Kubernetes at war on Azure

Krzysztof Pudłowski
Łódź, 23.10.2019

About me...

Long-time corporate employee. Currently, he works in the telecommunications sector. Physicist by profession, education completed with a doctorate from the University of Łódź. Practical knowledge of enterprise software development. Administrator, developer, team leader, IT expert. He holds professional certificates: MCP, MCSA: SQL Server 2012/2014, MCSE: Data Management and Analytics, MCSA: Cloud Platform, MCSE: Cloud Platform and Infrastructure , MPP Data Science , Microsoft Certified: Azure Administrator Associate , Microsoft Certified: Azure Data Scientist Associate

Blogger at wchmurze.cloud

Azure Kubernetes Service Changelog
Release 2019-10-14
This release is rolling out to all regions
Service Updates

With the official 2019-11-04 Azure CLI release, AKS will default new cluster creates to VM Scale-Sets and Standard Load Balancers (VMSS/SLB) instead of VM Availability Sets and Basic Load Balancers (VMAS/BLB).

From 2019-10-14 AKS Portal will default new cluster creates to VM Scale-Sets and Standard Load Balancers (VMSS/SLB) instead of VM Availability Sets and Basic Load Balancers (VMAS/BLB). Users can still explicitly choose VMAS and BLB.

From 2019-11-04 the CLI extension will have a new parameter --zones to replace --node-zones, which specifies the zones to be used by the cluster nodes.

Be prepared to changes in Azure

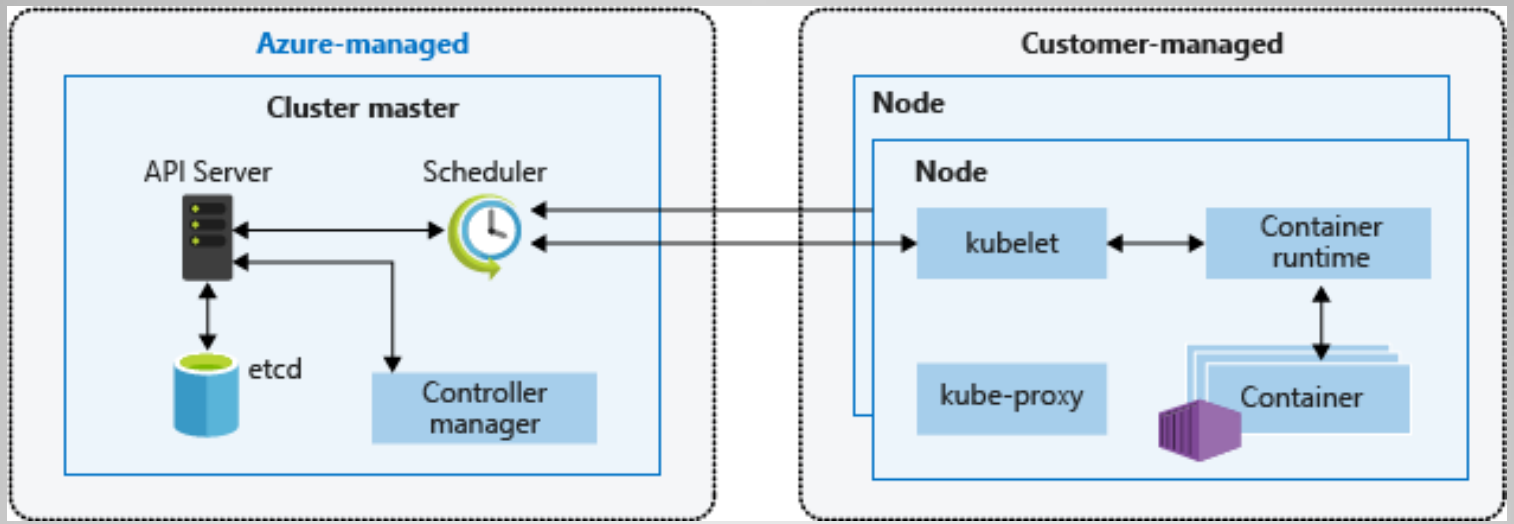
Resources to read

- https://bit.ly/2BIcEgd
- https://github.com/djkormo/k8s-AKS-primer

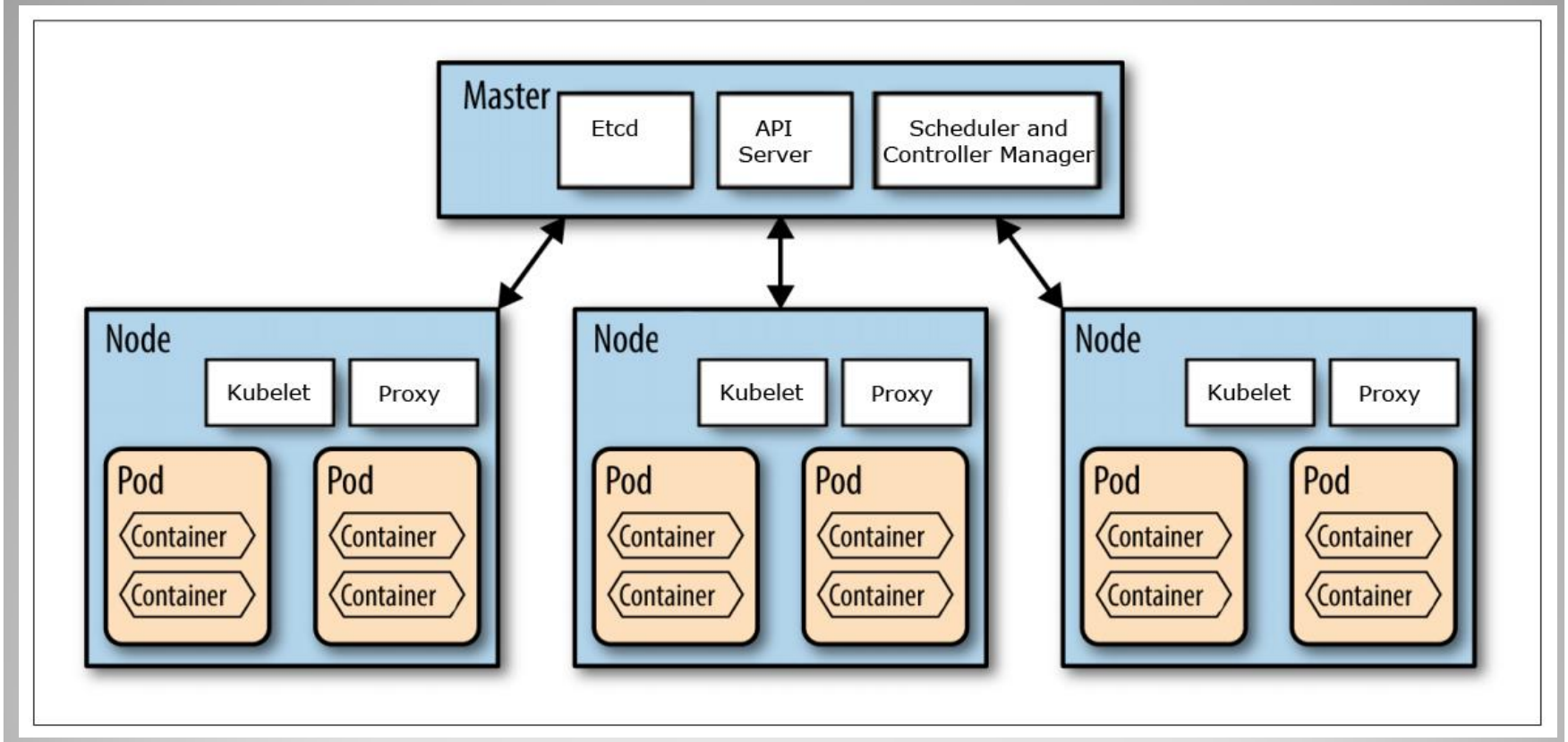
- https://www.youtube.com/watch?v=PH-2FfFD2PU

  Cluster master nodes provide the core Kubernetes services and orchestration of application workloads.

  Nodes run your application workloads.



Kubernetes in 5 minutes



Kubernetes in 5 minutes

Kubernetes limits

At v1.16, Kubernetes supports clusters with up to 5000 nodes. More specifically, we support configurations that meet *all* of the following criteria:

No more than 5000 nodes

No more than 150000 total pods

No more than 300000 total containers

No more than 100 pods per node

https://kubernetes.io/docs/setup/best-practices/cluster-large/

Limits, requests

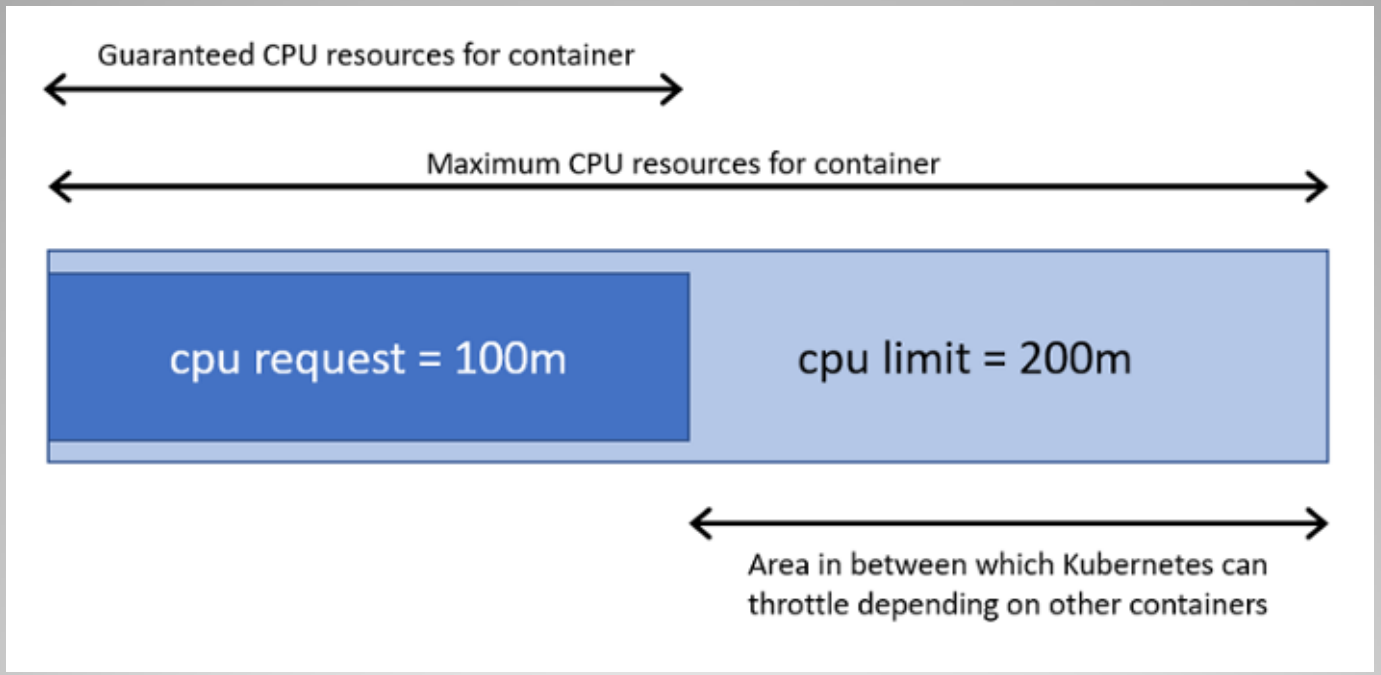
- Kubernetes employs requests and limits to control resources. Requests are guaranteed resources that a container is entitled to use. Limits, on the other hand, are the maximum resources or threshold a container can use.
- After reaching the limits, containers will be restricted. If a container requests a resource, Kubernetes will only schedule it on an available node that can provide those resources. These resources and limit are defined in the standard YAML configuration of your containers.
- In Kubernetes, there are two types of resources: CPU and Memory. CPU is measured in core units, and memory is specified in bytes.

CPU

- CPU resources are measured in millicore. If a node has 2 cores, the node's CPU capacity would be represented as 2000m. The unit suffix m stands for "thousandth of a core."
- 1000m or 1000 millicore is equal to 1 core. 4000m would represent 4 cores. 250 millicore per pod means 4 pods with a similar value of 250m can run on a single core. On a 4 core node, 16 pods each having 250m can run on that node.

Memory

- Memory is measured in bytes. However, you can express memory with various suffixes (E,P,T,G,M,K and Ei, Pi, Ti, Gi, Mi, Ki) to express mebibytes (Mi) to petabytes (Pi). **Most simply use Mi**.
- Like CPU, pods will never be scheduled if they require more resources than the capacity of a node. Unlike CPU, memory is not compressible. You can't make memory run slower or faster like CPU or network throttling. Pods will be terminated if it reaches the memory limit.



https://jaxenter.com/manage-container-resource-kubernetes-141977.html

https://vincentlauzon.com/2019/04/02/requests-vs-limits-in-kubernetes/

Limits, requests

```
apiVersion: v1
kind: Pod
metadata:
  name: pod-quota-1
spec:
  containers:
  - name: pod-quota-1
    image: nginx
```

```
apiVersion: v1
kind: Pod
metadata:
  name: pod-quota-2
spec:
  containers:
  - name: pod-quota-2
    image: redis
    resources:
      limits:
        memory: "1Gi"
        cpu: "800m"
      requests:
        memory: "700Mi"
        cpu: "400m"
```

Container limits

Namespaces

- Beyond the individual container resources, you may want to investigate setting limits on namespaces. So what is a namespace? Namespaces can be used to define a cluster of applications, departments, or environments. Simply, Namespace refers to scope or grouping of objects in a Kubernetes cluster
- At the Namespace level, you can set up ResourceQuotas and LimitRanges.

```
apiVersion: v1
kind: ResourceQuota
metadata:
  name: compute-resources
spec:
  hard:
    pods: "20"
    requests.cpu: "1"
    requests.memory: 1Gi
    limits.cpu: "2"
    limits.memory: 2Gi
    requests.nvidia.com/gpu: 3
```

https://github.com/djkormo/k8s-AKS-primer/tree/master/examples/quotas

Resource quota

```
apiVersion: v1
kind: LimitRange
metadata:
  name: limit-mem-cpu-per-container
spec:
  limits:
  - max:
      cpu: "800m"
      memory: "1Gi"
    min:
      cpu: "50m"
      memory: "50Mi"
    default:
      cpu: "200m"
      memory: "200Mi"
    defaultRequest:
      cpu: "100m"
      memory: "100Mi"
    type: Container
```

Limit range

DEMO

Resources quota/limit range

```
apiVersion: v1
kind: ResourceQuota
metadata:
  name: compute-resources
spec:
  hard:
    pods: "20"
    requests.cpu: "1"
    requests.memory: 1Gi
    limits.cpu: "2"
    limits.memory: 2Gi
    requests.nvidia.com/gpu: 3
```

Resource quotas

- **kubectl describe node**
- **kubectl top nodes**
- **kubectl top pods --all-namespaces**

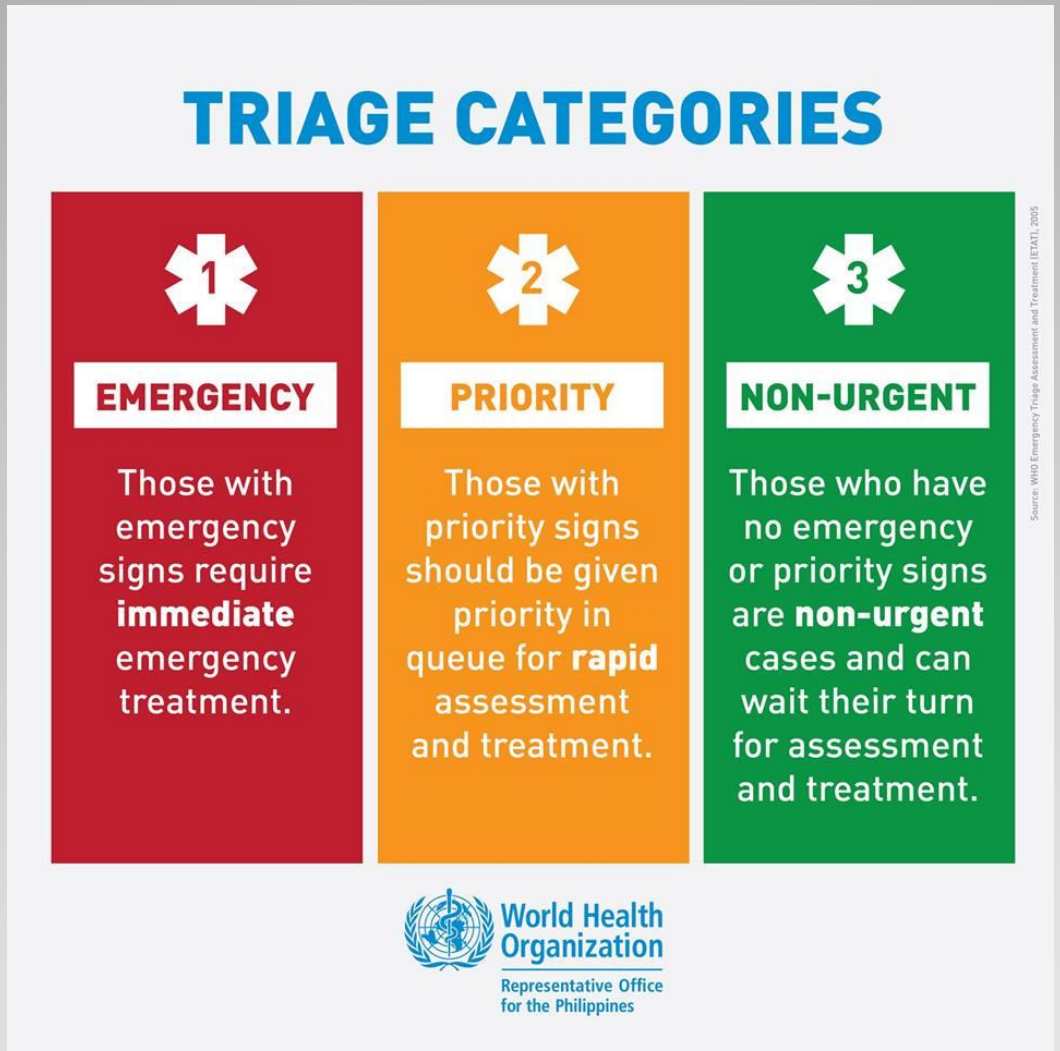
Nodes limits

https://github.com/Azure/aks-engine/blob/master/docs/topics/clusterdefinitions.md

"--max-pods"	"30", or "110" if using kubenet --network-plugin (i.e., "networkPlugin": "kubenet")

"--node-monitor-grace-period"	"40s"
"--pod-eviction-timeout"	"5m0s"

Node limits


TRIAGE CATEGORIES
1
EMERGENCY
Those with emergency signs require immediate emergency treatment.
2
PRIORITY
Those with priority signs should be given priority in queue for rapid assessment and treatment.
3
NON-URGENT
Those who have no emergency or priority signs are non-urgent cases and can wait their turn for assessment and treatment.
Source: WHO Emergency Triage Assessment and Treatment (ETAT), 2005
World Health Organization
Representative Office for the Philippines

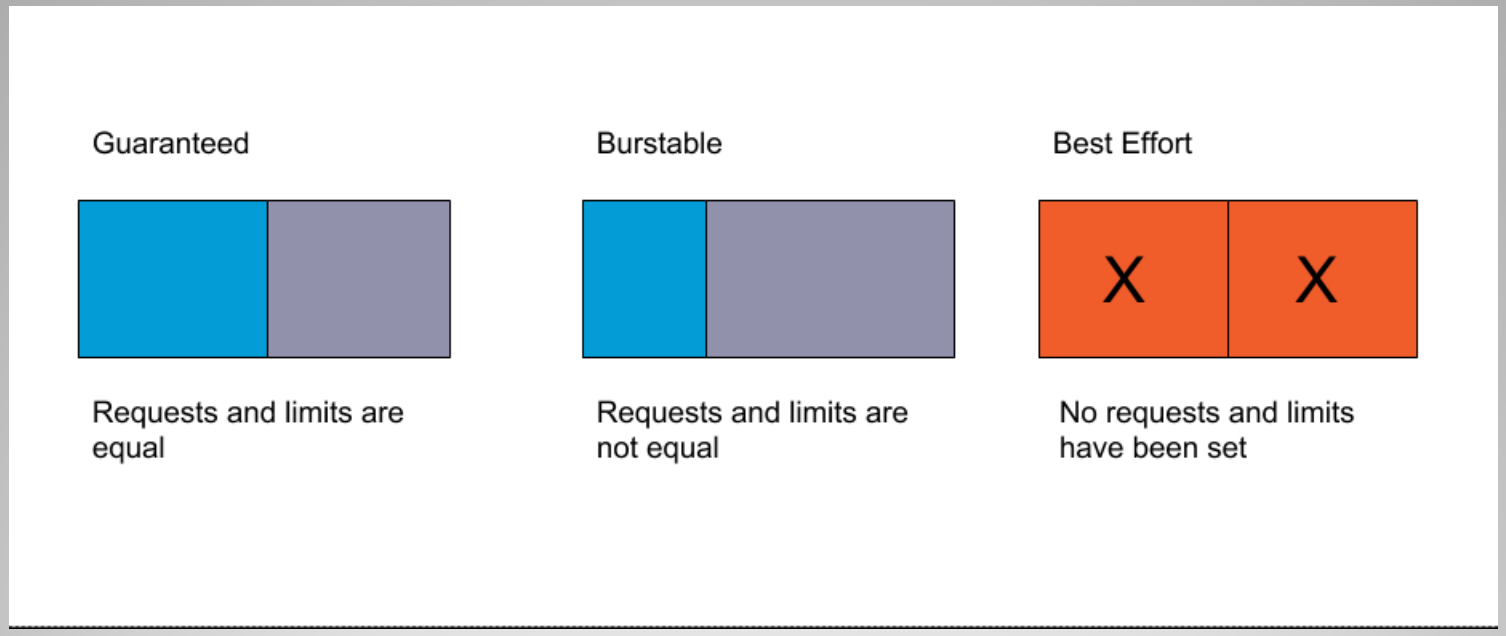
Triage

QoS (Quality of Service) Classes

When Kubernetes creates a pod, it assigns one of these three QoS classes:

Guaranteed
Burstable
BestEffort

Scheduler and QoS



https://www.weave.works/blog/kubernetes-pod-resource-limitations-and-quality-of-service

Quality of Service (QoS)

Guaranteed QoS

For a Pod to be given a QoS class of Guaranteed:

Every Container in the Pod must have a memory limit and a memory request, and they must be the same.

Every Container in the Pod must have a CPU limit and a CPU request, and they must be the same.

```
resources:
    limits:
      memory: "200Mi"
      cpu: "700m"
    requests:
      memory: "200Mi"
      cpu: "700m"
```

https://kubernetes.io/docs/tasks/configure-pod-container/quality-service-pod/#create-a-pod-that-gets-assigned-a-qos-class-of-guaranteed

Burstable QoS

Burstable class have two conditions:

They don’t meet the QoS guaranteed criteri a.

They have at least one memory or CPU req uest.

BestEffort is a container with no memory or CPU limits or requests. If you've never defined any resources before reading this, your pod is by default running in BestEffort.

Simply, BestEffort is when you don't set anything at all.

BestEffort QoS

Quality of Service Best Practices

- For production workloads, BestEffort is not recommended. Keep in mind, these containers will be killed first. Burstable is suitable for most generic workloads. For sensitive applications that may have spikes or anything that runs in a stateful set like databases, it is recommended to apply a QoS guaranteed class.

Node Failure

This is the workflow of what happens when a node gets down:

1- The Kubelet posts its status to the masters using ***–node-status-update-frequency=10s***

2- A node dies

3- The kube controller manager is the one monitoring the nodes, using ***––node-monitor-period=5s*** it checks, in the masters, the node status reported by the Kubelet.

4- Kube controller manager will see the node is unresponsive, and has this grace period ***–node-monitor-grace-period=40s*** until it considers the node unhealthy. This parameter must be N times ***node-status-update-frequency*** being N the number of retries allowed for the Kubelet to post node status. N is a constant in the code equals to **5**

DEMO

https://github.com/djkormo/k8s-AKS-primer/tree/master/examples/failure

Failure