Pets or cattle?

When we’re talking about servers and in particular virtual machines there are two types to consider, each of which have their own uses and when the wrong type is used it can cause problems further down the road, it is important to note that pets and cattle can be physical, virtual, cloud based or any combination of them, it can also be described as scale up vs scale out.

Pets

Pets are virtual machines that you care about, they are the ones who need to be looked after and “groomed” to make sure they are kept in tip top condition, generally these are the critical services that will underpin the environment such as active directory, sql servers, load balancers, mainframe systems etc.. these servers are typically hand built and managed given their underlying criticality.

Cattle

Cattle as the name implies are servers that if one goes “lame” you just replace it with another from the herd, they are not unique or special in any way, these servers are typically built using automation and if one fails it can be self fixed by for example restarting services or the service can route around it without issue, typically these are web servers that can be quickly deployed with software installed on them and then dropped in and out of service, physical servers can also fall into this category for example if a virtual host fails then services move to a different host until the hardware is replaced.

What’s the point?

If you think that this sounds very much like the way Dev-Op’s works then you’re right, these concepts have been created as away to explain the mind shift from traditional on premises to cloud and the shift from waterfall development to agile.
The ultimate goal is to assume that everything can and will fail and to build it in such a way that it doesn’t matter if they do. There are a few key concepts to remember:

- No local storage – if nothing is local then there is nothing stopping you removing the server, Azure blob service and Amazon S3 are perfect for this.
- No local sessions - customer sessions should live in a database or cache never locally.
- Redundancy - if a server goes down, another should be there to take over immediately. Think Autoscale, Azure Load Balancer or Elastic Load Balancer.
- Expect failure – expected absolutely everything to fail, services, components, racks, power, servers, EVERYTHING.
- Flexibility – Your apps should be able to route around any issues that occur by for example automatically dropping “sick” servers out of a loadbalanced farm.

So why should we build a herd?

This would free up countless hours of support time which could be better used streamlining operations and looking at further ways to automate processes, it also allows a much greater degree of flexibility when deploying services because you can ramp up and down services on demand and save on the cost of machines running when nobody is using your service as well