**Chapter 1 : Develop Azure Computer Solutions(~~20-25~~25-30%)**

**Section 1: Implement IaaS solutions**

**Provisioning VM’s:**

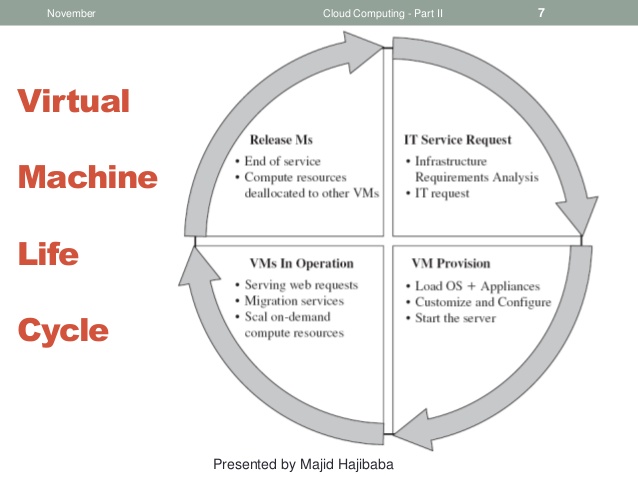
Server Provisioning is defining sever configuration based on the user requirements

Such as

* H/W [CPU, Storage, RAM, N/W, etc.]
* OS [Operating System]
* Application

**----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------**

**VM Provisioning Life Cycle:**



**IT Service Request:** Requirement Analysis

**VM Provision:** Load OS, Applications and Configuration

**VM In Operation:** Running & Providing Services

**Release VMs:** Deallocate Resources

**----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------**

**Configure VMs for remote access:**

**To Configure a VM for direct RDP Access:**

Run an **RDP** server or use built-in options to **enable Remote Desktop** for all users on the **System.** Example: Enabling remote connections in Windows 7 and Windows 2008 R2. Click Start, and then right-click **Computer**.

Verify that port 3389 is open on the **VM** operating **system** firewall.

**Enable Remote Management:**

In Server Manager, Click Local Server-> **Remote** management current setting (disabled).

Select **Enable Remote** management for this server.

Click OK.

**Prepare Your Virtual Machines for Remote Desktop:**

* Install **Remote Desktop Services Components** on physical servers or on virtual machines.
* Create **Windows Sever Virtual Machine in Azure**
* Create **three VMs** like following:
  + RD Session Host
  + Connection Broker
  + RD Web and RD Gateway
* Ensure the availability of your RDS deployment, create an availability set (under **High availability** in the VM Creation Process**)** and group multiple VMs in the availability Set.
* Connect to the virtual machine using the Remote Desktop Connection (RDC) client:
* In the Azure portal open the Resource groups view, and then click the resource group to use for the deployment.
* Select a new RDSH virtual machine (for example, Contoso-Sh1)
* Click **Connect -> Open** to open the Remote Desktop Client.
* In the client, click **Connect**, and then click **Use another user account.** Enter the user name and password for the local administrator account.
* Click **Yes** When warned about the certificate.

**Enable Remote Management:**

* In Server Manager, Click **Local Server -> Remote Management Current Setting (Disabled)**
* Select **Enable remote management for this server.**
* Click **OK.**

**Optional:**

You can temporarily set Windows Update to not automatically download and install updates. This helps prevent changes and System restarts while you deploy the RDSH Server.

* In Server Manager, Click **Local Server-> Workgroup Current Setting.**
* Click **Change -> Domain,** and then enter the domain name (for example, Contoso.com)
* Enter the domain administrator credentials
* Restart the Virtual Machine.

1. Repeat Steps 1 Through 4 for the **RD Web and GW Virtual Machine**
2. Repeat Steps 1 Through 4 for the **RD Connection Broker Virtual Machine.**
3. Initialize and format the attached disk on the RD Connection Broker Virtual Machine
   1. Connect to the RD Connection Broker Virtual Machine (Step 1 above)
   2. In Server Manager, click **Tools**-> **Computer Management**
   3. Click **Disk Management**
   4. Select the attached disk, then **MBR** (**Mater Boot Record),** and then click **OK.**
   5. In the **New Sample Volume** wizard, accept the default values but provide a applicable name for the **Volume Label** (like Shares)
4. On the RD Connection Broker virtual machine create file shares for the user profile disks and certificates:
   1. Open File Explorer, click **This PC,** and open the disk that you added for file shares.
   2. Click **Home** and **New Folder.**
   3. Enter a name for the user disks folder, for example, **UserDisks.**
   4. Right-click the new folder and the **Properties-> Sharing-> Advanced Sharing**
   5. Select **Share this folder** and click **Permissions**
   6. Select **Everyone,** and then click **Remove.** Now click **Add,** enter **Domain Admins,** and click **OK**.
   7. Select **Allow Full Control,** and then click **OK-> OK-> Close**
   8. Repeat Steps **c to g** to create shared folder for certificates.

**----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------**

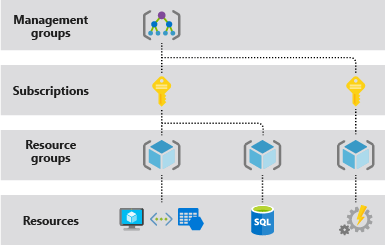
**Create ARM Templates:**

Azure Resource Manager Templates provide simple and unified way to create and manage your Azure environments for both developers and administrators. With simple JSON files you can deploy your environment in consistent fashion and at great speed.

**What is mean by ARM?**

**ARM –** Azure Resource Manager





It is a Centralized Resource Management System. It Provides the Following Key Concepts. There are,

* Multiple Deployment Interfaces
* Centralized Layer
* Secured with Azure AD
* Communicates with Resource Providers
* Hierarchical Scope Level
* Unified Language for Resource Management
* Declarative Speech
* Simple Files
* JSON Format
* Standard Properties (Schema, Content Version, parameters, Variables, Resources, Outputs, Functions)
* Multiple Resources Support
* Parameterization of our Templates
* Outputs
* Templated Resource
* The JSON Property “Type” : It Describes what service we are deploying, It is a mandatory field.
* Resource Provider Versions (API Versions – JSON Property)
* Multiple Ways of Deployment

**Key Benefits of ARM Templates:**

* **Infrastructure as Code** (IaC), **Policy** and **Roles** as Code
* **Declarative** Syntax
* **Repeatable** results
* Orchestration
* Built-in Validation
* Modular Files
* Tracked Templates
* Many **Authoring Tools**

**Additional Features of ARM Templates:**

* Functions and Expressions
* Linked and nested templates
* Dependencies
* References
* Export
* Loops
* Conditions

**When to Use ARM Templates:**

* Application Development and Maintenance
* CI/CD scenarios
* Azure governance (Policies, Roles, etc..)

**Tools:** ARM Template Viewer

**----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------**

**Create Container Images for Solutions by using Docker:**

**What is Docker?**

**Developer Says:** It works on my machine

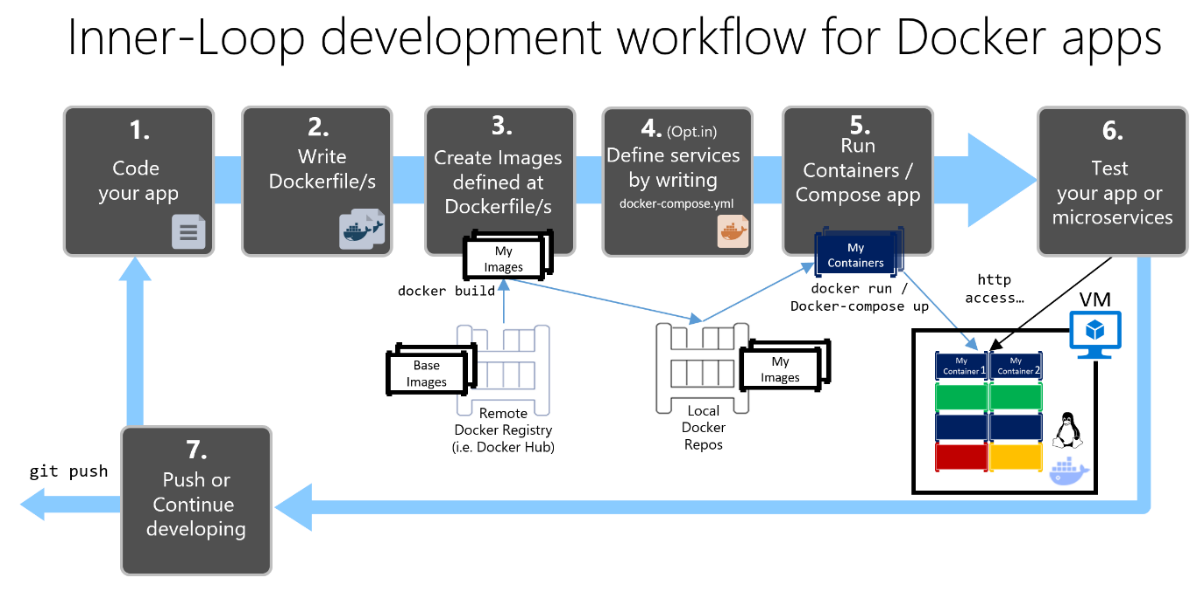
**Customer/Production Said:** It’s not working for us.

To solve above Debt, “Docker” was Introduced. The Dockers allows us to create containers.

This is called as “**Docker Container”.** The main features of docker is **“Portability”**

The Docker is categorized into 3 main things:

* Client-Side Application Program
* It can also be used as a “Service”, if we deployed it to the server.
* Social Networking Platform



**Docker Container:**

**Containers:** A **Container** is a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another.

**Benefits of Container:**

* Less Development Overhead
* Less Resources Consumed
* Smaller Size
* Faster Startup
* Reduced and simplified updates

**Docker Container:**

**Docker Container** is an isolated, secured shipping box, which is produced or created when the docker image is run. Docker containers can be run, started, stopped, deleted.

* All of our codes, application information was packed as a container here.
* The **Docker Container** is an Instances of the **Docker Image.**
* We can create **N number of Docker Containers** from the same **Docker Image.**
* It is the blueprint or place where the original application will runs.

It Consists of the following,

* Code Dependencies
* Configuration
* Process
* Networking Information
* O.S (Some Chunks) and More

**Docker Image:**

It is set of Read Only Files. Once a Docker Image is Built, it cannot be modified. Docker Images are used to build the docker containers.

When we run the **Docker Image** with **Docker Run** Command, it produces the output as **Docker Container.**

**Docker Image** is a read only template, composed of layered file system, needed to build a running docker container, basically the running instance of the image.

**Diff Between Docker Image Vs Docker Container:**

Docker Image is a Class

Docker Container is an Object of the Docker Image Class

**----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------**

**Publish an Image to the Azure Container Registry:**

**Before Ensure:**

* Azure CLI should be Installed on the System
* We should have a Azure Resource Group
* Open the PowerShell then execute the below commands

**Command:**

* docker images
* docker pull ImageName [Once Pull Completed]
* docker login RepositoryName [Once Login Succeeded]
* docker push RespositoryName

**Example:**

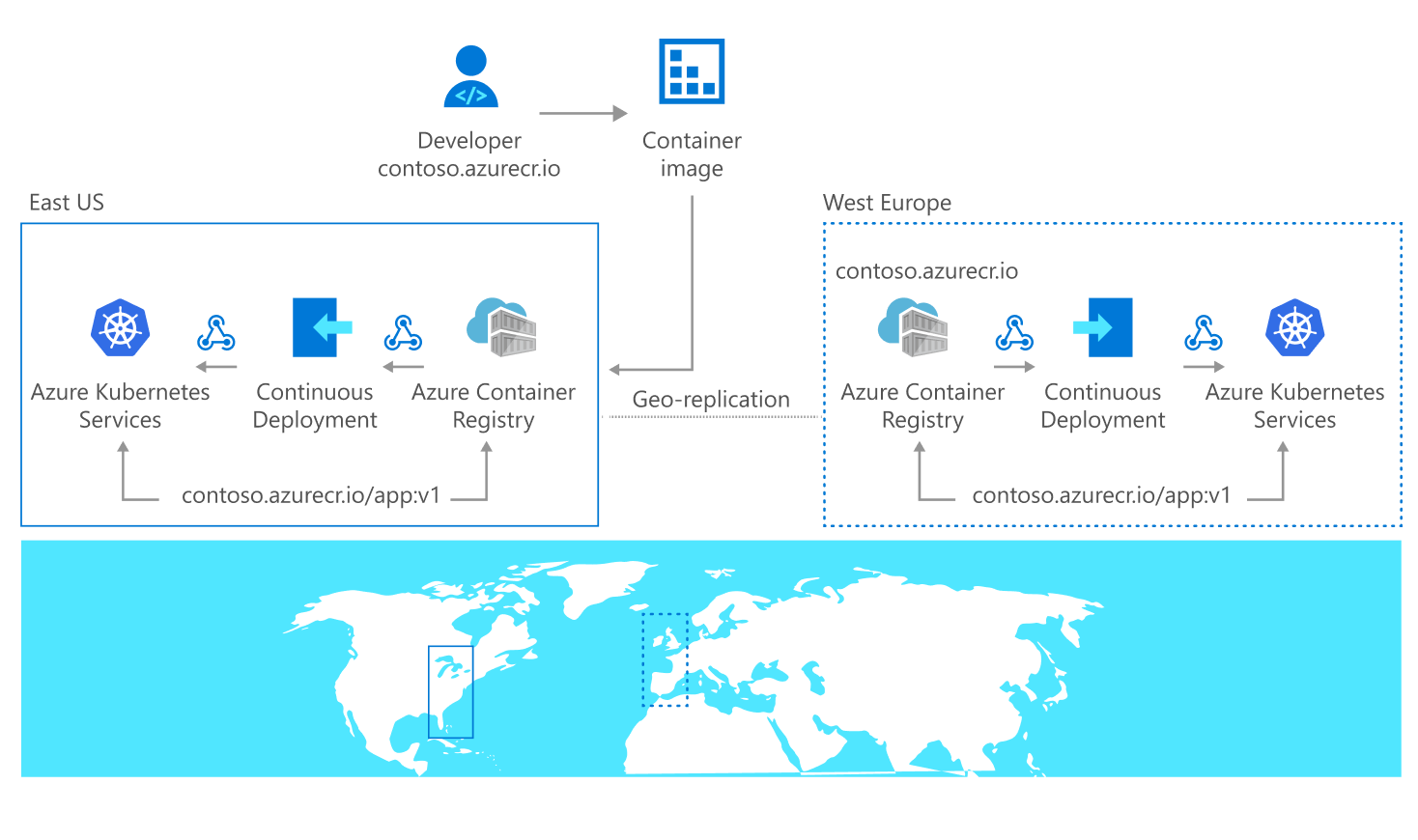
docker push testmyacr.azureecr.io/myaspnet

Here, RepositoryName is the docker container as we created from the docker image.

“testmyacr.azureecr.io/myaspnet”

**What is mean by Azure Container Registry?**

**Azure Container Registry** allows you to build, store, and manage **container** images and artifacts in a private **registry** for all types of **container** deployments. Use **Azure Container Registries** with your existing **container** development and deployment pipelines.



**----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------**

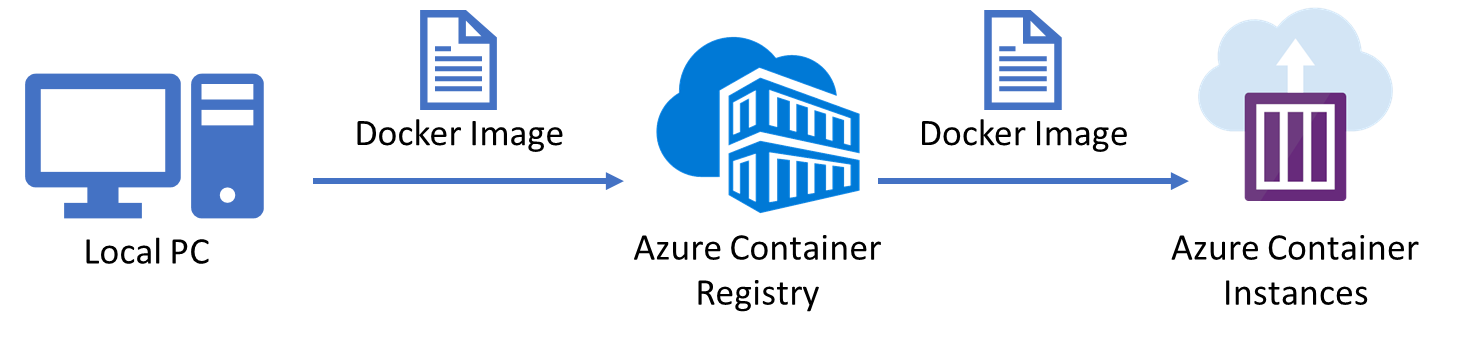
**Run Containers by using Azure Container Instance:**

**What is mean by Azure Container Instance?**

**Azure Container Instances** is a service that enables a developer to deploy **containers** on the **Microsoft Azure** public cloud without having to provision or manage any underlying infrastructure.

**Azure Container Instance** is a solution for any scenario that can operate in isolated **containers**, without orchestration. Run event-driven applications, quickly deploy from your **container** development pipelines, and run data processing and build jobs.

**Azure Container Instance** offers the fastest and simplest way to run a container in Azure, without having to provision any virtual machines and without having to adopt a higher-level service.

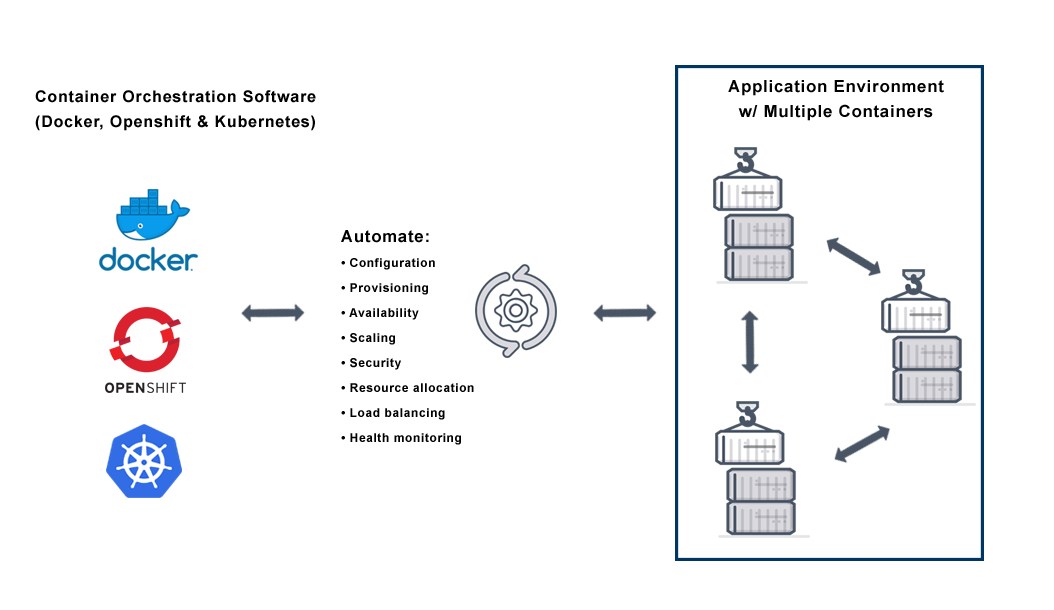


**Key Benefits:**

* Fast startup times – in just few seconds ACI can start up containers
* Public IP and DNS name
* **Custom Sizes:** Specify exact values for CPU cores and memory.
* We can host the containers in both Linux and Windows
  + **Linux And Windows:** Schedule both Windows and Linux containers using the same API.
* **Persistent Storage:** Mount Azure Files Shares directly to a container to retrieve and persist state.
* Virtual Network Deployment
* **Per Second Billing :** Incur costs only while the container is running.
* **Hypervisor-level-security:** Isolate your application as completely as it would be in a VM.

**What is mean by Orchestration in Azure?**

The task of automating and managing a large number of containers and how they interact is known as **orchestration**. Popular container orchestrators include Kubernetes, DC/OS, and Docker Swarm. **Azure** Container Instances provide some of the basic scheduling capabilities of **orchestration** platforms.



**Running Steps for Run Containers by using Azure Container Instance:**

1. Sign into the **Azure portal [azure.portal.com]** with your Azure Subscription.
2. Open the **Azure Cloud Shell** from the Azure portal using the Cloud Shell icon.



1. Create a new resource group with the name **learn-deploy-aci-rg** so that it will be easier to clean up these resources when you are finished with the module. If you choose a different resource group name, remember it for the rest of the exercises in this module. You also need to choose a region in which you want to create the resource group, for example **East US.**

**Azure CLI:**

az group create --name learn-deploy-aci-rg –location eastus

You create a container by providing a name, a Docker Image, and an Azure Resource Group to the **az container create** command. You can optionally expose the container to the Internet by specifying the DNS label. In this example, you deploy a container that host a small web app. You can also select the location to place the image – you will use the **East US** region, but you can change it to a location close to you.

1. You provide a DNS name to expose your container to the Internet. Your DNS name must be unique. For learning purposes, run this command from Cloud Shell to create a Bash variable that holds a unique name.

**Azure CLI:** DNS\_NAME\_LABEL=aci-demo-$RANDOM

1. Run the following az container create command to start a container instance.

**Azure CLI:**

az container create \

--resource-group learn-deploy-aci-rg \

--name mycontainer \

--image microsoft/aci-helloworld \

--ports 80 \

--dns-name-label $DNS\_NAME\_LABEL \

--location eastus

$DNS\_NAME\_LABEL specifies your DNS name. The image name, **microsoft/aci-helloworld**, refers to a Docker image hosted on Docker Hub that runs a basic Node.js web application.

1. When the az container create command completes, run az container show to check its status.

**Azure CLI:**

az container show \

--resource-group learn-deploy-aci-rg \

--name mycontainer \

--query "{FQDN:ipAddress.fqdn,ProvisioningState:provisioningState}" \

--out table

You see your container's fully qualified domain name (FQDN) and its provisioning state. Here's an example.

FQDN ProvisioningState

-------------------------------------- -------------------

aci-demo.eastus.azurecontainer.io Succeeded

If your container is in the **Creating** state, wait a few moments and run the command again until you see the **Succeeded** state.

1. From a browser, navigate to your container's FQDN to see it running. You see this

