From Zero to GitOps with AKS

Thomas Thornton

Karl Cooke

Speaker Intro – Thomas Thornton

- **Azure MVP & Microsoft Certified Trainer**
- **Azure Technical Specialist – Kainos in Belfast**
- **Azure Certified**
  - **Azure Solutions Architect**
  - **DevOps Engineer**
  - **Azure Security Engineer**
  - **Azure Administrator**
- **https://thomasthornton.cloud/**
- **Twitter:- https://twitter.com/tamstar1234**

Speaker Intro – Karl Cooke

- **Azure Consultant – Intercept B.V.**

- **Azure Certified**
  - **DevOps Engineer**
  - **Azure Security Engineer**
  - **Azure Administrator**

- **https://irishtechie.cloud**

- **Twitter:- https://twitter.com/Karl_ITNerd**

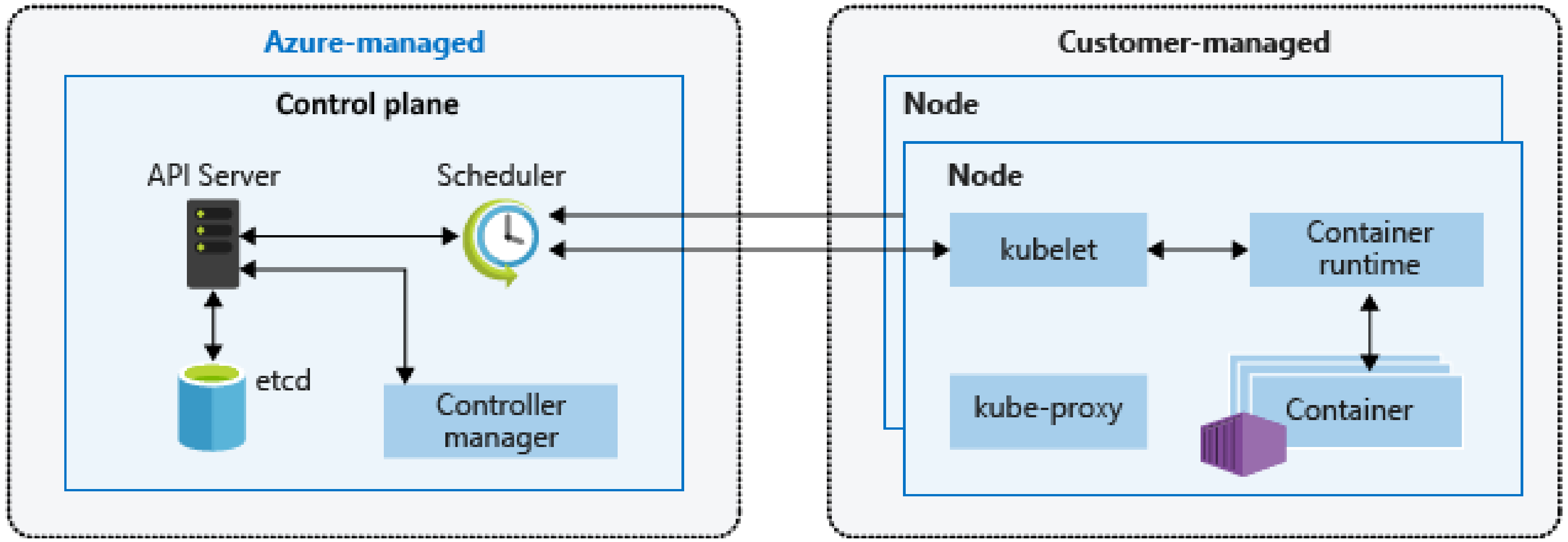
- **Co-organizer @ Limerick DotNet Azure User Group**

Agenda

- What is AKS?
- Azure AKS Architecture
    - Nodes and Node Pools
- Benefits of AKS
- GitOps
- GitOps Tooling
- Fluxv2
- Terraform
- Azure Pipelines
- Demos!
- Wrap-up/Key Takeaways

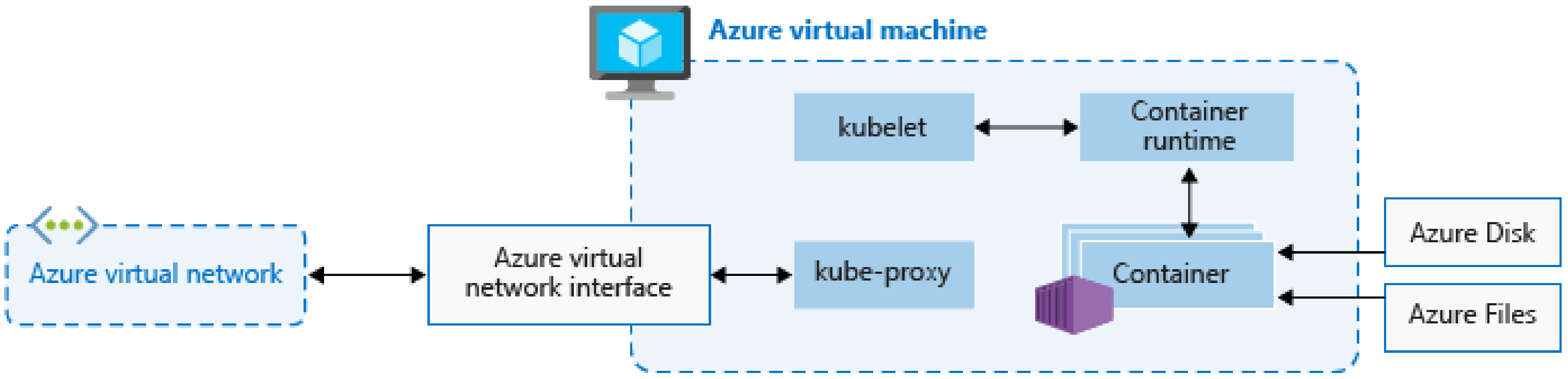
What is AKS?

- Azure Kubernetes Service (AKS) is a managed Kubernetes service
- Kubernetes Master nodes (control plane) are managed by Azure
- You only need to concentrate on the worker nodes



Azure AKS Architecture

Image Reference: docs.microsoft.com



Nodes and Node Pools

Image Reference: docs.microsoft.com

Benefits of AKS

- Scalability (Add additional compute if/when needed)
- No need to worry about master nodes or the backend infrastructure
- Reduces the initial setup and operational complexity of Kubernetes for Production workloads
- Keeping containerized apps up and running, can be complex – let AKS assist you!

GitOps

GitOps is a process that leverages the Git developer toolset for operations and management of cloud-native applications © rancher.com

GitOps in Kubernetes places the cluster into a desired state

Version Control with GitOps

The only components that are deployed on the cluster is from version control

Ensures what is deployed onto the clusters are correct

GitOps Tooling

GitOps Builds on immuatable infrastructure

Numerous tools are available to help you implement GitOps

In our demo, we will be using Fluxv2

Flux is a tool for keeping Kubernetes clusters in sync with sources of configuration (like Git repositories) and automating updates to configuration when there is new code to deploy.

Flux2

5 main components make up Flux2

- Source Controller
- Kustomize Controller
- Helm Controller
- Notification Controller
- Image Automation Controllers

Fluxv2 – Source Controller

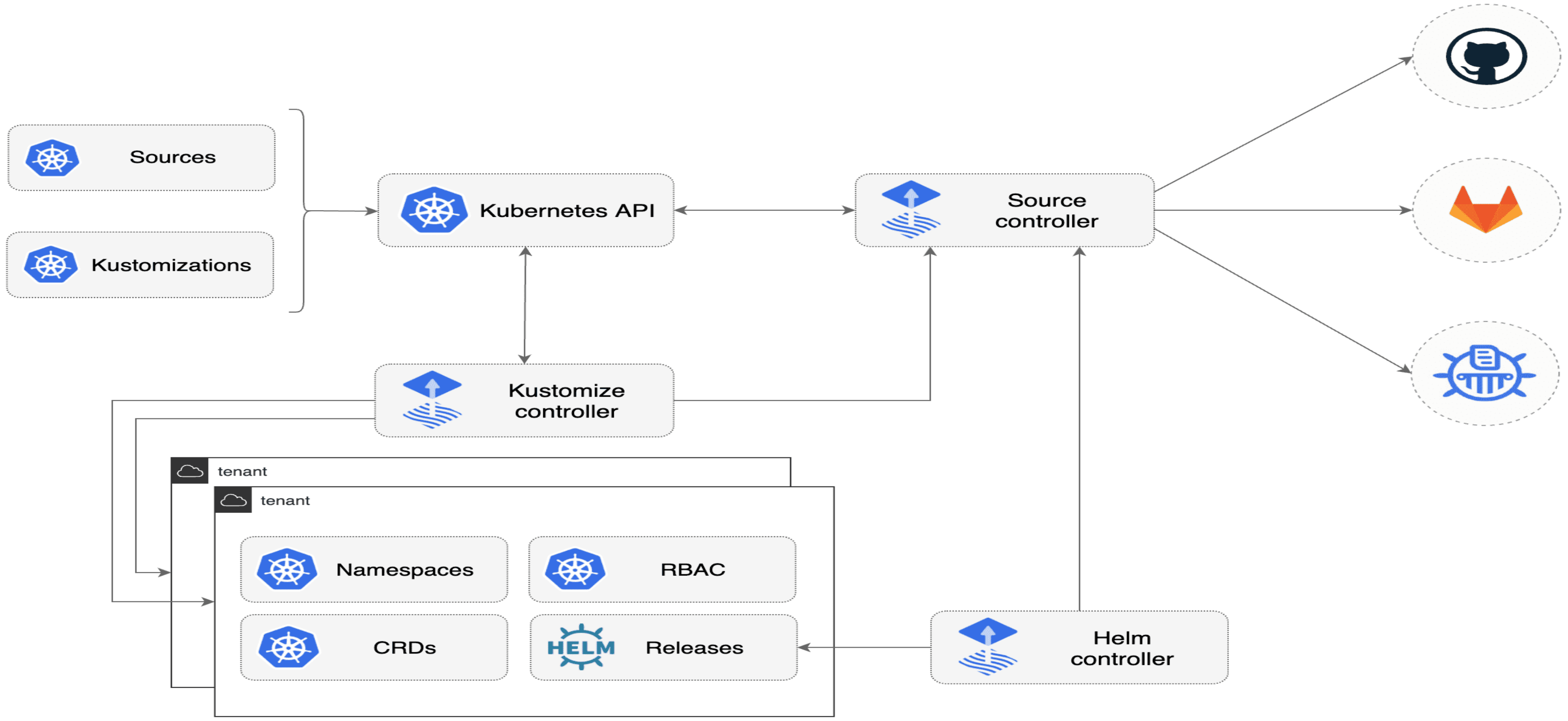


Image Reference: https://github.com/fluxcd/flux2

Fluxv2 – Kustomize Controller

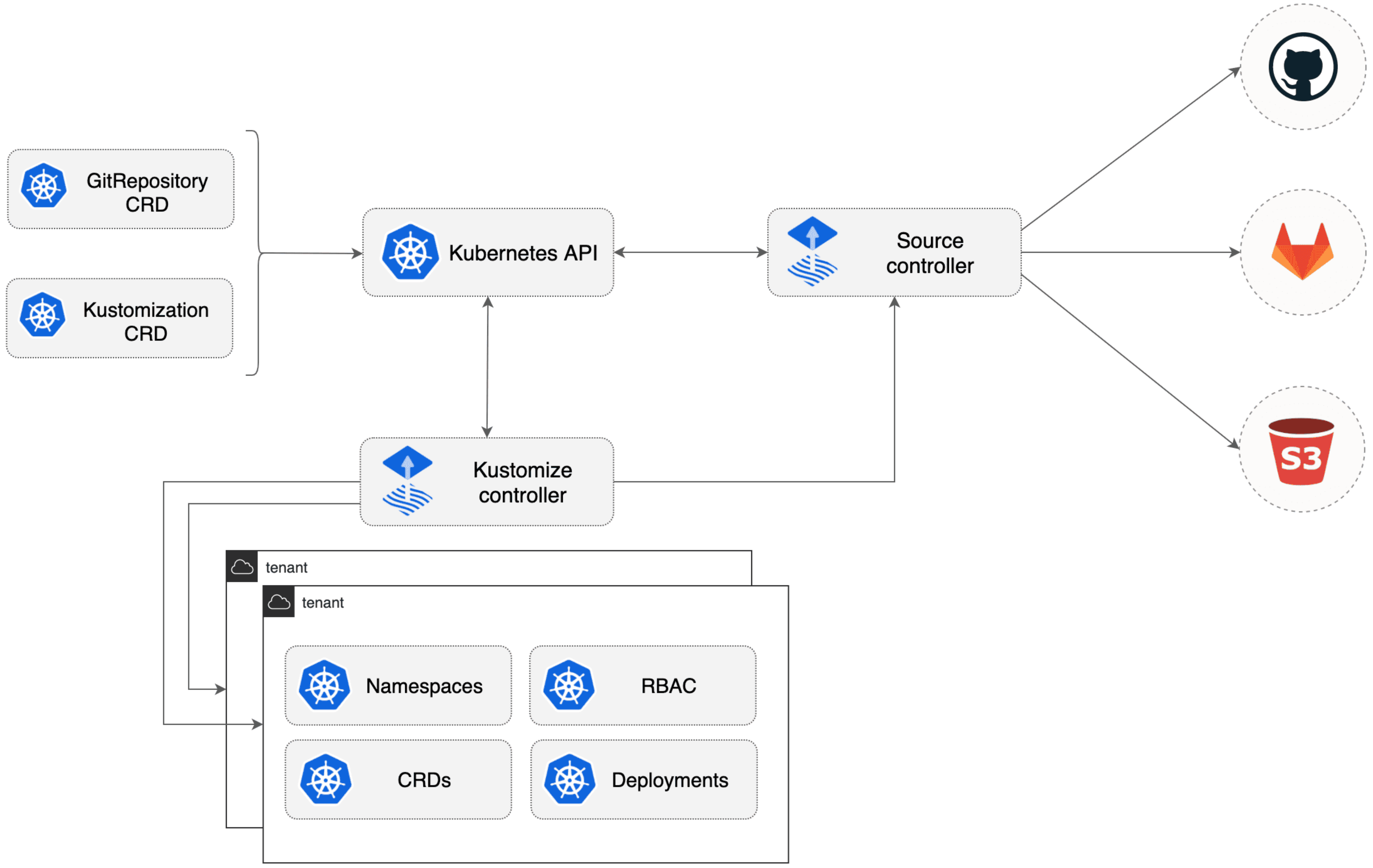


Image Reference: https://github.com/fluxcd/flux2

Fluxv2 – Helm Controller

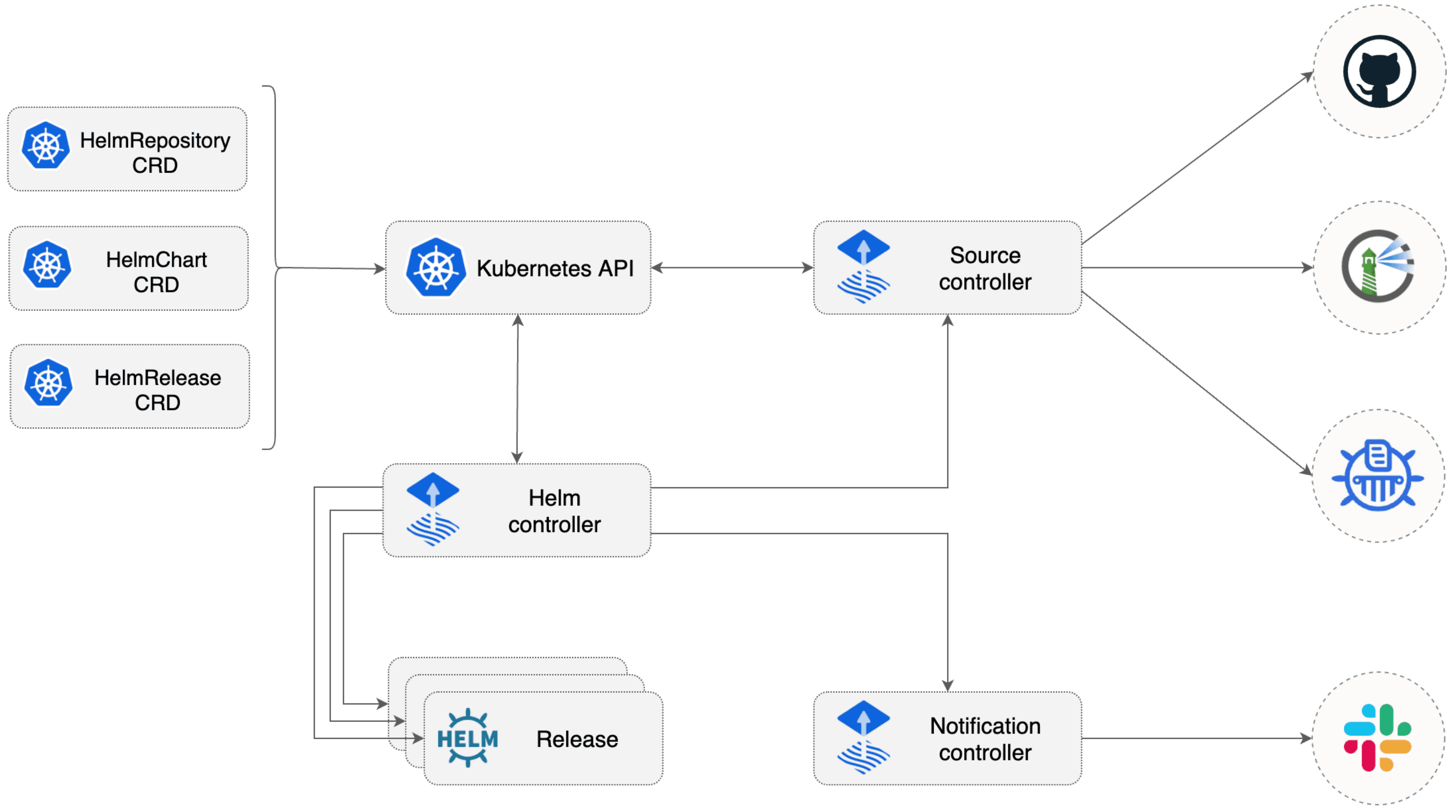


Image Reference: https://github.com/fluxcd/flux2

Fluxv2 – Notification Controller

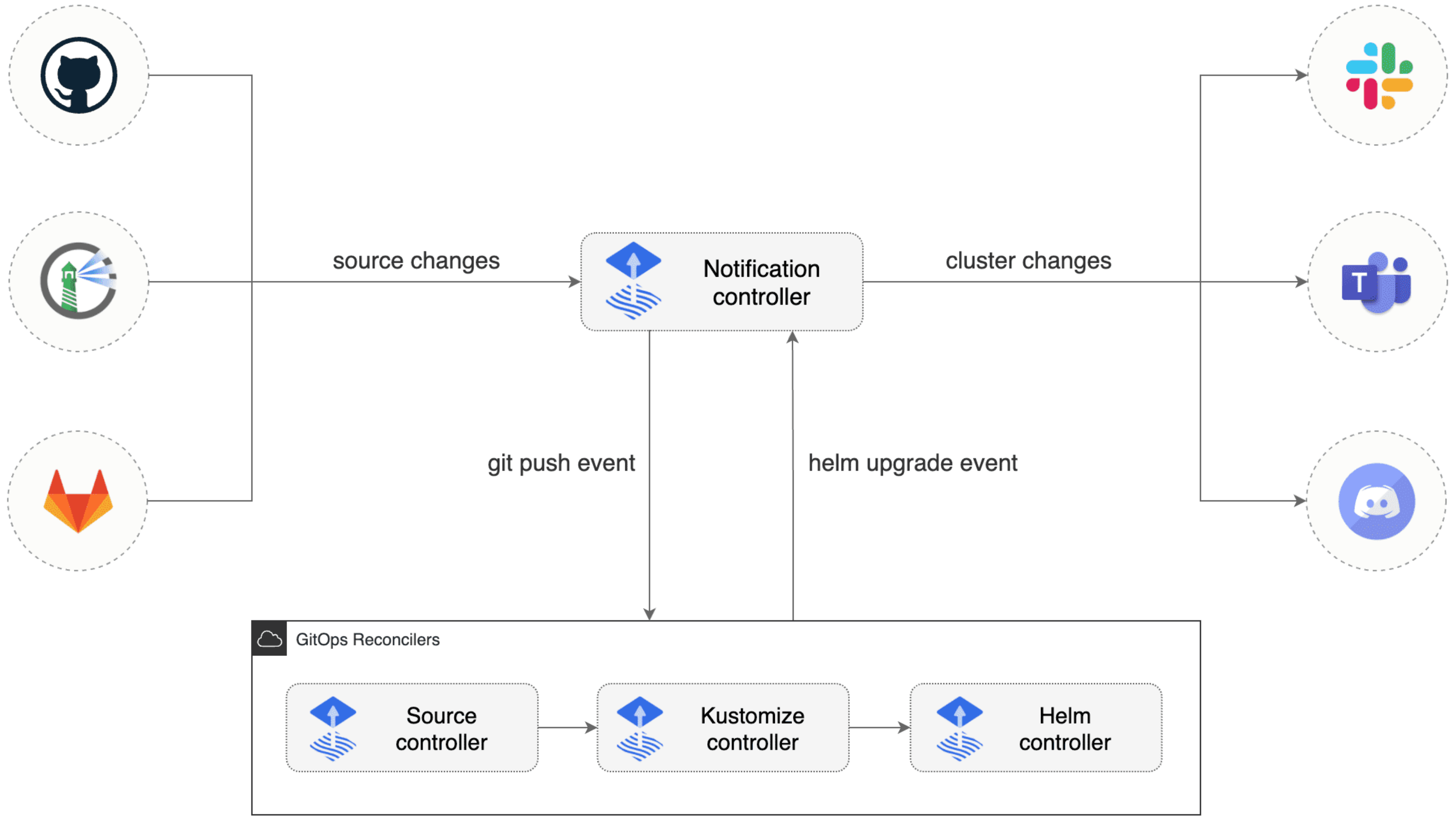


Image Reference: https://github.com/fluxcd/flux2

Fluxv2 – Image Automation

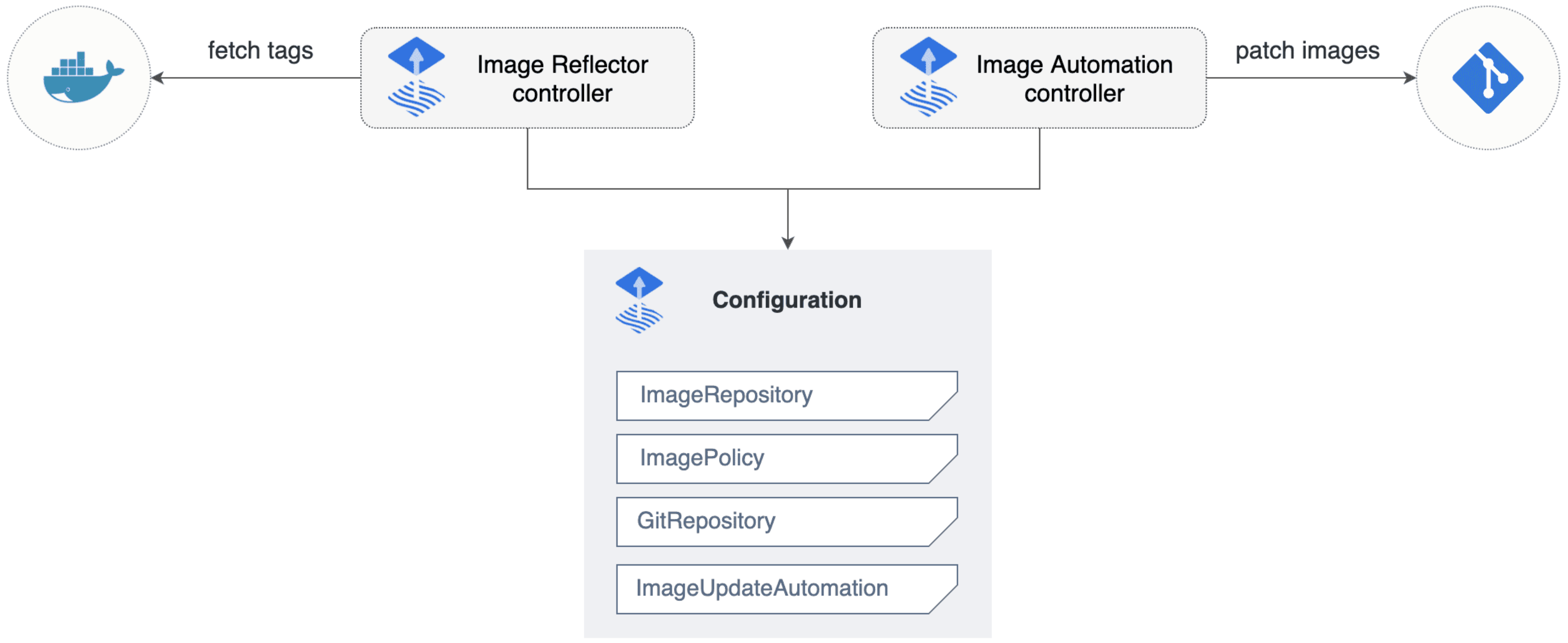


Image Reference: https://github.com/fluxcd/flux2

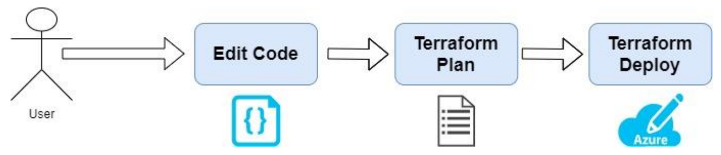
Deploying AKS using Azure DevOps and Terraform

- Deploy initial Infrastructure for AKS
- Terraform deploys the infrastructure into Azure
- Deployed using Azure DevOps Pipelines

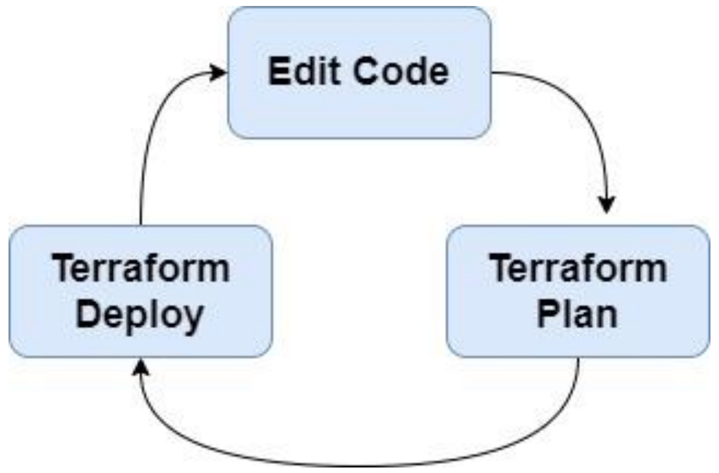
What is Terraform?

- A way to manage Azure
- Easy to read and write
- Declarative
- Driven via the Azure API
- OpenSource Free
- Disposable Environments
- Lowers the potential for human errors while deploying and managing infrastructure.

Terraform Workflow



- **terraform init -** Initialize a Terraform working directory
- **terraform plan -** Generate and show an execution plan
- **terraform apply -** Builds or changes infrastructure
- **terraform output -** Read an output from a state file
- **terraform destroy -** Destroy Terraform-managed infrastructure



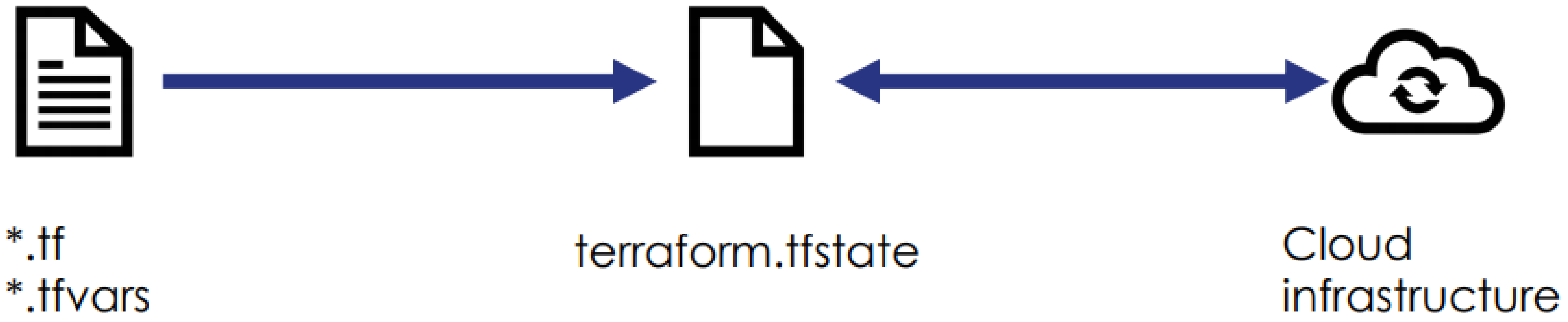
Terraform Teminology

Providers represent a cloud provider or a local provider

Resources can be invoked to create/update infrastructure locally or on the cloud.

State is representation of the infrastructure created/updated by terraform.

Data Sources are “read-only” resources



Terraform State

- Terraform must store state about your managed infrastructure and configuration.

- This state is used by Terraform to map real world resources to your configuration, keep track of metadata, and to improve performance for large infrastructures.

- This state is stored by default in a local file named "terraform.tfstate", but it can also be stored remotely, which works better in a team environment.

Azure Pipelines

Azure Pipelines is a cloud service that you can use to automatically build and test your code project and make it available to other users. It works with just about any language or project type.

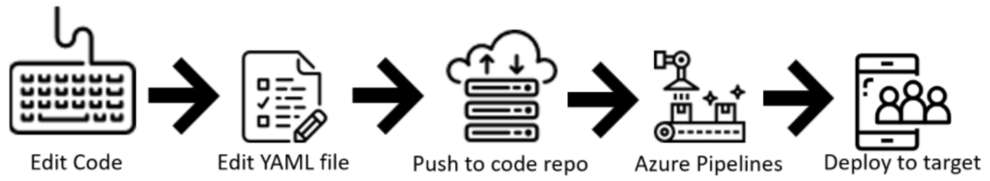
Azure Pipelines combines continuous integration (CI) and continuous delivery (CD) to constantly and consistently test and build your code and ship it to any target.

Within our environment, we will be using Azure Pipelines to deploy our Terraform code

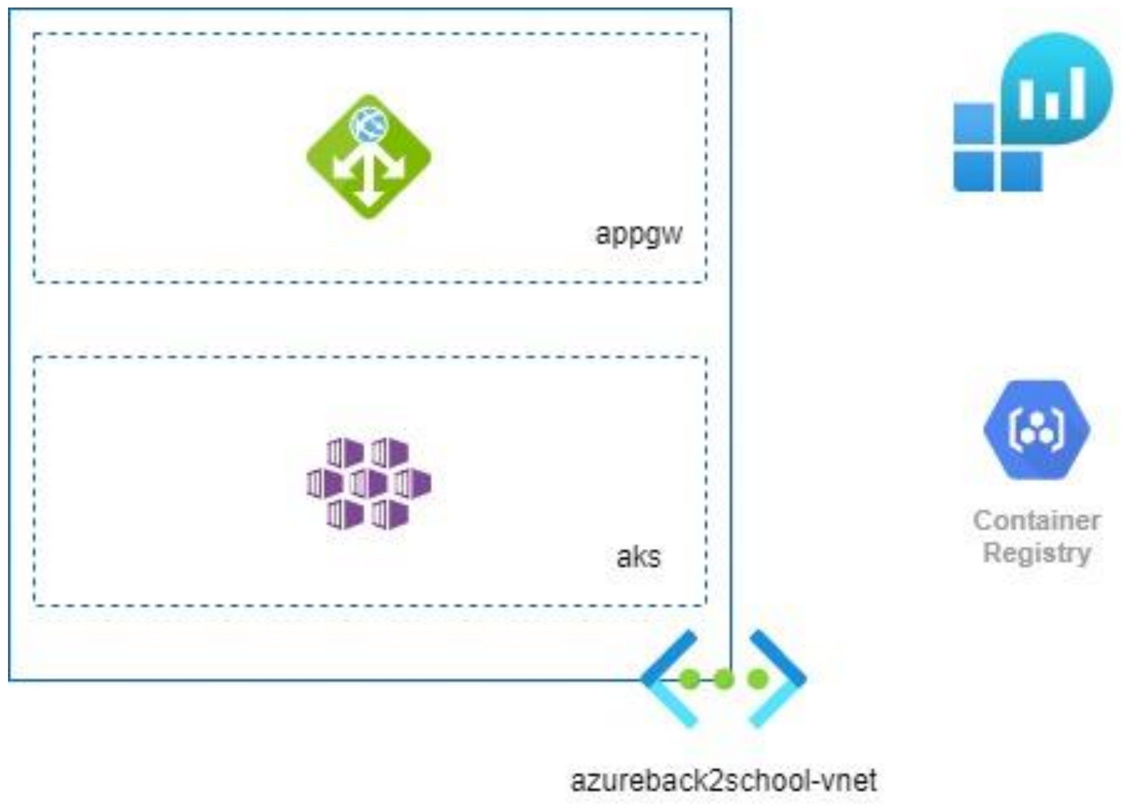
You define pipelines using the YAML syntax or through the Azure DevOps portal – we will be using YAML

Azure Pipelines via Code

- You define your pipeline in a YAML file within your repo, **azure-pipelines.yml** for example
- The YAML pipeline is versioned the same way as your Terraform code.
- It will follow the same branching structure allowing you to have a pull-request process for any changes to any Pipelines that you may make
- Will look at some cool pipeline/repo additions



The GitOps Journey…the beginning.



Demo Time

Demo Time – what did we cover?

- Azure Pipeline Deployment (Using Templates)
- Bootstrapping AKS Cluster
- GitOps
- Updating images via GitOps

Key Takeaways

- Terraform is readable and quite user friendly
- The beginning of CI/CD deployments
- Intro to GitOps
- Test outside of your pipeline
- GitOps & Immutable Infrastructure is awesome!
- The urge to see even more of Azure, DevOps & GitOps! ☺

Recommended blog posts

1. **Deploy Terraform using Azure DevOps** https://thomasthornton.cloud/2020/07/08/deploy-terraform-using-azure-devops/
2. **Creating Templates in Azure DevOps Pipelines** https://thomasthornton.cloud/2021/08/16/creating-templates-in-azure-devops-pipelines/
3. **Flux2** https://github.com/fluxcd/flux2
4. **GitOps for Azure Kubernetes Service** https://docs.microsoft.com/en-us/azure/architecture/example-scenario/gitops-aks/gitops-blueprint-aks
5. **GitHub for Flux2** https://github.com/thomast1906/azure-back-to-school-2021-flux
6. GitHub for demo files /slides https://github.com/thomast1906/azure-back-to-school-2021

Community Resources

- https://azurebacktoschool.github.io/
- Azure Blog's
- YouTube
- User Groups & Virtual Conferences
- GitHub Resources
- CloudFamily.info #cloudfamily
- Community #azurefamily