Go to <https://portal.azure.com>

**Azure Resource Manager ARM:**

for beginner <https://www.youtube.com/watch?v=VQ_rixthPCI>

<https://www.youtube.com/watch?v=VQ_rixthPCI>

To download existing ARM go to Resource groups => your resource group => Automation script => Download => unzip

To clean downloaded ARM template <https://www.youtube.com/watch?v=myYTGsONrn0>

Use git while cleaning and validate before every commit. Reduce number of parameters instead use variables.

Create ARM project => Cloude => AzureResourceGroup => Blank Template

Add new item => DeploymentProject => Azure Resource Manager Deployment Template

Copy past or replace (only from **template.json to azuredeploy.json**) the downloaded ARM template. Right click the project => Validate => put/past default values in parameters dialog if required.

Note: do not replace power shell file.

If get SecurityError than sign the PowerShell script by going to folder => press shift => click open power shell window here => .\sign.ps1

Note: copy sign.ps1 file in the same folder. After every change sign again by .\sign.ps1

**Service Bus**:

<https://docs.microsoft.com/en-us/azure/service-bus-messaging/service-bus-dotnet-get-started-with-queues>

create a namespace: Create a resource => Integration => Service Bus => Create namespace

To create further rules with more constrained rights for regular senders and receivers See Service Bus authentication and authorization <https://docs.microsoft.com/en-us/azure/service-bus-messaging/service-bus-authentication-and-authorization>

Shared Access Signature (SAS): click the newly created namespace => click Shared access policies =>

click RootManageSharedAccessKey => click the copy button next to Primary Connection String

**Create a queue**: click the newly created namespace => click Queues => click + Queue => Enter the queue Name => click Create

Console application to send messages:

Add the Service Bus NuGet package “Microsoft.Azure.ServiceBus”

**Azure Key Vault**: Use Azure Key Vault to encrypt keys, tokens, certificates and small secrets like passwords. Applications have no direct access to keys. Monitor and audit your key use with Azure logging – pipe logs into Azure HDInsight or your security information and event management (SIEM).

<https://docs.microsoft.com/en-au/azure/key-vault/key-vault-get-started>

web app: <https://docs.microsoft.com/en-us/azure/key-vault/tutorial-net-create-vault-azure-web-app>

Azure CLI: open command

az login

az keyvault create --name "KeyVault1000" --resource-group "AzurePoc" --location "West US"

az keyvault secret set --vault-name "KeyVault1000" --name "AppKey" --value "password123"

to show/check: az keyvault secret show --name "AppKey" --vault-name "KeyVault1000"

Secret Identifier url: Go to KeyVault1000 => Secrets => AppKey => Current Version => Secret Identifier

Install these NuGet packages: “Microsoft.Azure.Services.AppAuthentication”, “Microsoft.Azure.KeyVault”

**Azure Storage Explorer**: with Azure Storage Explorer manage the contents of your storage account with Azure Storage Explorer. Upload, download, and manage blobs, files, queues, tables, and Cosmos DB entities.

**Secret Manager tool:** stores sensitive data for development work outside of your project tree. This approach helps prevent the accidental sharing of app secrets within source code.

<https://docs.microsoft.com/en-au/aspnet/core/security/app-secrets?view=aspnetcore-2.2&tabs=windows>

**Redis Caches**: <https://docs.microsoft.com/en-au/azure/azure-cache-for-redis/>

<https://docs.microsoft.com/en-us/azure/azure-cache-for-redis/cache-web-app-howto>

Videos: <https://www.youtube.com/results?search_query=azure+redis+cache>

ASP.NET Session State Provider for Azure Cache for Redis: <https://docs.microsoft.com/en-us/azure/azure-cache-for-redis/cache-aspnet-session-state-provider>

Azure Cache for Redis can cache both .NET objects and primitive data types (basic data types like int, string), but before a .NET object can be cached it must be serialized. Use JsonConvert.SerializeObject(obj) and JsonConvert.DeserializeObject<type>(obj).

to set or get a string value use StringSet or StringGet methods.

Locate your cache instance and your application in the same region.

Redis works best with smaller values.

Reuse connections - Creating new connections is expensive and increases latency, so reuse connections as much as possible.

Set an expiration value on your keys. See the Expire and ExpireAt commands for more details.

Install-Package StackExchange.Redis

**Redis Session State Provider**: Store ASP.NET session state in the cache.

<https://docs.microsoft.com/en-us/azure/azure-cache-for-redis/cache-aspnet-session-state-provider>

Azure Cache for Redis provides a session state provider that you can use to store your session state in-memory with Azure Cache for Redis instead of a SQL Server database.

Create Asp.Net Web App => Cloud => Asp.Net Web App (.Net Framework Standard) => MVC

Install-Package Microsoft.Web.RedisSessionStateProvider