Microsoft

Kubernetes – The Big Picture

Pets vs Cattle

Keep Things in Perspective

- In the ***old way*** of doing things, we treat our servers like pets, for example Bob the mail server. If Bob goes down, it's all hands-on deck. The CEO can't get his email and it's the end of the world!
- In the ***new way***, servers are numbered, like cattle in a herd. For example, www001 to www100. When one server goes down, it's taken out to pasture (made into hamburger) and replaced on the line.
- In the age of Infrastructure-as-Code, don't think of your workloads as pets.
- Kubernetes treats your Pods like cattle. If one stops working, it doesn't try to fix it. It simply replaces it with a new instance (new IP, fresh memory, etc.) and terminates the old one.
- It's not personal; it's just cattle.

Know What are Pets and What are Cattle

Cattle

- Pods
- Replica Sets
- Deployments
- …everything else in K8s
- Entire Kubernetes clusters
- Virtual Infrastructure (aka most Azure Resources)

Pets

- Databases
- Data Disks
- File Shares
- Source Code
- DevOps Pipelines
- Infrastructure Scripts (ARM, Bicep, TerraForm, etc.)

How to Treat Your Kubernetes Clusters Like Cattle, Not Pets

Kubernetes is Designed For Stateless Workloads

The instance you store data in your cluster, it becomes your pet!
You then become responsible for its care and maintenance.

Consider outsourcing your data storage to a managed service instead?

Done for you:

- Automatic backup
- Replication
- Geo-Redundancy
- Auto-Scaling
- Fully-Managed
- SLAs

Azure SQL

Radis Cache

Azure Disk

Azure MySQL

Cosmos DB

PostgreSQL

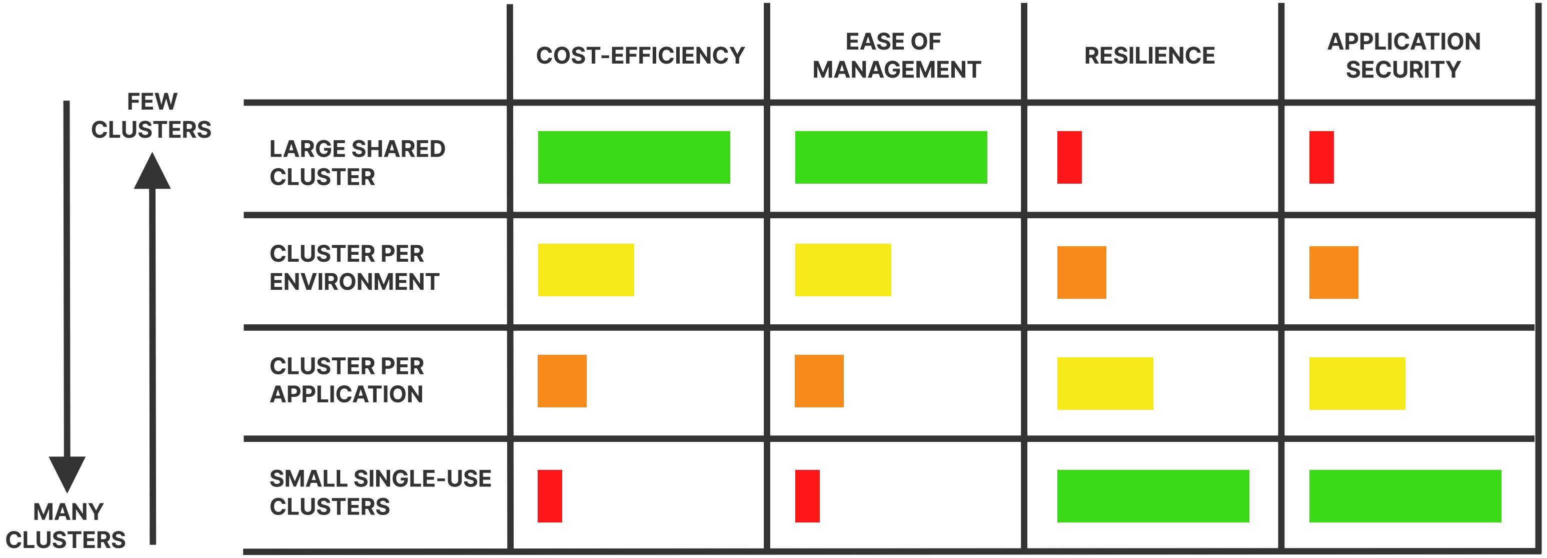
Azure File Share

Azure Queues

Service Bus

Kubernetes Architecture Best Practices

- Don’t put all your workloads in the same cluster so it becomes a pet!
- Separate workloads into logical boundaries: One cluster for Dev/Test/QA, one for Staging/Production.
- Segment applications into multiple clusters (not just namespaces).
- Architecting Kubernetes clusters — how many should you have?



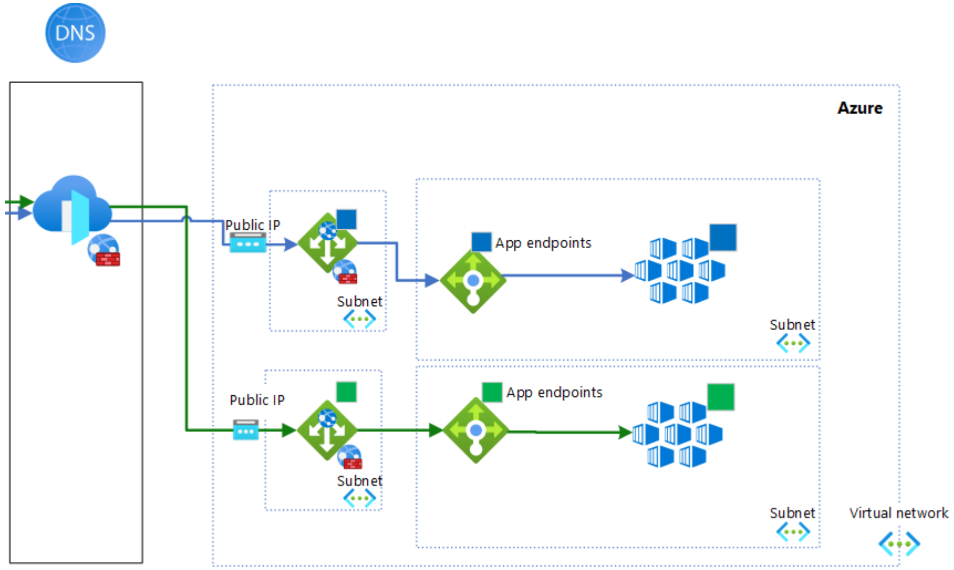
	COST-EFFICIENCY	EASE OF MANAGEMENT	RESILIENCE	APPLICATION SECURITY
LARGE SHARED CLUSTER	🟩	🟩	🟥	🟥
CLUSTER PER ENVIRONMENT	🟨	🟨	🟧	🟧
CLUSTER PER APPLICATION	🟧	🟧	🟨	🟨
SMALL SINGLE-USE CLUSTERS	🟥	🟥	🟩	🟩

Always Host Production Workloads Separately

- **ALWAYS** put production workloads on a separate cluster!
  - Keep security separate
  - Upgrade/scale independently from Dev/Test/QA
  - Ensure production never affected by bad Dev/Test deployments.
  - Can keep production clusters in different subscriptions
  - Easily segment operating expenses.

Upgrading Kubernetes Version

- To upgrade the version of Kubernetes after it’s installed:
  - Upgrade the Control Plane
  - Upgrade the Worker Nodes one at a time
  - Kubernetes performs node upgrades by:
    - Creating a new node
    - Moving existing workloads onto that node
    - Deleting the previous node
- Another option is to NOT upgrade a cluster at all.
  - Create a new cluster
  - Deploy workloads (using DevOps).
  - Verify the workloads then update DNS to the new cluster.
  - When everything is verified again, delete old cluster
  - This process is doing a Blue-Green deployment of AKS at the cluster level.



Microsoft

Thank you