Corndel Digital.

in association with

Softwire

Workshop: Module 14

Advanced Immutable Infrastructure

Agenda

1000 Welcome and Register

Part 1: Container Orchestration
- Recap (20 mins)
- Exercise (100 mins)

1200 Lunch Break (1 hour)

1300 Part 2: Autoscaling
- Recap (20 mins)
- Exercise (160 mins)

Objectives of this Workshop

- Understand the differences between various common container orchestration tools

- Introduce the basics around Kubernetes, including the architecture, configuring pods & services, and deploying workloads
  - Today we’ll be working with Kubernetes clusters managed by the Azure Kubernetes Service (AKS)

- Setup scaling policies for your cluster based on various performance metrics

Part 1

Container Orchestration

Recap

What is Container Orchestration?

Container orchestration is the practice of automating as much of the management of containers as we can.

It might involve:

- Provisioning a set of machines (bare-metal or virtual) on which to run the containers
- Automatically rolling out updates to containers
- Scaling the system (up & down) to meet demand
- Managing networking between the containers themselves
- Managing networking between the containers and the outside world

Recap

Kubernetes

Kubernetes is an example of a container orchestration tool.

- It provisions containers across a **Cluster** of **Nodes**
  - Nodes are worker machines available for Kubernetes to use
  - They can be physical or virtual machines
- Kubernetes will run **Pods** on the nodes
  - A Pod can contain multiple tightly coupled containers, but is often a wrapper for a single container
- A **Service** represents a logical group of Pods
  - This allows our systems to talk to a single place, which can then worry about how to interface with our dynamically changing Pods

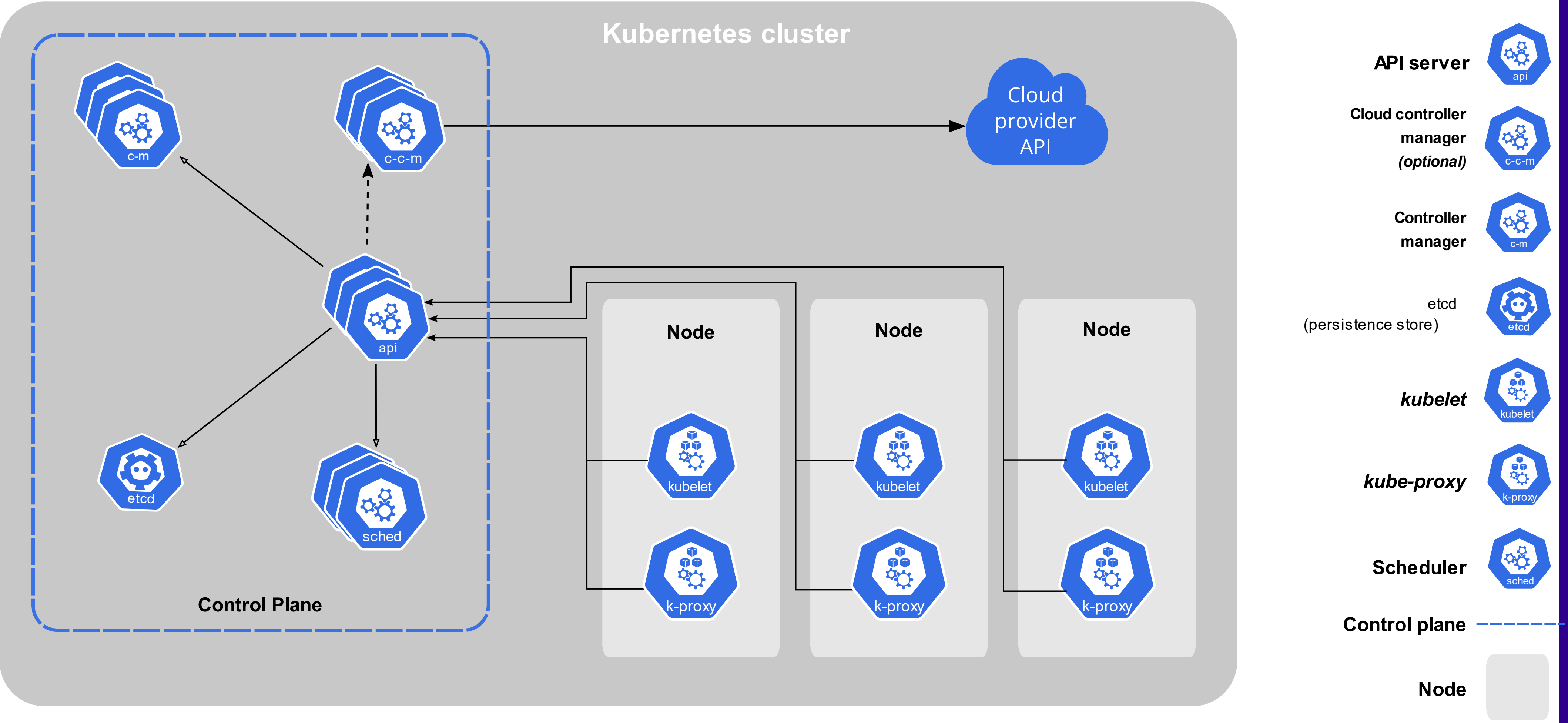

Image credit: Kubernetes docs https://kubernetes.io/docs/concepts/overview/components/

Recap

Helm

Helm is a package manager for Kubernetes - packages are called **Charts**

Helm:

- Assists with creating new Charts
- Allows installing existing Charts from a **Chart Repository**
- Manages the release cycle of charts you’ve deployed
- Reduces duplication through templating to make it easy to share your configuration across environments

Exercise

Clone this repository and follow the instructions:

https://github.com/CorndelWithSoftwire/DevOps-Course-Workshop-Module-14-Learners

Part 2

Autoscaling

Recap

Autoscaling

We’ve previously seen examples of autoscaling; Azure Functions were able to scale to handle varying load.

Kubernetes is able to manage autoscaling in several dimensions:

- Within our cluster by scaling pods horizontally
  - We will often provide guidance on the minimum & maximum number of pods allowed in a deployment
  - We may provide guidance on what usage we want Kubernetes to target, e.g. 80% cpu
- Scaling pods vertically, according to the requests and limits that we assign
- In some platforms we can allow Kubernetes to horizontally scale the number of nodes that it has available; claiming an extra machine when needed and releasing it when not

Exercise

Continue with this morning’s instructions

https://github.com/CorndelWithSoftwire/DevOps-Course-Workshop-Module-14-Learners

Thank You!

Please click here to share feedback, or scan the QR code below