**Azure CLI command**: is best tool for azure automation

Install Azure CLI on Windows: <https://docs.microsoft.com/en-us/cli/azure/install-azure-cli-windows?view=azure-cli-latest>

Install and log in to azure portal using az login.

Quick start: <https://www.youtube.com/watch?v=67brGZc7Slk>

**Azure resource manager**: The infrastructure for your application is typically made up of many components – maybe a virtual machine, storage account, and virtual network, or a web app, database, database server, and third-party services. You don't see these components as separate entities, instead you see them as related and interdependent parts of a single entity.

Using ARM you can repeatedly deploy your solution throughout the development lifecycle and have confidence your resources are deployed in a consistent state.

Also can manage your infrastructure through declarative templates rather than scripts.

You want to **deploy, manage, and monitor them as a group**. Azure Resource Manager enables you to work with the resources in your solution as a group. You can deploy, update, or delete all the resources for your solution in a single, coordinated operation. You use a template for deployment and that template can work for different environments such as **testing, staging, and production**. Resource Manager provides security, auditing, and tagging features to help you manage your resources after deployment.

**Resource provider**: Microsoft.Compute, which supplies the virtual machine resource, Microsoft.Storage, which supplies the storage account resource, and Microsoft.Web, which supplies resources related to web apps.

**Resource Manager template** - A JavaScript Object Notation (JSON) file that defines one or more resources to deploy to a resource group. It also defines the dependencies between the deployed resources. The template can be used to deploy the resources consistently and repeatedly. Consists of Parameters, variables, resources etc.

**Deploy**: we can deploy from VS by right click on project and deploy, using power shell etc.

Example: <https://www.youtube.com/watch?v=myYTGsONrn0> or <https://www.youtube.com/watch?v=VQ_rixthPCI>

To download existing/ already running app template

go **Azure portal** => on left panel click **Resource groups** => select the **Resource group** => select **Automation script** => download

Web Role – A web role is basically used to deploy a website, using languages supported by the IIS platform like, PHP, .NET etc. It is configured and customized to run web applications.

Worker Role – A worker role is more like an help to the Web role, it used to execute background processes unlike the Web Role which is used to deploy the website.

VM Role – The VM role is used by a user to schedule tasks and other windows services. This role can be used to customize the machines on which the web and worker role is running.

**Azure Service Fabric** is a distributed systems platform that makes it easy to package, deploy, and manage scalable and reliable micro-services.

Service Fabric: is alternative to Docker Swarm, DC/OS, Kubernetes and other containers technologies to develop scalable services.

Service Fabric enables you to build applications that consist of micro services. Stateless micro services (such as Web APIs, protocol gateways and web proxies) do not maintain a mutable state outside a request and its response from the service.

State-full micro services (such as Web Applications, user accounts, databases, devices, shopping carts, and queues) maintain a mutable, authoritative state beyond the request and its response.

difference between Service Bus Queues and Storage Queues?

**Azure Storage Queue** is simple and the developer experience is quite good. One sender on receiver. FIFO (first in first out). It uses the local Azure Storage Emulator and debugging is made quite easy. The tooling for Azure Storage Queues allows you to easily peek at the top 32 messages. messages can be strings, XML, Json etc.

**Azure Service Bus** is used for enterprise systems. Azure Service Bus Queues have the ability of deleting themselves after a configurable amount of idle time. This feature is very practical when you create Queues for each user, because if a user hasn’t interacted with a Queue for the past month, it automatically gets clean it up.

**Azure Redis Cache** is an open source **in-memory** data structure store, used as a database, cache and message broker.

**Azure Key Vault**:

IAAS example VM, PAAS (Web Apps, Mobile Apps in Azure) Platform as a Service, SAAS (Office 365, WebEx, Dropbox )

**Azure Resource Manager**: any single entity like storage account, a virtual machine, web server, database server, load balancer, virtual network is a resource and we can Group them to control, manage and access. We can also do role based access (for different groups of users) control.

Using PowerShell, we can deploy resource using scripts.

**Service Fabric**: is alternative to Docker Swarm, DC/OS, Kubernetes and other containers technologies to develop scalable services. Many existing Microsoft Azure cloud services run on top of Service fabric like document db, SQL server, power Bi, Skype for Business etc.

**Service Fabrics applications programing models**: there are there general approaches

1. Reliable services are of two types (easiest one like console app, )
   1. Stateless: console applications, **Web API** get fit here.
   2. State-full: with its own transections storage (can store information)
2. Reliable Actors: is build on top of State-full reliable services
3. Guest Executables: can run any application written in any language without any code change.

**Service Fabric Cluster** (use local cluster): we need to setup cluster at azure (Service Fabric cluster in Azure using the Azure portal) before publishing Service Fabric app.

The local Service Fabric cluster on a developer workstation version is just like the real cluster in cloud/production system as there is no difference in between both.

Service Fabric only runs on 64-bit OS and can use VS 2017 community Edition.

For development environment we need:

* Install Service Fabric tooling
* Install Service Fabric SDK (with includes Service Fabric Cluster)

Install certificates (Install-Certificates and Install-Certificates\_User) from O:\HEIMS\O Developer\

**Normal Application Vs Reliable Service (which is written using Service Fabric):**

Both are similar except it only runs on x64 bit OS, little learning curve. Designed to survive the services outage, while running on multiple machines.

**Service Fabric project**: Normally there is only on Service Fabric project in whole solution. It contains how to description how to deploy the application, what are the application parameters, and which services it consists of.

**Service Lifecycle**: both state-full and stateless service have identical sequence of events.

state-full services have slightly more complicated life.

**Azure Service Fabric Runtime**: it runs you exe, registers the service type and service is ready.

It creates service instance and then ask to create listeners (**ServiceInstanceListener**), after that open listener loop (time to listen incoming requests). After this RunAsync (CancellationToken ).

Before Service Fabric shutdowns the service it cancels the Cancellation Token passed in RunAsync. After this it closes all the listeners and finally destroy class instance. We should care about the Cancellation Token in case of state-full service.

**State API**: Azure Service Fabric has it’s state api for each service type which is isolated from each other. It’s much faster than talking to the external database etc. as it stores state on the local machine on the same disk. There is no need to open network connections, login to the databases, handshaking and encryption etc. Every time some data is written on it is replicated to few other multiple copies and no data is lost if one fails. So if primery copy gets fails, it makes another copy active.

The simplest way to store anything in a service is by using built-in **ReliableStateManager**. It’s like a normal dictionary collection.

All the operation in Service Fabric requires Transections just like database transections.

Quorum is set of one primary and few Replica/inactive copies of a service. Size of the Quorum is 3 nodes (I active/primary and 2 Replica).

Communication between the services we can use Service Remoting (is built-in and is default and preferred way), other options are WCF and Http (like web api). For Service Remoting we need to add nugget package Services.Remoting.

**Actor Model Support**: is a framework built on top of Reliable Services. It’s most popular. Actor = code (Reliable Services) + state (e.g. Reliable State) + mailbox + can send messages (e.g. using service remoting).

Message sent and received are like method calls. In mail box messages are queued and it can only process one message at a time and returns the results.

Multiple actors can run parallel, but one actor processes messages sequentially.

In VS add nugget package ServiceFabric.Actors.

Cluster = Nodes

Node = normal computer with OS and Service Fabric runtime. Each node can host multiple applications.

**SF Deployment**: Need MS Build & PowerShell

Another option is Azure CLI

MSBuild Path: C:\Windows\Microsoft.NET\Framework\v4.0.30319

Configure MSBuild path in System Properties => Environment Variables => Path => add

**Generate the Package**: Run power shell as admin and go to the solution folder and just type MsBuild

Managing State

Azure account?

StateManager

Azure workloads

developer command prompt => dotnet build / dotnet publish