@asksrihari

Azure Kubernetes Service

The BIG Picture

Notes based on
Pluralsight course

Srihari Sridharan

@asksrihari

Azure Kubernetes Service
The Big Picture

11/02/2022

You will learn

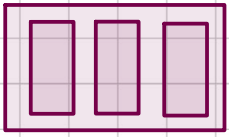
- What is AKS?
- AKS in Action

Other courses

Azure Container Instances
Designing a compute strategy for Microsoft Azure

1. What is AKS?

You should know about "Containers"



@ The case for kubernetes

Traditional software Development

Monolith

Build Package Deploy Migrate (if hardware fails)

Present Microservices

Independent, Distributed, etc.

Horizontal vs. Vertical scaling!

Scale specific features of the service!

Explore the pros and cons of microservices!

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

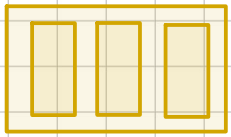
Srihari Sridharan

@asksrihari

Microservices → needs → Containers → need → Orchestration → K8S.

AKS = K8S + Azure

⑥ Containers 101



Containers ← lightweight design of a process
isolation of an operating system.

Containers vs. Virtualization

↳ Search and learn

Containers use namespaces offer isolation of system resources

File Systems
Process Ids
User Ids
Network Interfaces
} Single Namespace

The process will only see resources that are inside the same namespace.

Process doesn't belong to one namespace but one namespace of each kind.

Namespace kinds

- Mount (mnt)
- Process ID (pid)
- Network (net)
- Inter process communication (ipc)
- UTS
- User ID (user)

Srihari Sridharan

eg) UTS namespace determines what hostname and domain name the process running inside that namespace sees.

What network namespace a process belongs to determines which network interfaces the application running inside the process sees.

Each container uses its own network namespace and therefore each container sees its own set of network interfaces.

---

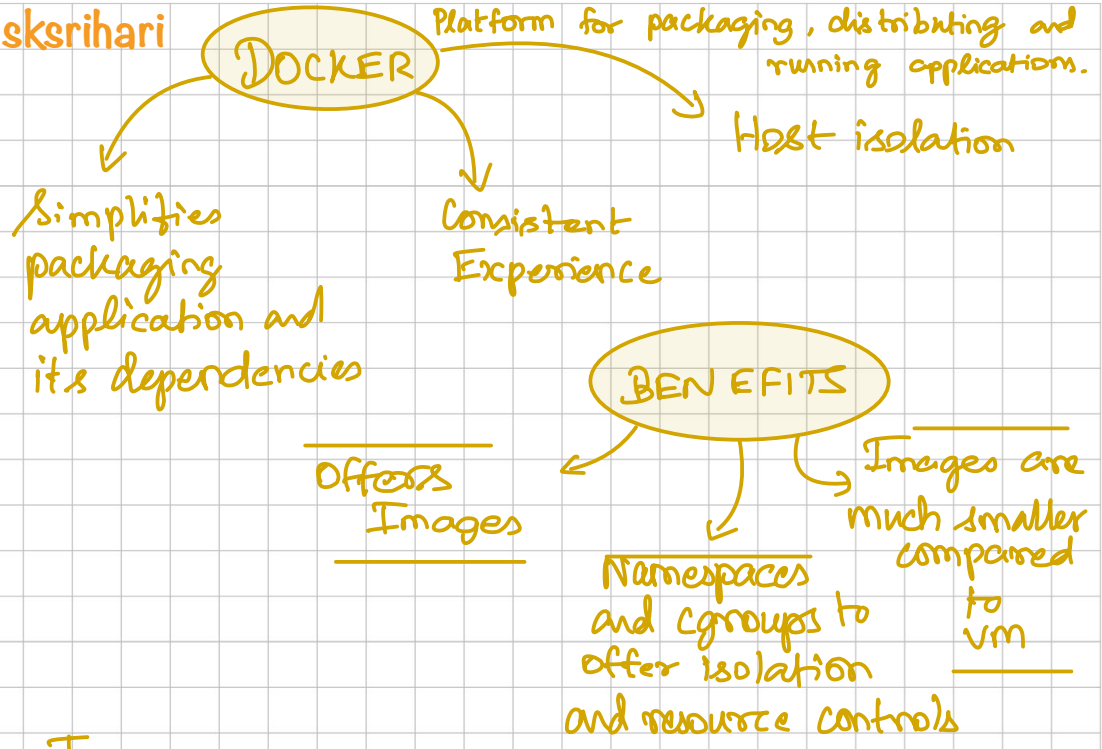
* How can you limit the amount of system resources a container can consume?

Answer: Cgroups → Control Groups

Linux kernel feature that limits the resource usage of a process (or a group of processes)

A process cannot use more than the configured amount of CPU, memory, network bandwidth, and so on.

Cgroups prevent processes from hogging resources reserved for other processes.



Images

The image contains the filesystem and the path to the application executable when you run the image.

Registries

A Docker registry is a repository that stores your Docker images and allows easy sharing of those images between different people and computers.

Container

A running container is a process running on the host running Docker, but it's completely isolated from host and other processes.

@asksrihari

Docker Workflow

① Build image & push to registry ② Pull image ③ Run image

See Container Management using Docker.

ⓒ What is kubernetes ?

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 size of the cluster.

Master Nodes → deploys → cluster of Worker Nodes

Dev submits App Descriptor

K8S Cluster Architecture

Master Node
- API Server
- Scheduler
- Controller
- etcd

Worker Nodes
- Kubelet
- Kube Proxy
- Container Runtime

Srihari Sridharan

Running Applications in K8S

1. Package your appln. in to one or more containers
2. Push those images to an image registry
3. Post app descriptor to K8S API server
4. Scheduler schedules the containers on available workers
5. Kubelet instructs nodes to download container images
6. Kubelet instructs nodes to run the containers

Benefits

Simplifies appln. deployment
Better hardware utilization
Health monitoring and self healing
Automatic scaling.

Kubernetes Objects

Kubectl <operation> <object> <resource name> <optional flags>

Pod Smallest unit that K8S manages

Made up of one or more containers

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

Pods aren't durable!

Srihari Sridharan

All containers for a pod will be running 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 containers.

A pod has an explicit lifecycle and will remain in a node 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 K8S creates internal to the cluster.

If no namespace is mentioned K8S assumes you are working with default namespace.

Nodes

Node is a machine that is added to the kubernetes cluster.

The master node is the brain of K8S 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 k8s cluster are expected to be connected to each other and share a private cluster-wide network.

K8S runs containers within a pod within this isolated network.

Handling IPs, DNS entries etc.

Controllers

Desired state deployment
Controllers are the brain behind this

ReplicaSets Associated with a Pod and indicates how many instances of that Pod should be running within the cluster

also implies a controller that watches ongoing state and knows how many of your pod to keep running.

Replica set is commonly wrapped in turn by a deployment.

Deployment Recommended way to run code on K8S

↓

Wraps around and extends the replica set

Includes metadata settings to know how many pods to keep running.

Services K8S resource used to provide an abstraction through to your pod agnostic of the specific instances that are running.

Can contain a policy.

Emulates a software load balancer within Kubernetes.

© What is AKS?

Self hosting K8S cluster

Manually install the master and worker nodes.

LOT OF WORK!

↓ Handled by AKS

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

AKS. Deployment, management and operations of Kubernetes

Provisioning, upgrading and scaling on demand.

Master node is managed by Azure
Offload responsibility to Azure
Pay for agents only

Kubectl is part of cloud shell.

Benefits

- Version upgrades and patching
- Easy cluster scaling
- Self healing hosted control plane (master)
- cost savings

(+) Beyond Managed K8S

Think about the ecosystem of the cloud vendor.

Azure Container Instances (ACI)

Azure Container Registry (ACR)

Azure Service Fabric (SF)

Azure App Service (AAS)

Azure Batch Service (ABS)

ACI

Deploy containers without worrying about underlying infrastructure

Easily start deploying containers for targeted usecases.

'Dockerize' your application and execute it in one click.

ACR Repository for container images.

SF Foundational technology powering core Azure infrastructure as well as other Microsoft services.
Highly available and durable services at cloud scale.
Microsoft's container orchestrator.

AAS PaaS Platform
Built in autoscaling and load balancing and CI/CD with GitHub.

ABS Cloud scale job scheduling and compute management.
Batch pool to run tasks in Docker containers.

Tool Batch Shipyard!

2. AKS in Action

@ Quick tour of the sample app.

https://github.com/manojnair/myapp

Docker images

https://hub.docker.com/r/manojnair/myapp/tags

Run

> docker container run --name myapp -d -p 8081:80 manojnair/myapp:v1

visit http://localhost:8081

Similarly run V2, V3 and V4 in 8082, 8083 and 8084

NOTE: You can use the ones above or write your own apps and build images

ⓑ Deploying an AKS cluster

RG aks-rg1 Stuff using Azure portal / CLI

Create k8s cluster

Various blades in portal

Basic | Node Pools | Authentication | Networking

Basic → Set defaults

Node Pools → Better to use system assigned managed identity. Relieves the burden of renewing the

Authentication | Integration | Tags

@asksrihari credentials for service principal.

RBAC : enable

Network Configuration → Azure CNI

↳ Look at course Implementing Managed Identities for Microsoft Azure Resources

Integration

Azure monitor default workspace for log analytics.

Tags Tag your resources as required

In azure CLI

az account show

az configure --defaults group=aks-rg1

Set RG for subsequent CLI commands

az aks get-credentials --name <cluster name>

eg az aks get-credentials --name aksdemo1

Fetch the credentials and merge it into current context.

Whatever you used in Basic section.

az aks install-cli ← to install kubectl

Srihari Sridharan

© Deploy the application to AKS cluster imperatively.

K8S management techniques.

Imperative commands	for development projects	low learning curve
Imperative object configuration	for production projects	Moderate learning curve
Declarative object configuration	for production projects	high learning curve

This sample uses (imperative commands) approach

```
kubectl create deployment myapp
        --image=manojnair/myapp:v1
        --replicas=1
```

Service

Creating a service of type load balancer

In AKS this creates an azure load balancer

```
kubectl expose deployment myapp --type=LoadBalancer
        --port=80 --target-port=80
```

Before running the above command have another terminal with watch enabled.

```
kubectl get svc --watch
```

Creates the load balancer!

d) Scaling the deployment manually

To scale deployments

```
kubectl scale deployment myapp --replicas=3
```

Application is powered by 3 replicas!

To get max pods that can be supported by a cluster and agent pool

```
az aks nodepool show --cluster-name aksdemo1
    --name agentpool --query "maxPods"
```

Note If maxPods is 110 k8s cannot create these many pods for your application as

```
kubectl get pods --namespace kube-system
```

Add more worker nodes incase you wish to increase the number of replicas (pods) beyond the node's max pods.

@ Scaling nodes manually

```
kubectl get nodes
```

returns the number of nodes.

to scale nodes

```
az aks scale
  --resource-group aks-rgl
  --name aksdemo1
  --nodecount 2
  --no-wait
```

returns immediately we don't need to wait!

To check the status

```
az aks nodepool show --name agentpool
     --cluster-name aksdemo1
     --query "[count, provisioningState]"
```

use

```
kubectl get nodes
```

(f) Updating the application

Rollouts Manage the rollout of a resource
Valid resource types include deployments, daemonsets and statefulsets.

Usage

```
kubectl rollout SUBCOMMAND
```

To check the rollout history

```
kubectl rollout history deployment/myapp
```

Describe the deployment

```
kubectl describe deployment myapp
```

You can find the container and image details
Image manojnair/myapp:v1

Method 1

```
kubectl set image
        deployment/myapp
        myapp = manojnair/myapp:v2
        --record = true
```

Tracks the change ↗

As of 2022 FEB there is a warning that --record will be deprecated.

Refresh the app and you will see V2.

Method 2

```
kubectl edit deployment myapp
--record=true
```

Refresh the app and you will see V3.

Check

```
kubectl rollout history deployment/myapp
```

Shows the history of changes.

9) Rolling back the applications to previous versions

To undo a deployment

```
kubectl rollout undo deployment/myapp
```

NOTE: If you keep running this command repeatedly, then you will basically keep toggling between the last 2 deployments

To revert to specific version in history

```
kubectl rollout undo deployment/myapp
--to-revision 1
```

IMPORTANT NOTE

When we undo a rollout, K8S creates a new item in history and removes the old entry. (This explains why it keeps swapping between last two rollouts when we undo twice or more)

(h) Using declarative approach to deploy kubernetes objects

myapp2.yml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: myapp2
  labels:
    app: myapp2
spec:
  replicas: 3
  selector:
    matchLabels:
      app: myapp2
  template:
    metadata:
      labels:
        app: myapp2
    spec:
      containers:
```

@asksrihari

```
- name: myapp2
  image: manojnair/myapp:v2
  ports:
    - containerPort: 80
---
apiVersion: v1
kind: Service
metadata:
  name: myapp2
spec:
  selector:
    app: myapp2
  type: LoadBalancer
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
```

Pod image

separator for multiple resources in a file

load balancing service

Build K8S objects declaratively and manage using source control

```
kubectl apply -f ./myapp2.yml
```

To clean up

```
kubectl delete deployment myapp
kubectl delete service myapp
```

(i) Pushing the images to ACR

Azure Container Registry

ACR + AKS - Best of both worlds!

Create ACR in portal or using IAC.

Registry name, location, SKU → Basic / Premium

SKU → Stock Keeping Unit.

registry_name.azurecr.io

eg. myappacr01

Networking and encryption are available only for premium SKU.

Validate and create

AKS agent pool nodes need permission to pull images from ACR.

Access Control → Add Role Assignment

Role - reader

Select the system assigned managed identity →

aksdemo1-agentpool identity.

SAVE!

Push images to ACR

```
docker image ls
```

```
docker tag manojnair/myapp:v1
           myappacr01.azurecr.io/myapp:v1
```

```
az acr login --name = myappacr01
```

```
docker push myappacr01.azurecr.io/myapp:v1
```

Create the deployment with the ACR image

```
kubectl create deployment myappacr
    --image = myappacr01.azurecr.io/myapp:v1
```

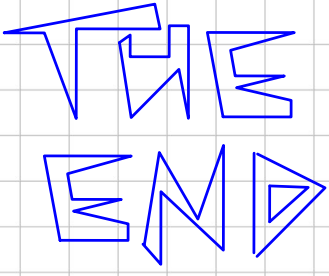
This command can be run and the images can be pulled only by agent pool nodes.

```
kubectl expose deployment myappacr
        --type = LoadBalancer
        --target-port = 80
        --port = 80
```

Image is securely stored in ACR!

Next Steps

- Learning paths on Pluralsight k8s
  - Certified Kubernetes Administrator
  - Certified Kubernetes Application Developer
  - Kubernetes Administration
  - Using kubernetes as a developer
- Azure Kubernetes Service Workshop
  - https://aka.ms/learn/aksworkshop



Srihari Sridharan