Creation of resource group myResourceGroup in eastus location (Resource group - logical group in which Azure resources are deployed and managed)

```
az group create --name myResourceGroup --location eastus
```

Enable_cluster_monitoring:

1. Check if Microsoft.OperationsManagement and Microsoft.OperationalInsights are registered on subscription

```
az provider show -n Microsoft.OperationsManagement -o table
az provider show -n Microsoft.OperationalInsights -o table
```

2. If not registered, register them:

```
az provider register --namespace
Microsoft.OperationsManagement
az provider register --namespace Microsoft.OperationalInsights
```

3. Instantiate AKS Cluster

```
az aks create --resource-group myResourceGroup --name
myAKSCluster --node-count 1 --enable-addons monitoring --
generate-ssh-keys
```

i. `–enable-addons monitoring` - to enable Azure monitor for containers (https://docs.microsoft.com/en-us/azure/azure-monitor/containers/container-insights-overview)

Connect_to_the_cluster:

1. Configure **kubectl** to connect to kubernetes cluster
    1. Download credentials and to configure **kubectl** (Updates **~/.kube/config** (configuration file used by kubectl))

```
az aks get-credentials --resource-group myResourceGroup --name
myAKSCluster
```

2. Verify connection to the cluster

```
kubectl get nodes
```

3. Run application:

i. `vi Dockerfile`

```
FROM php:5-apache
COPY index.php /var/www/html/index.php
RUN chmod a+rx index.php
```

ii. `vi index.php`

```php
<?php
    $x = 0.0001;
    for ($i = 0; $i <= 1000000; $i++) {
        $x += sqrt($x);
    }
    echo “OK!”;
?>
```

iii. vi php-apache.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
    name: php-apache
spec:
    selector:
        matchLabels:
            run: php-apache
    replicas: 1
    template:
        metadata:
            labels:
                run: php-apache
        spec:
            containers:
            - name: php-apache
              image: k8s.gcr.io/hpa-example
              ports:
              - containerPort: 80
              resources:
                  limits:
                      cpu: 500m
                  requests:
                      cpu: 200m
—
apiVersion: v1
kind: Service
metadata:
    name: php-apache
    labels:
        runs: php-apache
spec:
    ports:
    - port: 80
    selector:
        run: php-apache
```

iv. Deploy the application:

```
kubectl apply -f php-apache.yaml
```

4. Creation of Horizontal Pod Autoscaler

```
kubectl autoscale deployment php-apache –cpu-percent=50 –min=1
–max=10
```

5. Check current status of autoscaler:

```
kubectl get hpa
```

6. Update existing AKS cluster to enable cluster autoscaler

```
az aks update \
    –resource-group myResourceGroup \
    –name myAKSCluster \
    –enable-cluster-autoscaler \
    –min-count 1 \
    –max-count 3
```

    i. It takes a few minutes to update cluster and configure cluster autoscaler settings

7. Increase Load

```
kubectl run -i –tty load-generator –rm –image=busybox –
restart=Never – /bin/sh -c "while sleep 0.01; do wget -q -O-
http://php-apache; done"
```

    i. Within a minute or so, we should see higher CPU load by executing the following:

```
kubectl get hpa
kubectl get deployment php-apache
kubectl get nodes
```

8. Stop load

```
kubectl get hpa
kubectl get deployment php-apache
kubectl get nodes
```

To_Scale_Up_Faster:

1. Update Horizontal Pod Autoscaler with more Pods:

```
kubectl edit hpa php-apache
```

    i. Change the following entries:

```
spec:
```

```
    maxReplicas: 20
    …
    targetCPUUtilizationPercentage: 20
```

2. Update Cluster Autoscaler with More Nodes:

```
az aks update –resource-group myResourceGroup –name
myAKSCluster –update-cluster-autoscaler –max-count 4
```

3. Increase Load

```
kubectl run -i –tty load-generator –rm –image=busybox –
restart=Never – /bin/sh -c "while sleep 0.01; do wget -q -O-
http://php-apache; done"
```

Delete the setup:

```
az group delete –name myResourceGroup –yes –no-wait
```