Azure Kubernetes Services (AKS) - The Big Picture

WHAT IS AZURE CONTAINER SERVICE (AKS)?

Manoj Ravikumar Nair
CLOUD SOLUTIONS ARCHITECT

@powershellpro http://manojnair.in

Containers – Computing Redefined

Containers have become the ‘preferred’ choice for compute

Kubernetes, the most popular choice for ‘Container Orchestration’

Containers? Orchestration? Kubernetes?

Module Overview

The Case for Kubernetes

Containers 101

What is Kubernetes?

Kubernetes Objects

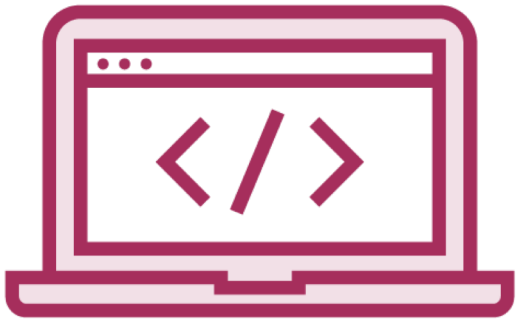
What is Azure Kubernetes Service (AKS)?

Going Beyond Managed Kubernetes

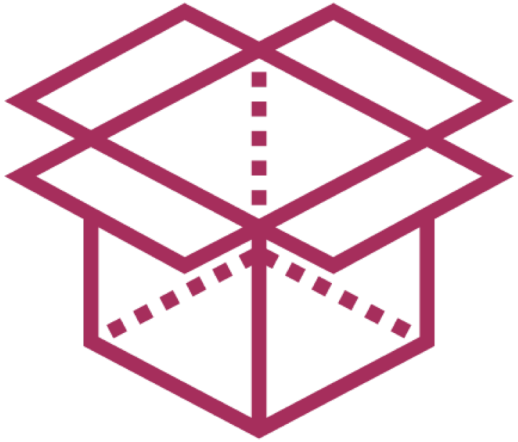
Module Summary

The Case for Kubernetes

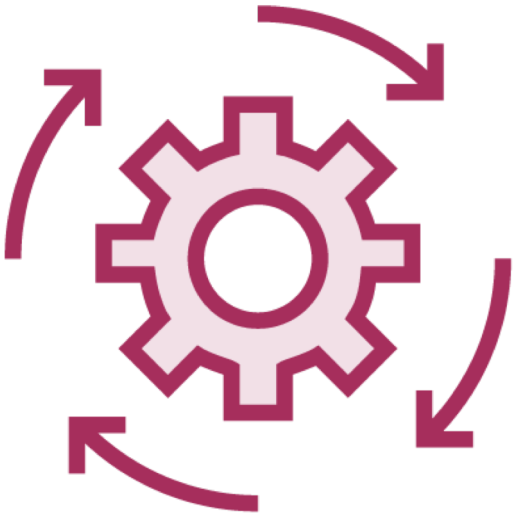
Traditional Software Development



Build software as a single code base



Package the code into a deployable entity



Deploy and monitor health

Migrate to healthy servers if hardware fails

Monoliths – the Good and the Bad

Fewer moving parts enables easy deployment

Longer Release cycles

Update to one functionality requires redeployment of the entire codebase

Microservices

Monoliths are broken down into smaller, independently running components called Microservices

Microservices are decoupled from each other, and therefore they can be developed, deployed, updated and scaled individually

Each microservice exposes its functionality or services via an interface, typically a REST API

Microservices – the Good Parts

An application is sum of its components

Better fault isolation

Choice of Language

Components can be spread across multiple servers

Containers facilitate Microservices Architecture

Microservices – the Bad Parts

Many components, many moving parts

Difficult to manage inter-communication

Difficult to achieve high resource utilization

Manual management can be difficult

We need automation, which includes automatic scheduling automatic configuration, supervision, and failure-handling

We need an ‘Orchestrator’

Kubernetes

Kubernetes...

Enables developers to deploy their applications themselves and as often as they want

Enables the ops team by automatically monitoring and rescheduling those apps in the event of a hardware failure

The focus from supervising individual apps to mostly supervising and managing Kubernetes and the rest of the infrastructure

Abstracts away the hardware infrastructure and exposes your whole datacenter as a single gigantic computational resource

Azure Kubernetes Service (AKS)

```
Kubernetes + Microsoft Azure = Pure Awesomeness
```

What is Kubernetes?

Kubernetes

A software system that allows you to easily deploy and manage containerized applications on top of it

Exposes the underlying infrastructure as a single computational resource

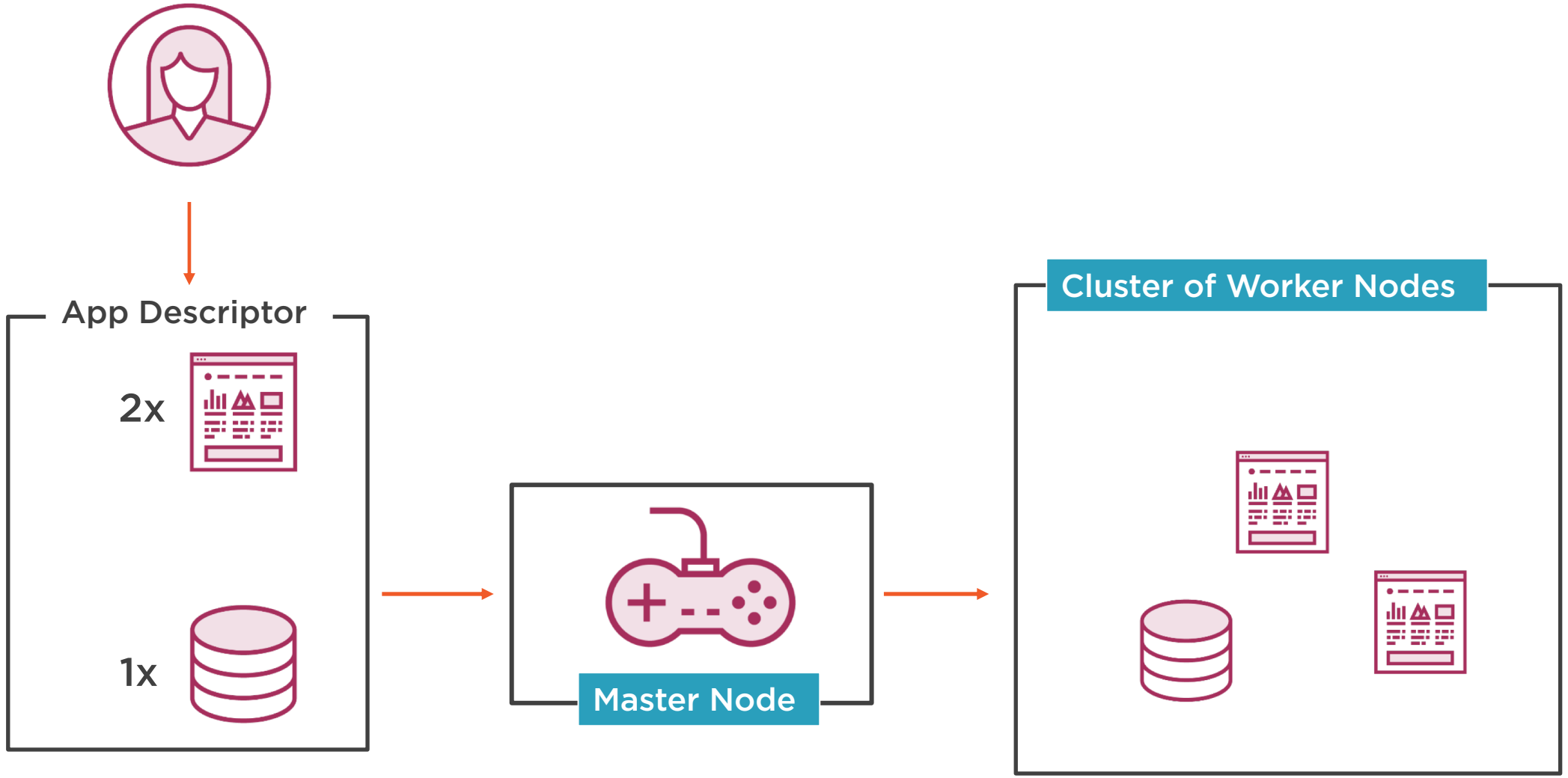
Consistent deployment experience regardless of the size of the cluster

Kubernetes Cluster Node Types

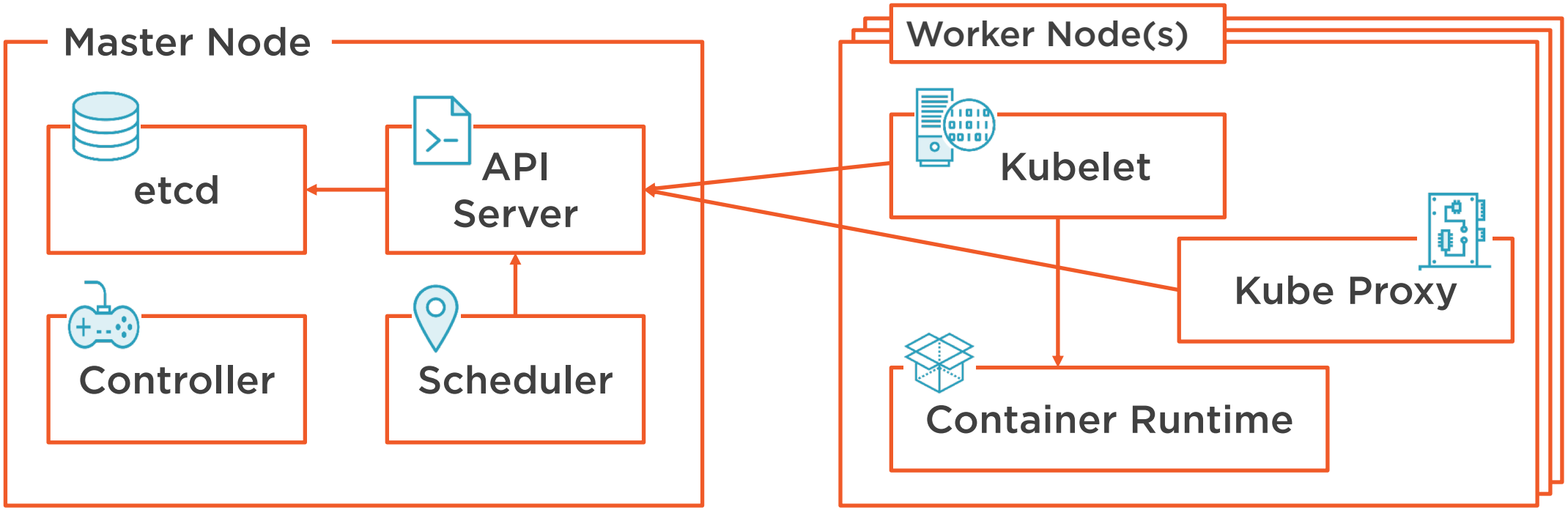
Master Node(s)

Worker Nodes(s)

Kubernetes System



Kubernetes Cluster Architecture



Running an Application in Kubernetes

Step 1

Package your application in to one or more containers

Step 2

Push those images to an Image Registry

Step 3

Post App Descriptor to the Kubernetes API Server

Step 4

Scheduler schedules containers on available Worker Nodes

Step 5

Kubelet instructs Nodes to download Container Images

Step 6

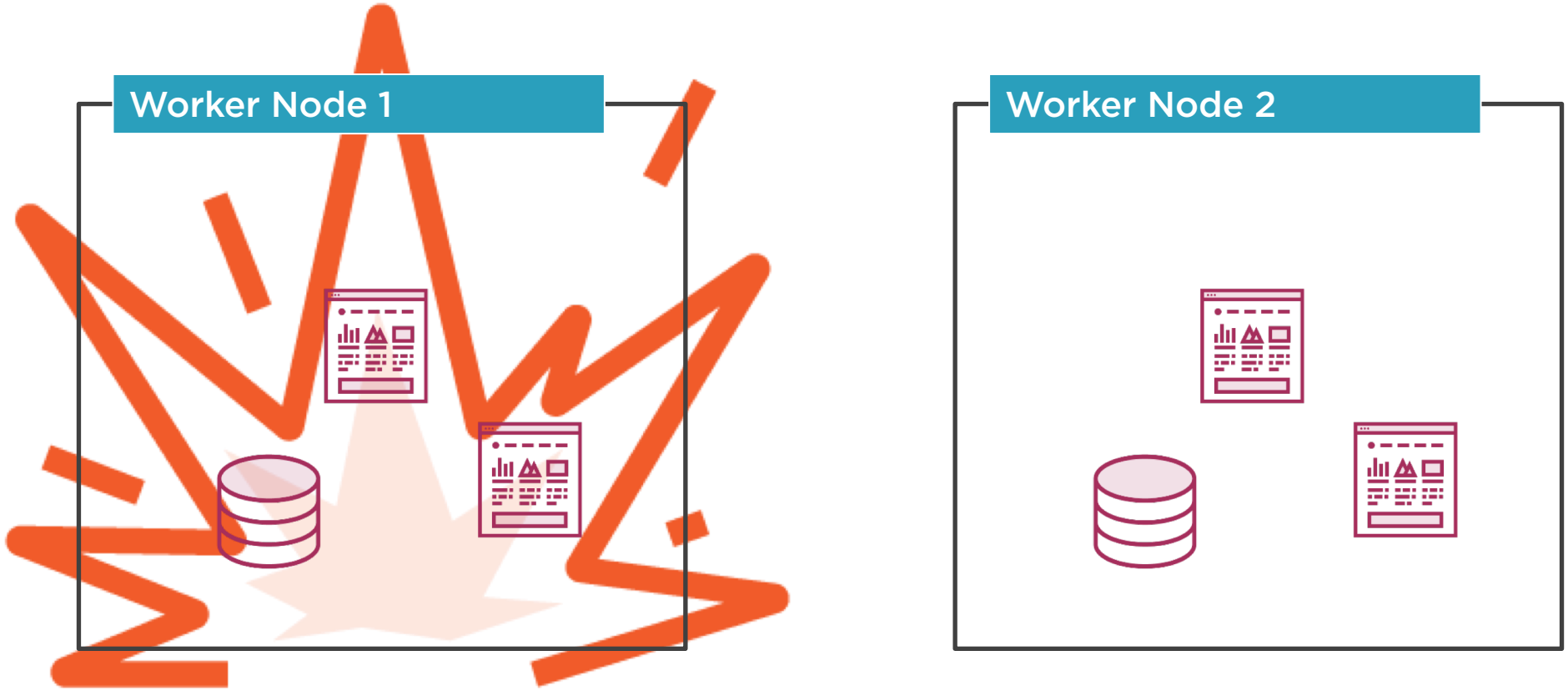
Kubelet instructs Nodes to run the Containers

Simplifies Application Deployment

Better Hardware Utilization

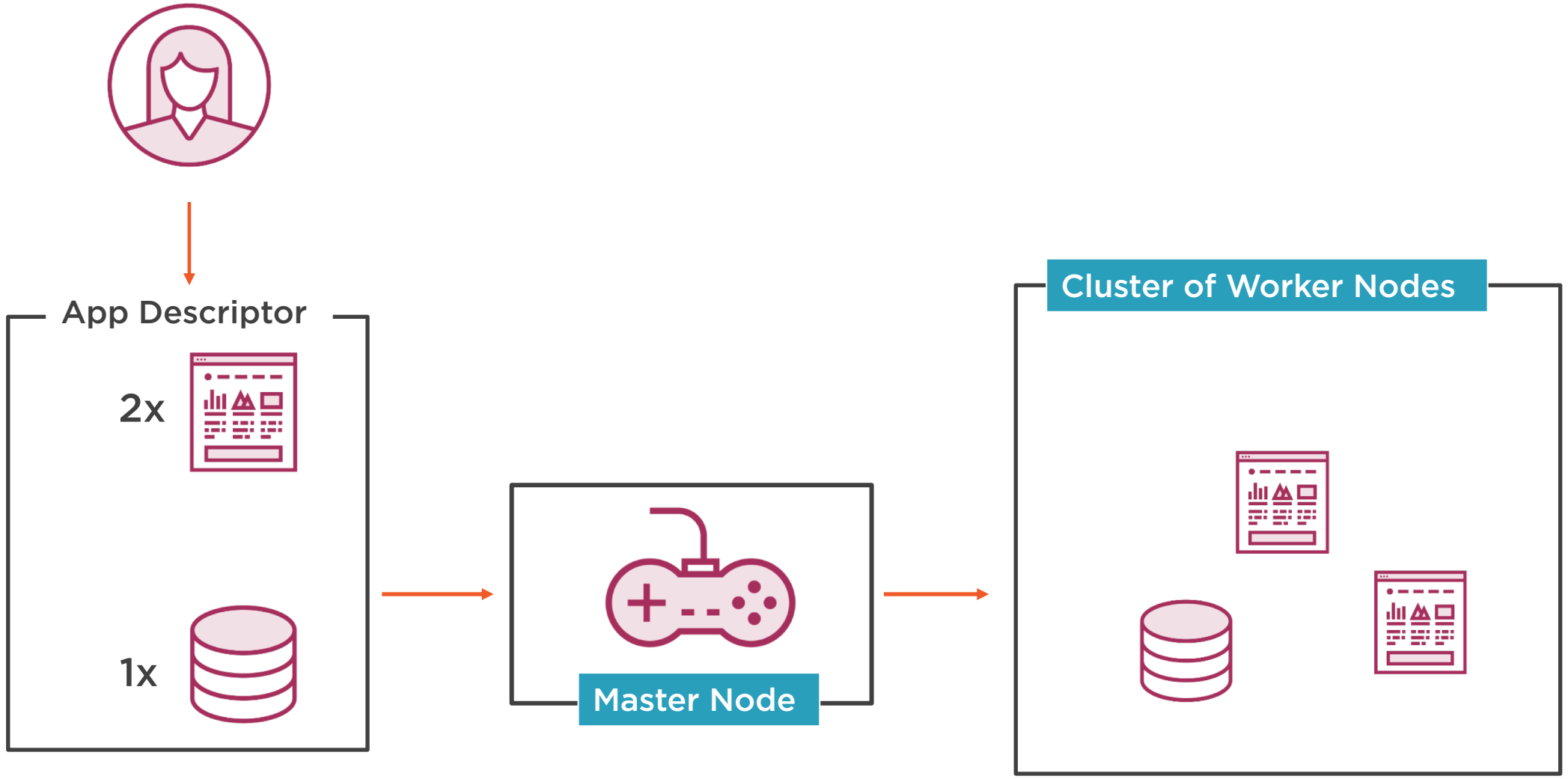
Health Monitoring with Self-Healing

Node Failure Triggers Container Rescheduling

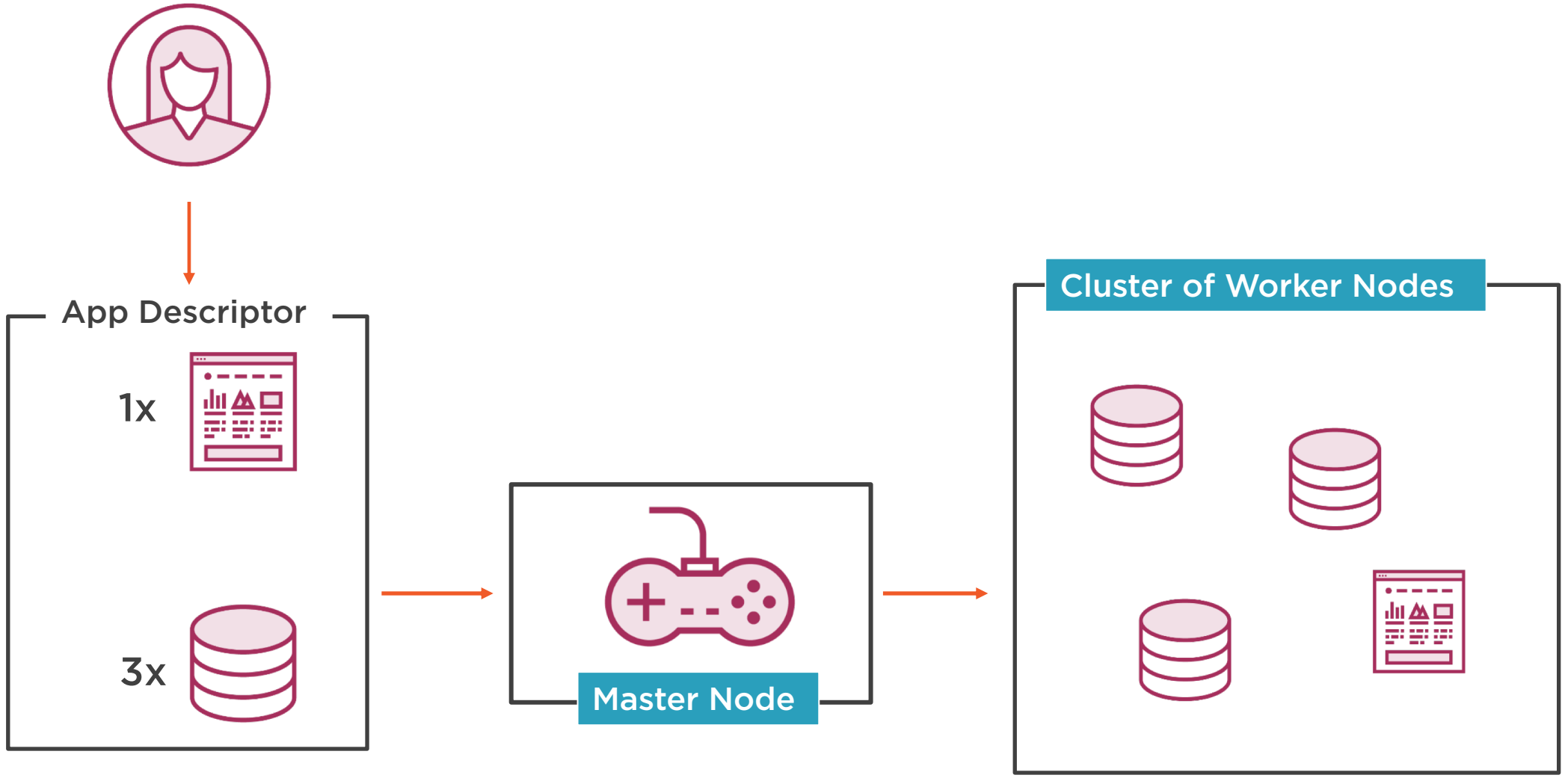


Automatic Scaling

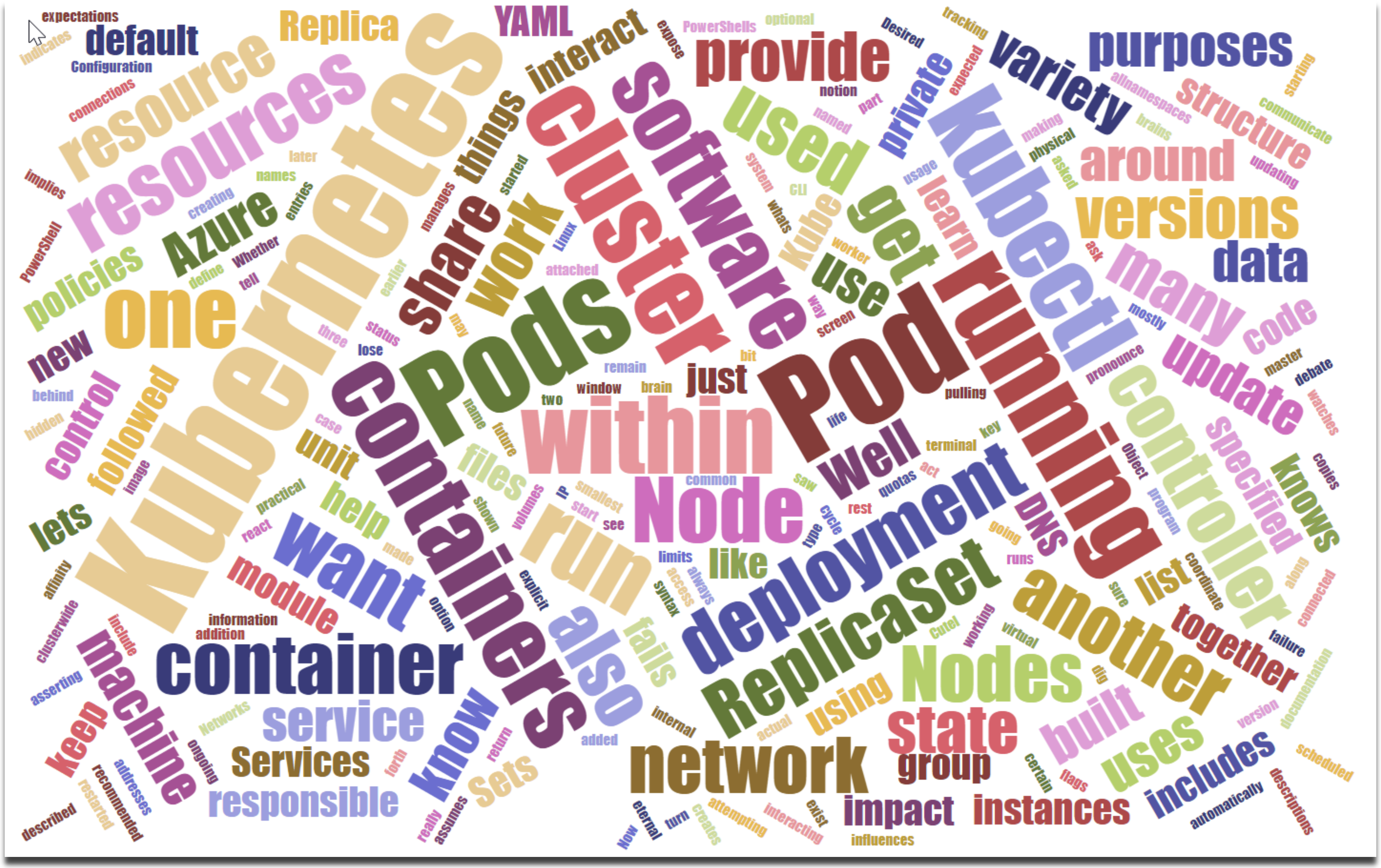
Automatic Scaling



Automatic Scaling



Excited!!!


Kubernetes
Pod
Pods
containers
within
cluster
running
kubectl
Node
deployment
ReplicaSet
container
network
software
resources
resource
controller
Nodes
another
want
run
one
state
used
provide
purposes
variety
versions
around
structure
data
many
update
share
work
Azure
policies
default
Replica
YAML
interact
things
get
use
learn
private
Kube
service
Services
responsible
machine
module
know
also
help
unit
files
fails
Sets
followed
control
new
lets
keep
just
Well
like
using
impact
instances
group
built
uses
includes
together
list
knows
specified
code
expectations
Configuration
connections
later
names
creating
entries
implies
PowerShell
Whether
define
tell
earlier
three
status
lose
behind
remain
window
brain
two
hidden
case
image
practical
react
made
affinity
clusterwide
include
information
addition
asserting
Networks
ongoing
addresses
recommended
restarted
described
return
forth
really
assumes
internal
added
actual
always
access
syntax
explicit
option
limits
see
start
smallest
common
IP
volumes
shown
future
name
may
attached
Linux
started
manages
expose
PowerShells
optional
notion
named
part
system
CLI
whats
worker
way
screen
bit
pulling
life
terminal
key
act
quotas
rest
cycle
type
saw
DNS
going
runs
Cutel
working
virtual
dig
certain
flags
attempting
interacting
exist
influences
creates
turn
eternal
Now
automatically
descriptions
scheduled
version
documentation
failure
connected
along
sure
coordinate
program
Object
copies
watches
debate
master
pronounce
mostly
ask
usage
asked
physical
making
brains
allnamespaces
updating
communicate
starting
Desired
tracking
expected

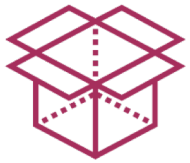
Kubernetes Objects

Kubectl

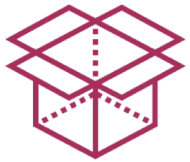
```
kubectl <operation> <object> <resource name> <optional flags>
```

```
kubectl get nodes
```

Pods



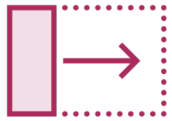
Smallest unit that Kubernetes manages



A Pod is made up of one or more containers and information associated with those containers

Querying a pod returns a data structure that contains information about containers and its metadata

Characteristics of a Pod



All the containers for a Pod will be run on the same Node

Any container running within a Pod will share the Node's network with any other containers in the same Pod

Containers within a Pod can share files through volumes, attached to the containers

A Pod has an explicit life cycle, and will always remain on the Node in which it was started

Namespaces

Pods are collected into namespaces, which are used to group Pods

Namespaces can be used to provide quotas and limits around resource usage and have an impact on DNS names that Kubernetes creates internal to the cluster

If no namespace is specified when interacting with Kubernetes through kubectl, the command assumes you are working with the default namespace, named default

Nodes

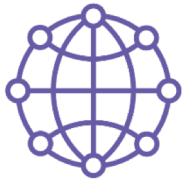
Node is a machine that is added to the Kubernetes Cluster

The master node is the brain of Kubernetes while the worker nodes do the actual work of pulling container images and running pods

Networks

All the containers in a Pod share the Node's network

All Nodes in a Kubernetes cluster are expected to be connected to each other and share a private cluster-wide network



Kubernetes runs containers within a Pod within this isolated network

Controllers

Kubernetes encourages desired state deployment

You assert you want one or more resources to be in a certain state and with specific versions

Controllers are where the brains exist for tracking those resources and attempting to run your software as you described

ReplicaSet

A ReplicaSet is associated with a Pod and indicates how many instances of that Pod should be running within the cluster

A ReplicaSet also implies a controller that watches the ongoing state and knows how many instances of your Pods to keep running

A ReplicaSet is commonly wrapped in turn by a deployment

Deployments

Recommended way to run code on Kubernetes

The deployment controller wraps around and extends the ReplicaSet controller

Includes metadata settings to know how many Pods to keep running

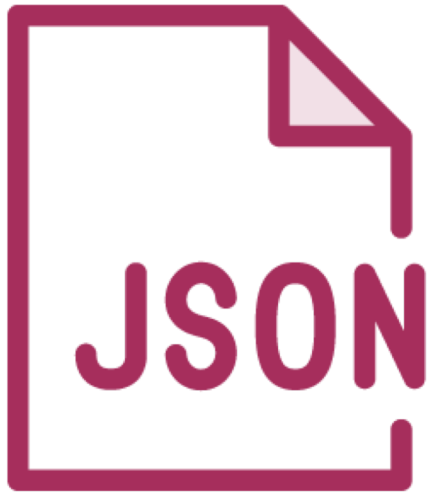
Services

Kubernetes resource used to provide an abstraction through to your Pods agnostic of the specific instances that are running

Can contain a Policy

Emulates a ‘software load balancer’ within Kubernetes

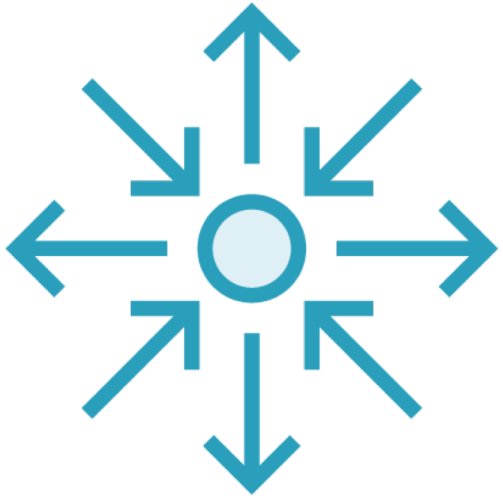
Representing Kubernetes Resources



What Is AKS?

Self-hosting Kubernetes Cluster

Manually install the master and worker nodes



Need to consider high availability of the Master, adding additional worker nodes, patching, upgrades etc.

Microsoft Azure

Azure Kubernetes Service

Simplifies deployment, management, and operations of Kubernetes

Azure Kubernetes Service or AKS is currently in public preview

Makes it quick and easy to deploy and manage containerized applications without container orchestration expertise

Eliminates the burden of ongoing operations and maintenance by provisioning, upgrading, and scaling resources on demand

Azure Kubernetes Service

Master Node(s) managed by Microsoft

Access to enterprise-grade features of Microsoft Azure

Reduces the complexity and operational overhead of managing a Kubernetes cluster by offloading much of that responsibility to Azure

Handles critical tasks like health monitoring and maintenance for you

Benefits of Azure Kubernetes Service

Automated Kubernetes version upgrades and patching

Easy cluster scaling

Self-healing hosted control plane (masters)

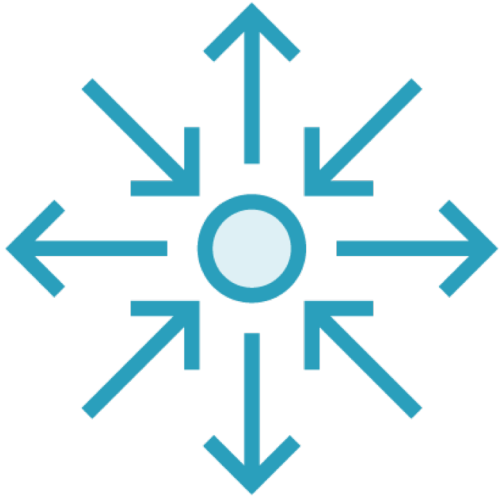
Cost savings

Why should you use AKS when there are other vendors out there offering a managed Kubernetes platform ?

Beyond Managed Kubernetes?

Self-Hosting Kubernetes Cluster

Manually install the master and worker nodes



Need to consider high availability of the Master, adding additional worker nodes, patching, upgrades etc.

Microsoft Azure

Azure Container Instances (ACI)

Deploy your containers without worrying about underlying infrastructure

Easily start deploying containers for targeted use cases

‘Dockerize’ your application and execute it in one click

Service Fabric

Service Fabric is the foundational technology powering core Azure Infrastructure as well as other Microsoft services

Designed to deliver highly available and durable services at cloud-scale

Microsoft's container orchestrator deploying microservices across a cluster of machines

Service Fabric is a perfect choice as a container orchestrator

Service Fabric Deployment Options

Runs Everywhere, Azure, On-premises, other Cloud Vendors

Runs on both, Windows and Linux Operating Systems

Builds stateful services, either with the built-in programming models or with containerized stateful services

Azure Web App for Containers

Azure Web Apps is the industry leader for PaaS Platform

Web App for Containers extends Azure Web Apps support to include Containers

Built-in Autoscaling and Load Balancing and CI/CD with GitHub, Docker and Azure Container Registry

Azure Batch Service

Cloud-scale job scheduling and compute management

Now includes support to set up a batch pool to run tasks in Docker containers

Provides an easy way to run batch tasks without having to manage application packages and dependencies

Both Windows and Linux containers are supported

Module Summary

Microservices implemented via Containers warrant an ‘Orchestrator’

Containers offer repeatable and consistent deployment environment

Kubernetes offers declarative configuration and automation

Azure Kubernetes offers managed Kubernetes platform that can be scaled on demand

Microsoft Azure offers a rich suite of services for building your applications