI/CD.
- No intermediary formats. Direct usage of provided APIs.
- Several converters available: arm2pulumi, crd2pulumi, kube2pulumi, tf2pulumi
- Possibility to automate Pulumi workflows itself via API, instead of using the CLI.
- Documentation and example resources available
  - https://www.pulumi.com/docs/get-started/
  - https://github.com/pulumi/examples
  - https://www.pulumi.com/registry/packages/kubernetes/
  - https://github.com/pulumi/pulumi-eks

Amazon CDK - Define cloud infrastructure in your favorite programming language and deploy it using CloudFormation

aws Cloud Development Kit

- AWS CDK supports TypeScript, JavaScript, Python, Java, C#/.Net, and (in developer preview) Go
- Many, many advantages (according to their website):
  - Use logic (if statements, for-loops, etc) when defining your infrastructure
  - Use object-oriented techniques to create a model of your system
  - Organize your project into logical modules, share and reuse your infrastructure as a library
  - Define high level abstractions, share them, and publish them to your team and company
  - Testing your infrastructure code using industry-standard protocols and tools
  - Use your existing code review workflow and features such as code completion within your IDE
- Good documentation and example resources available
  - https://docs.aws.amazon.com/cdk/latest/guide/home.html
  - https://cdkworkshop.com
  - https://docs.aws.amazon.com/cdk/api/v1/docs/aws-eks-readme.html
- Currently AWS only, AWS CloudFormation is still present as final output.

The Kubernetes Cluster API Way



- Official Kubernetes sub-project
- Declarative APIs and tooling to provision, upgrade, and operate multiple Kubernetes clusters
- Work in different environments, both on-premises and in the cloud
- Reuse and integrate existing ecosystem components rather than duplicating


THESE ARE THE WAYS.
Which one do you take?

Meetups & Talks before X-Mas

QA|WARE

1.12. in Mainz!

6.12. online!

qaware.de

QA|WARE
SOFTWARE ENGINEERING

QAware GmbH
Aschauer Straße 32
81549 München
Tel. +49 89 232315-0
info@qaware.de


twitter.com/qaware


linkedin.com/company/qaware-gmbh


xing.com/companies/qawaregmbh


slideshare.net/qaware


github.com/qaware