Azure

Blob

Access Tier → Hot

→ Cool (atleast 30 days) (20% savings)

↓ saves money on storage

Archive (cheapest) atleast 180 days (90% savings)

24 hours required to access after placing request.

SAS Key

↳ Specify Time, permissions

IP addresses

Https / Http

Signed key

NS :- StackExchange.Redis

Microsoft.Azure.Management.Redis

FLUSHALL ASYNC

↳ ~~Delete all~~ keys in background in a different thread.

Rediscache → Access keys → Primary connection string

Secondary connection string

```
Private Static Lazy<ConnectionMultiplexer> lazyConnection =
                                    new Lazy<ConnectionMultiplexer>
{
    return ConnectionMultiplexer.Connect(cacheConnection);
});
Public Static ConnectionMultiplexer Connection
    Get
        return lazyConnection.Value
```

Date
Page

→ get from azure rediscache page

Private static string cacheConnection = " ";

main

{ IDatabase cache = lazyConnection.Value.GetDatabase();
cache.StringGet("session333").ToString();
cache.StringSet("session333", "writing");
cache.KeyExpire("session333", DateTime.Now.AddMinutes(1));

lazyConnection.Value.Dispose();

}

Cache.Stringset
String get
Hash get
Hash set
Sortedset

CDN
- microsoft
- Akamai
- Verizon

} has dynamic delivery feature

1) Create CDN Profile
2) Create CDN endpoint
Name —
OriginType - Storage
- webapp
- custom origin
- cloudservice

Origin hostname
origin path
Origin hostheader

Protocol http 80
https 443

optimized for → General web delivery
Static content

Redis - backend cache
CDN - frontend Cache

Purge → CDN purged on new deployment

Vm → Diagnostic logs
↳ send to app insights

Function app → app insights

logs → archive to storage account
Stream to event hub
Send to log analytics

Date
Page

Create vm :-

New - AzResourcegroup
- Resource Group Name " "
- Location " "

New - AzVM
- Resource Group Name " "
- Location " "
- Name " "
- VirtualNetworkName " "
- SubnetName " "
- SecurityGroupName " "
- PublicIPAddress " "
- Credential $cred

$cred = Get-Credential

Updat - AzVm

Stop - AzVm - Force

Start - AzVm

vm encryption :

Register - AzResource Provider - ProviderNamespace "Microsoft.KeyVault"

New - AzResourceGroup - Location " "
- Name " "

New - AzKeyVault - Location " "
- ResourceGroup Name " "
- VaultName " "
- EnabledForDiskEncryption

classmate
Date
Page

Add - AzureKeyVaultKey -VaultName " "
-Name " "
-Destination "Software")

Set - AzVmDiskEncryption -ResourceGroupName " "
- VmName " "
- DiskEncryptionKeyVaultUrl " " → KeyVault.VaultUri;
-DiskEncryptionKeyVaultId → KeyVault.ResourceId
- KeyEncryptionKeyUrl → ~~KeyVault.ResourceId~~
- KeyEncryptionKeyVaultId ↓
(Get-AzKeyVaultKey -VaultName "
-Name).Key.Kid;

Set - AzVmDiskEncryptionStatus -ResourceGroupName " "
-VmName " ";

az keyvault create --name
-resource-group
-location
-enabled-for-disk-encryption True

az keyvault key create --vaultName
--name
--protection software

az vm create

az vm encryption enable

Batch :-

```
CloudBlobClient blobClient = StorageAccount.CreateCloudBlobClient();
```

Upload files to blob.

Create Batch Pool

```
CloudPool pool = batchClient.PoolOperations.CreatePool( , );
                                                  poolNodeCount
                                                  poolId, PoolVmSize,
                                                  VmConfiguration
pool.Commit();
```

Create Batch Job :-

```
CloudJob job = batchClient.JobOperations.CreateJob();

Job.Id = JobId;
Job.PoolInformation =

Job.Commit();
```

Create Task :-

```
CloudTask task = new CloudTask(taskId, taskCommandLine);
                                       ↑ inputfile name

task.ResourceFiles = new List<ResourceFile> {inputFiles};

tasks.Add(task);

batchClient.JobOperations.AddTask(JobId, tasks);
```

AKS:

az aks create

--resource-group

--name

--node-count

--enable-addons

--generate-ssh-keys

Connect to cluster:

az aks install-cli

az aks get-credentials --resource-group "" --name ""

kubectl get nodes

Kube manifest file:

azure...yaml

kubectl apply -f azure...yaml
(deploy the application)

kubectl get service azure-vote-front --watch

Build docker image:

docker build "." (Build & tag)
-t aci-app
↓
tag

docker images
docker run -d → run in background
-p → publish port
8080:80 app

az acr create --resource-group " "
--name <acrname>
--sku Basic
--admin-enabled true

az acr login --name <acrname>
az acr show --name <acrnam>
--query loginServer
--output table

docker images

docker tag aci-app <acrloginserver> --:v1
docker push <acrloginserver> --:

deploy containers:→

az container create --resource-group " "
--name " "
--image <acrloginserver>
--cpu
--memory
--registry-login-server " "
--registry-username " "
--registry-password " "
--dns-name-label
--ports

az container show
az container logs --resource-group " "
--name " "

Enable Push notifications:-

```
using Microsoft.Azure.NotificationHubs;

String notificationhubname =
String notificationhubconnection = . . .

NotificationHubClient hub = NotificationHubClient.CreateClientFrom
                            ConnectionString(notificationhubconnection,
                                             notificationhubname);

Var result = await hub.SendGcmNativeNotificationAsync()
```

Offline Sync:

IMobileServiceTable (Table)

To support offline use, your app should use the sync table APIs such as IMobileServiceSyncTable

Sync context :- IMobileServiceClient

Date
Page

App Service:

az webapp deployment user set --username " "
--password " "

az group create --name " "
--location " "

az appservice plan create --name " "
--resourcegroup " "
--sku FREE

az webapp create --resourcegroup " "
--plan " "
--name " "
--deployment-local-git

az resource update --name " "
--resource group " "
--namespace Microsoft.Web
--resource-type config
--parent sites/<app-name>
--set properties.cors.allowedOrigins
="[ ]"
--api-version " "

Azure Functions :-

Type	Trigger	Input	Output
Blob storage	✓	✓	✓
Cosmos db	✓	✓	✓
event Grid	✓		
event hubs	✓		✓
Http & Webhooks	✓		✓
Graph		✓	✓
Notification hubs			✓
Queue storage	✓		✓
Send Grid			✓
ServiceBus	✓		✓
Table storage		✓	✓
Timer	✓		
Twilio			✓

Cosmos db - table

```
CloudStorageAccount StorageAccount = - - ;
CloudTableClient tableClient = StorageAccount.CreateCloudTable
                                Client -->

CloudTable table = tableClient.GetTableReference(tableName);
```

Use TableEntity

Insert

```
TableOperation t = TableOperation.InsertOrMerge( ) ;
TableResult r = await Tab.ExecuteAsync(t);
```

Page

Blobs - metadata

```
CloudStorageAccount storageAccount = CloudStorageAccount.
                                     Parse(connString);

CloudBlobClient blobClient = StorageAccount.CreateCloud
                             BlobClient();

CloudBlobContainer container = blobClient.GetContainer
                               Reference("mycontainer");

container.CreateIfNotExist();
await container.FetchAttributesAsync();

Container.Metadata.Add("doctype", "");
Container.metadata["category"] = "guidance";
await Container.SetMetadataAsync();
```

a2 Storage

a2 Storage blob lease acquire
a2 Storage blob lease break
a2 storage blob lease release
renew
change

Immutable blobs: time base retention (WORM)
legal holds
write once
read many

AzureAD user :

New - AzureADUser -DisplayName " "
-PasswordProfile " "
-UserPrincipalName " "
- AccountEnabled True
-mail NickName " "

New -AzRoleAssignment -SignInName " "
- RoleDefinitionName "Reader"
- Scope $subScope

Get -AzRoleAssignment - SignInName " "
-Scope $subScope

Key Vault :-

az KeyVault Create --name " "
--resourcegroup " "

az KeyVault secret set --vault name " "
--name " "
--value " "

AppInsights:

Microsoft.ApplicationInsights.Web

LogicApp Custom Connectors :-

i) OpenAPI file
OpenAPI from URL
Import a postman collection

APIM :

New - AzAPImanagement --ResourceGroupName " "
- Location " "
- Name " "
- Organization " "
- AdminEmail " "
- SKU "Developer"

EventGrid

Sources		Event handlers
Blob		
Resource Groups		Functions
Azure Subscription		LogicApps
Eventhubs	→ eventGrid →	Eventhubs, storage queues
MediaService		
IOT hub		Hybrid connections, Webhooks, Azure automation
ServiceBus		
Azure maps		
CloudEvents sources		
Custom events		

Storage queue
~~Service Bus~~ :

classmate
Date
Page

CloudQueueClient queueClient = StorageAccount.CreateCloudQueueClient();

CloudQueue q = queueclient.GetQueueReference("myqueue");

q.CreateIfNotExists()

CloudQueueMessage msg = new CloudQueueMessage("text");

~~queue~~ q. addmessage(msg);

q. Peekmessage(); → Peek

msg. setmessageContent("updated contents");
q. updatemessage(msg, Timespan.FromSeconds(60.0), messageUpdateFields.Content);

q.Deletemessage(msg);

Date
Page

ServiceBus Queue:-

Microsoft.Azure.ServiceBus

TopicClient

SubscriptionClient

QueueClient

await topicClient.SendAsync(message);