What Is Kubernetes?

- Kubernetes is an open-source container orchestration system for automating the deployment, scaling, and management of containerized applications.
- It was originally developed by Google and is now maintained by the Cloud Native Computing Foundation (CNCF).
- Kubernetes provides a platform-agnostic way to manage and deploy containerized applications, making it possible to run them on a variety of infrastructure, including on-premises, in the cloud, or in a hybrid environment.

Why use Kubernetes?

- Scalability
- High availability
- Self-healing
- Load balancing
- Flexibility
- Automated rollouts and rollbacks
- Reduced operational overhead
- Large and active community

How Kubernetes works?

Kubernetes works by using a master-slave architecture to manage containerized applications:

- The developer uses kubectl command line tool to interact with the Kubernetes master node, which controls worker nodes.
- The worker nodes run the containerized application and communicate with the master node to ensure desired state is met.
- Kubernetes automatically distributes incoming traffic, monitor the running containers, and performs self-healing if needed.
- The developer can scale and update the application through the Kubernetes API server.

What is GitOps?

- GitOps is a way of implementing Continuous Deployment for cloud native applications, it uses Git as a source of truth for declarative infrastructure and application code.
- It uses a pipeline to automatically deploy changes and bring the system to the desired state, defined in a Git repository, and continuously monitors the cluster for drift.
- GitOps also facilitates collaboration, traceability, and rollbacks by using version control in the Git repository.

Why use Kubernetes?

- Scalability
- High availability
- Self-healing
- Load balancing
- Flexibility
- Automated rollouts and rollbacks
- Reduced operational overhead
- Large and active community

gitops

How Kubernetes works?

Kubernetes works by using a master-slave architecture to manage containerized applications:

- The developer uses kubectl command line tool to interact with the Kubernetes master node, which controls worker nodes.
- The worker nodes run the containerized application and communicate with the master node to ensure desired state is met.
- Kubernetes automatically distributes incoming traffic, monitor the running containers, and performs self-healing if needed.
- The developer can scale and update the application through the Kubernetes API server.

Your slide deck

Start writing!

Marp

Markdown Presentation Ecosystem

https://marp.app/

How to write slides

Split pages by horizontal ruler (`---`). It's very simple! 😆

```
# Slide 1

foobar

---

# Slide 2

foobar
```

Declare to use KaTeX in this Markdown

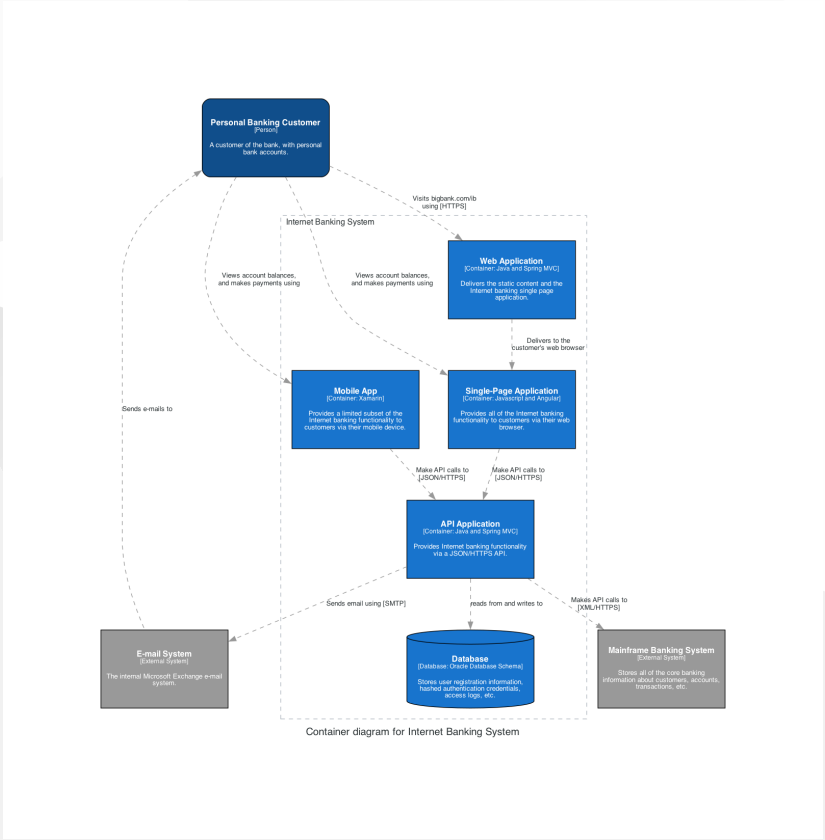
Google
Image

> Blockquote

- xxx
- xxx
- xx

asdasdsa

```
# dasnjlndkjnkasjd
sdbkabskhjdb
1. s
2. 2
3. m
```



Write 100 Words a Day, Every Day for One Month

That's what we do here. We've been doing it since May 2001. Which, by our reckoning, makes us the oldest continuously running daily writing project on the web.

Split backgrounds

The space of a slide content will shrink to the right side.

Split + Multiple BGs

The space of a slide content will shrink to the left side.

Split backgrounds with specified size

Slide 1: Introduction

- Brief overview of the problem of managing infrastructure resources across multiple cloud providers
- Introduction to Crossplane as a solution for multi-cloud management

Slide 2: Key Features of Crossplane

- Composite resources and claims:
- Provider abstraction:
- Multi-cloud management:
- Infrastructure as code:
- Cost management:

Slide 3: How Crossplane Works

- Overview of the installation and configuration process
- Explanation of the components of Crossplane (e.g. Provider, Composite resource, Claim)
- Diagrams or visual aids to help explain the process

Slide 4: Use Cases of Crossplane

- Multi-cloud management:
- Infrastructure as code:
- Cost management:
- Improved security:
- Improved collaboration:
- Container networking and security
- Database as a service:

Slide 5: Benefits of Using Crossplane

- Reduced vendor lock-in:
- Improved efficiency:
- Better cost management:
- Improved security:
- Improved governance:
- Improved collaboration:

Slide 6: Real-world Examples

- Use cases of Crossplane in production
- Examples of companies that are using Crossplane
  ** Adobe
  ** Ticketmaster
  ** GoDaddy
  ** Shopify
  ** OpenAI
  ** Twitch

Slide 7: How to Get Started with Crossplane

- Getting started with Crossplane is relatively simple and can be broken down into the following steps:
- Links to documentation and resources:

Slide 8: Best Practices for Using Crossplane

- Using Crossplane effectively requires following certain best practices:

Slide 9: Crossplane vs (terraform and CDK for Terraform) vs Pulumi

- Terraform vs. Pulumi vs. Crossplane – Infrastructure as Code (IaC) Tools Compared

My story:

The story about a company that is moving their infrastructure and applications from a private cloud to a public cloud, specifically using AWS. They have a mix of different types of applications, including Kubernetes, Cloud Foundry, and VMs. The customer wants to use a unified approach and adopt cloud native technologies and practices in the process. The team has gone through various tools and solutions in their migration journey, including terraform, cloudfromthon, CDK, and now crossplane and argocd. The current system uses AWS Service Catalog and CloudFormation and CDK, but they are now looking to implement crossplane and argocd as a solution.

I have moved from team to team and have encountered teams that

Summary

What is Crossplane?
Crossplane is an open-source multi-cloud control plane that enables users to provision and manage infrastructure resources across multiple cloud providers. It allows users to define infrastructure resources as code, such as a PostgreSQL database instance, and use a single API to provision and manage those resources across different cloud providers. Crossplane uses a concept of composite resources and claims to provide a consistent and unified way to provision and manage infrastructure resources, similar to how Persistent Volumes (PVs) and Persistent Volume Claims (PVCs) work in Kubernetes.

Why use Crossplane?