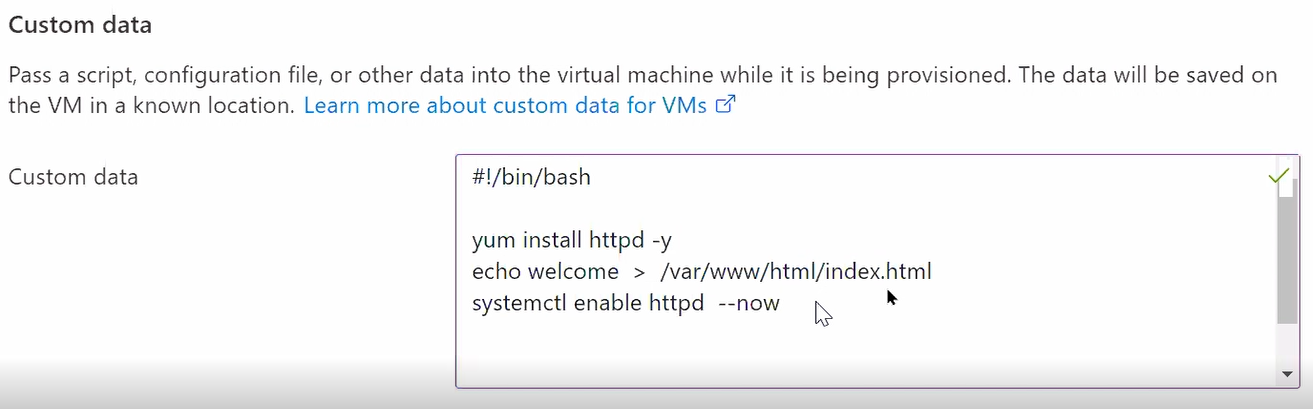
**Day 1**

* Pay as you go model
* Data center
* Cloud computing
* If you get ram/CPU and OS from azure known as instance or virtual machines.
* Availability zone
* High availability
* No Reputation loss
* Region
* In azure cloud it is compulsory to create a **resource group** (like namespace in k8s).
  + Storage, virtual machines and all other resources we have to put inside this resource group.
  + It will isolate your resources.
  + In real world multiple teams are working with one single account.
    - So by this way we can set budget and manage many other things for multiple teams.
  + One you terminate a resource group complete resources will be deleted we don’t have to worry about this.
* Availability set
  + It will create a multiple copy in different racks.
  + Also known as **rack awareness**.
  + Maximum they will give 3 fault domains.
    - 3 different racks for multiple copies.
  + What is difference between fault domains & update domains.
* Here VPC is known as virtual network (VNET).
* While launching vm in advanced menu we have **custom data** using this we can do some operation automatically without login.



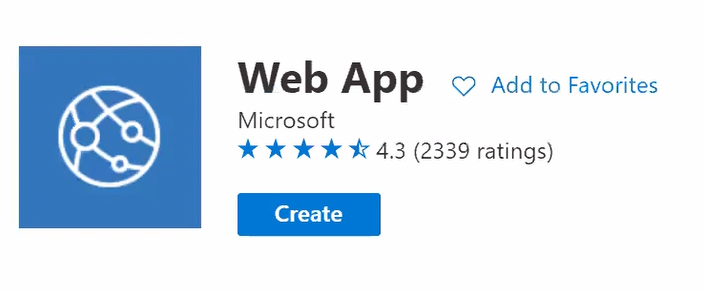
* At the end of any vm or any other service creation they will give you one **template** which you can use this in future purpose.
  + You can also download it.
* After creating a vm you can see it in resource groups.
* In azure cloud they have by default enable firewall of redhat OS.
  + Firewall-cmd --add-service=httpd
* Load balancer As A Service
  + We always give client a frond end IP.
  + After creating lb you have to add your backend pool.
* For azure cli we have command called **az**.
  + Az login
* You can use **cloud shell** also.
  + You can also switch here between power shell and bash shell.
  + Az vm list

**Day 2**

* Az vm list --output table
* Now days there are mainly two ways for launching a OS.

1. Vm
2. Containerization

* One of the famous tools for launching a container is docker.
  + We can use k8s and his master-slave architecture for easy management.
  + But if we configured master and all other setup manually it is very time consuming and hectic.
  + So instead of configured all the things manually we can use managed service of azure.
    - For this we have service called **azure Kubernetes service (aks).**
* Az aks show
* Az aks create -g lwgrp1 -n kubecluster1 --node-count 2 --generate-ssh-keys –enable-addons monitoring
  + Here we don’t have to give master node.
    - By default, they will create a master for you.
  + In python library for aks we have a small issue.
    - If you run any command in aks function and in your folder name anywhere you have space it will give you error.
    - This thing comes from generate-ssh-keys.
  + Ssh-keygen -f lw-ssh
  + Az aks create -g lwgrp1 -n kubecluster1 --node-count --enable-addons monitoring --ssh-key-value lw-ssh.pub
  + Az aks install-cli --install-location=kube/
  + Az aks get-credentials --name kubecluster1 --reource-group lwgrp1
  + Kubectl get pods
* Kubectl create deployment web1 --image=vimal13/apache-webserver-php
  + Kubectl expose deploy web1 --port=80 --type=LoadBalancer
    - It will use azure load balancer.
* **PAAS using github.** 
  + We can also use CI/CD pipline using Jenkins and integrate with this.
  + For implementing PAAS we have resources called **azure app services**.
    - You just have to upload your code here.
    - We have build our code by our own self, but we want updating of application will also be automatic.
    - Here we can use Github.
      * Github will go to our app(PAAS) automatically with something changed and ask him to change this.
      * Now PAAS go to github and copy code.
    - For this we have to set workflow in github.
* Search for web app.



* + App name should be unique because they are giving you some DNS.
  + Click language you want (python).
  + When you want to use Web App service you have to tell your plan (app service plan).
  + Click on manage deployment center.
    - Select github.
    - Connect your account.
    - As soon as you give account details, deployment center will send some file which you can see in your github portal.
    - It will also create a **workflow**, which is available in action menu of github.
    - Github has create a workflow and they will push it to the azure app service.
      * Here we don’t have python file so it will give you error.
    - As soon as you push to Github a code, workflow in github automatically start and it will send your code to the azure app service.
      * While pushing if it will give you error, first fetch then push.
* You can also use CLI for this.
  + Az webapp list --output table
  + They will upload some dummy code for you.
  + For continuous deployment azure has one program called.
* Create a github repository.
  + Create a simple code.
  + Upload this code into github.
* **Object storage** 
  + Azure has a service called **blob** for STAAS.
  + Search for storage account.
  + Feel some basic details.
  + In azure **bucket** is known as **container**.
  + Now you can upload a file.
* Database & role summary high level.
* **Day 1**
  + What is cloud computing?
  + Resource group
  + Launching virtual machine
    - Availability set
      * Fault domain
      * Update domain
    - VNET
    - Custom data
    - Template
  + Load balancer as a service
    - Backend pool
  + CLI
    - Az vm list
    - Az vm list --output table
  + Cloud shell
* **Day 2** 
  + Kubernetes
    - Aks
  + Azure app services
  + Web app
  + Workflow
  + Object storage
    - Bucket = container
  + Database
  + Role