AZ-203 Developer Training

Azure app services and Azure Functions

Agenda

- What is Azure Apps Service
- What is Azure
- Mobile App
- What is Azure Logic Apps
- Deploying WebApps
- Comparison of VM vs WebApp vs Cloud
- Services FTP Transfer
- App Service Web App for Containers WebJobs
- Diagnostics Logging
- Azure Functions
- Containers for Azure Functions

Azure Apps Service

What Is Azure Apps Service?

- App Service provides a comprehensive platform for building cloud-based applications.
- App Service provides a hosted service with which developers can build mobile and web apps.
- The Web Apps feature is a platform of technologies that enable you to build web apps in Azure without having to deploy, configure, and maintain your own Azure VMs.
- You can build web apps by using the ASP.NET, PHP, Node.js, Java, and Python frameworks and a range of programming languages, such as C#, HTML5, PHP, Java, Node.js, and Python.
- They also integrate with common development environments such as Visual Studio and GitHub.
- With App Service, you pay for the Azure compute resources you use.

Azure Web Apps: Features

Visual Studio integration

- Dedicated tools in Visual Studio streamline the work of creating, deploying, and debugging.

API and mobile features

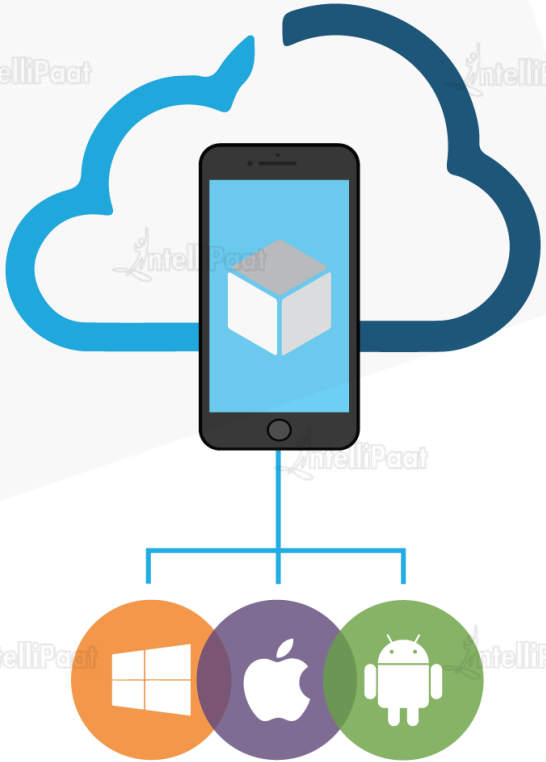
- Web Apps provides turn-key CORS support for RESTful API scenarios and simplifies mobile app scenarios by enabling authentication, offline data sync, push notifications, and more.

Serverless code

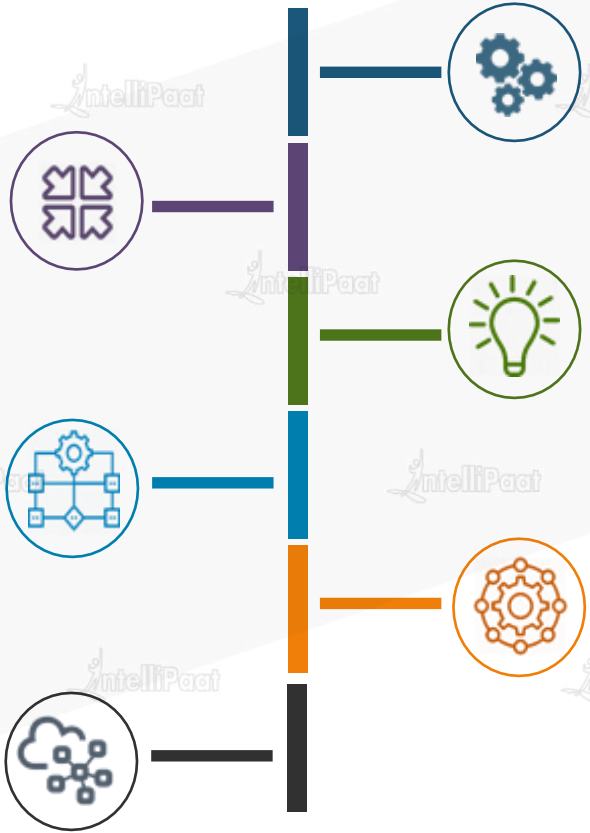
- Run a code snippet or script on-demand without having to explicitly provision or manage infrastructure, and pay only for the compute time your code actually uses.

What Is Azure Mobile App?

- The Mobile Apps feature is a part of App Service.
- It provides a platform for building and hosting backend services for mobile applications.
- The Mobile Apps feature allows developers to build cross-platform apps that can run on Windows, iOS, or Android.
- These apps can operate exclusively in the cloud or connect with your on-premises infrastructure for authentication and authorization purposes.
- They can benefit from the built-in push notification engine that can send personalized push notifications to almost any mobile device that is using iOS, Android, or Windows.



What Is Azure Logic Apps?



1. Logic apps automate business processes by linking together cloud-based apps, such as Office 365, Google Services, and Salesforce.

2. With the Logic Apps feature, you can use a visual designer to combine connectors available from Azure Marketplace for different integration scenarios.

3. Logic Apps uses a workflow engine to implement business processes that you designed and relies on connectors to provide user access.

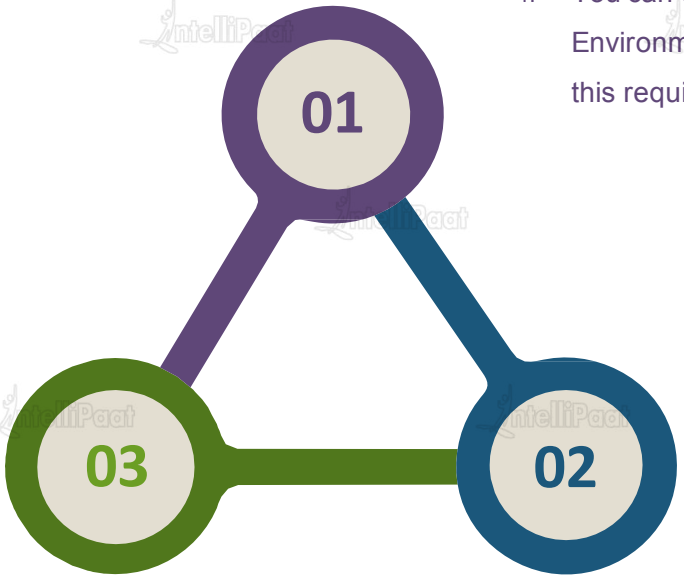
4. Connectors also facilitate initiating new workflow instances via triggers based on events that you define.

5. Each step in the workflow is an action that accesses data or services through a connector.

6. More advanced integration scenarios can use rules, transformations, validations, and features that are part of BizTalk Services.

Azure App Service Environment

1. You can use the App Service Environment to accommodate this requirement.



2. Business-critical apps often require highly scalable, isolated, and dedicated environments.

3. You can use the App Service Environment to host web apps, mobile apps, and API apps that require highly scalable compute resources, isolation, and direct virtual network connectivity.

Azure Web Apps:
Deploying Web Apps

- You can deploy your web apps by using several methods:

Copying files manually by using FTP	Synchronizing files and folders to App Service from a cloud storage service, such as OneDrive or Dropbox.

- App Service also supports deployments by using the Web Deploy technology.
- This approach is available with Visual Studio, WebMatrix, and Visual Studio Team Services.
- If you want to perform deployments by using Git or FTP, you must configure deployment credentials.
- Knowledge of deployment credentials will allow you to upload the web app's code and content to the new web app to make it available for browsing.

Azure Web Apps: Comparison

To host a web application in Azure, you can use:

Azure VMs, Web Apps, or Azure Cloud Services.

Azure VMs

- Azure VM can host any web server, such as IIS or Apache.
- You can host supporting servers, such as SQL Server instances that host databases, on other VMs.
- You can use Azure VM load balancing or Azure VM Scale Sets to scale the web application horizontally.
- If you choose to host a web application on Azure VMs, you have maximum control over their OS and supporting software components.

Web Apps

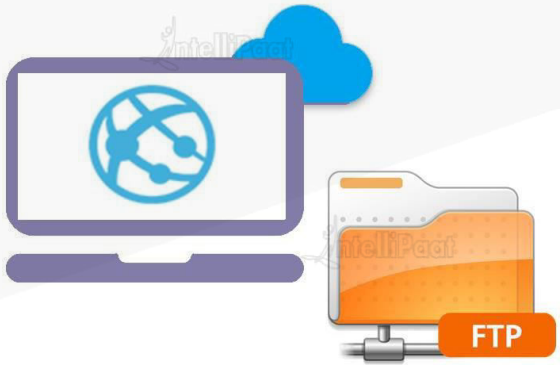
- Alternatively, you can choose to host your web application using the Web Apps feature.
- You can upload a custom web application code into it or deploy any code from the Azure.
- You can also scale web apps horizontally by changing the number of instances and relying on Azure built-in load balancing to distribute the traffic across them.
- You also cannot establish an RDP or SSH connection to the virtual machine hosting a web app.

Cloud Services

- You also can choose to build a web application as a cloud service.
- A cloud service consists of a web role, which serves as the front-end of an application, and a worker role, which is responsible for running background tasks.
- You can scale each role independently by specifying the number of role instances, which gives you more control over scalability compared to web apps.
- Since role instances run Windows Server, you can connect to them using RDP.
- PaaS cloud services are unique to Azure.
- Existing web applications might require significant modification before they can run as a cloud service.

Azure Web Apps: FTP Transfer

- To deploy a web app by using FTP, you must configure your client with the destination URL of the remote FTP server and the credentials that FTP can use to authenticate.
- In addition, you must choose either the active or the passive FTP mode.
- These are the Azure web app deployment credentials.



Hands-on

Hands-on

- Configure and deploy the code via Web App
- Configure and deploy the code via the cloud storage service

WebJobs

WebJobs: Overview

WebJobs is a feature of Azure App Service that enables you to run a program or script in the same context as a web app, API app, or mobile app. There is no additional cost to use WebJobs.

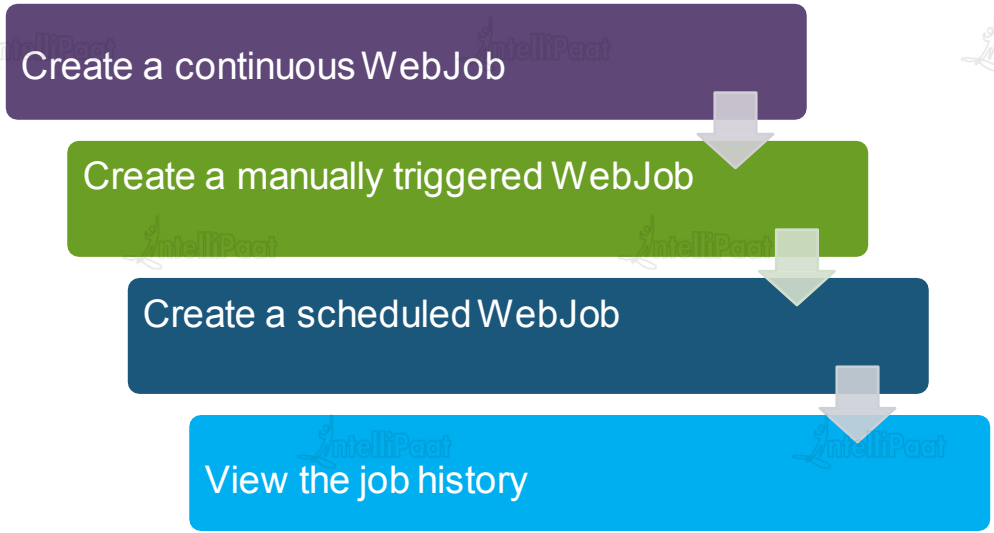
WebJob Types

Continuous	Triggered
Starts immediately when the WebJob is created. To keep the job from ending, the program or script typically does its work inside an endless loop. If the job ends, you can restart it.	Starts only when triggered manually or on a schedule.
Runs on all instances that the web app runs on. You can optionally restrict the WebJob to a single instance.	Runs on a single instance that Azure selects for load balancing.
Supports remote debugging.	Doesn't support remote debugging.

Note: WebJobs is not yet supported for App Service on Linux.

Run Background Tasks in Azure App Service (WebJobs)

Steps to Follow



Hands-on

Hands-on

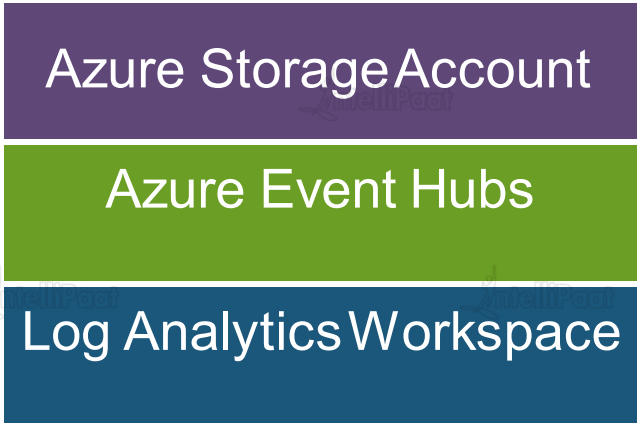
- Develop and deploy WebJobs using Visual Studio Azure App Service Configure environment variables to run the container
  - Create and publish WebJobs as .NET Core console apps
  - Publish a .NET Core WebJob to App Service from Visual Studio
  - Enable WebJobs deployment for an existing Console Application project
  - Schedule a triggered WebJob (using CRON expression)

You may take a reference from the official Microsoft document:
https://docs.microsoft.com/en-us/azure/app-service/webjobs-dotnet-deploy-vs

Diagnostics Logging

Diagnostics Logging: Overview

With Azure diagnostic logs, you can view core analytics and save them into one or more destinations including:



This feature is available on CDN endpoints for all pricing tiers.

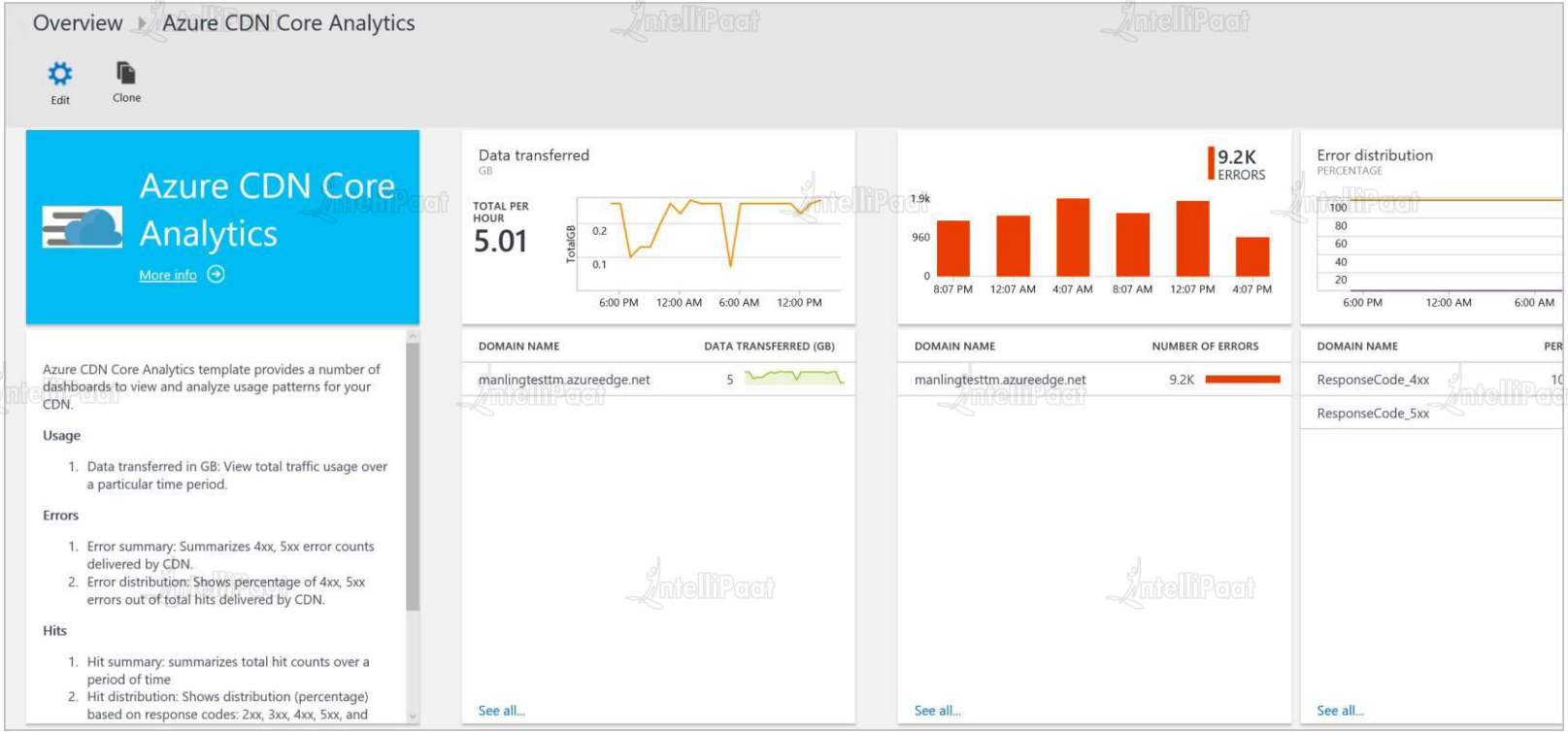
How Does It Help?

Azure diagnostics logs allow you to export basic usage metrics from your CDN endpoint to a variety of sources so that you can consume them in a customized way. For example, you can do the following types of data export:

- Export data to blob storage, export to CSV, and generate graphs in Excel
- Export data to Event Hubs and correlate with data from other Azure services
- Export data to Azure Monitor logs and view data in your own Log Analytics workspace

CDN Core Analytics View

The following diagram shows a typical CDN core analytics view of data.



Enabling Diagnostics Logging for Apps in Azure App Service

The first step is to do the application diagnostics.

Application diagnostics allows you to capture information produced by a web application. ASP.NET applications can use the System.Diagnostics.Trace class to log information to the application diagnostics log. For example:

```
System.Diagnostics.Trace.TraceError("If you're seeing this, something bad happened");
```

At runtime, you can retrieve these logs to help with troubleshooting.

App Service also logs deployment information when you publish content to an app. It happens automatically, and there are no configuration settings for deployment logging. Deployment logging allows you to determine why a deployment failed. For example, if you use a custom deployment script, you might use deployment logging to determine why the script is failing.

Enabling Diagnostics Logging for Apps in Azure App Service

- Steps to follow to enable the diagnostics logs
- Click on Settings > Diagnostics
- Choose the Level

For Application logging, you can turn on the file system option temporarily for debugging purposes. This option turns off automatically in 12 hours. You can also turn on the blob storage option to select a blob container to write logs to.

Next step is to **Download the Logs**

Diagnostic information stored to the app file system can be accessed directly using FTP. It can also be downloaded as a Zip archive using Azure CLI.

Enabling Diagnostics Logging for Apps in Azure App Service

- Download with **Azure CLI**
- **To download the log files using the Azure Command**
  - Line Interface
  - Open a new command prompt
  - PowerShell, Bash, or Terminal session
- **After completing the above steps, use the below command:**

```
az webapp log download --resource-group resourcegroupname --name appname
```

- This command saves the logs for the app named ***appname*** to a file named diagnostics.zip in the current directory.

Enabling Diagnostics Logging for Apps in Azure App Service

Now, view logs in Application Insights by adding the app **SDK** in **VS** and then add the **Trace Listener package** to your project. Next, **upload** and **generate the log data**.

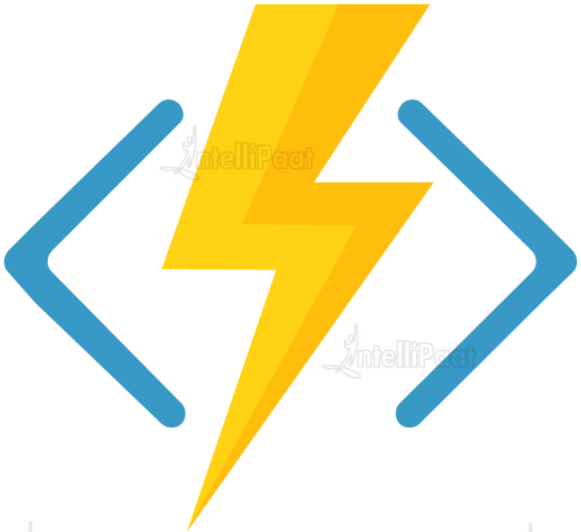
Next step is to **Stream logs** with Azure CLI and **check** all your logs.

Azure Functions

Functions as a Service(FaaS)

A platform for running “functions”, which are simply your code running in response to an event

Azure Functions



- Speed up your development
- Reduce your costs
- Scale automatically

Supported Languages

Supported Languages

Triggers

Triggers

Timer: run a function on a schedule

Message: listen for messages on a queue

HTTP Request: implement web APIs or webhooks

Many more: Azure Storage blob creation, Cosmos DB new row

Bindings

Bindings

- Eliminate repetitive boilerplate code
- Focus on core business logic

Serverless Architecture

Serverless Architecture

- Servers are managed for you
- Servers are abstracted away
- Per-second billing model
- Only pay when your code runs
- Monthly free grant
- Automatic scale
- Multiple servers meetdemand

Simpler

Cheaper

More Scalable

Hosting Models

Hosting Models

Consumption Plan (Serverless)	App Service Plan	Docker Container
Per-second billing	Reserved servers	Run anywhere
1,000,000 executions	Predictable monthly cost	On premises
400,000 GBs		Other cloud providers

Development Environments

Development Environments

Azure Portal	Visual Studio 2017	Azure Functions Core Tools
https://portal.azure.com	Powerful IDE	Cross platform
Experiments	Azure Functions extensions	Visual Studio Code
Proof of concept	Debug and test locally	Azure Functions extensions

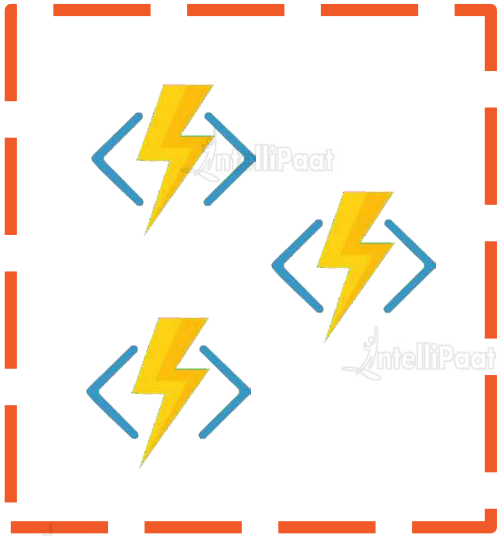
Additional Features

Additional Features

Security	Durable Functions	Proxies
API keys	Define workflows	Route incoming requests
Identity provider integration	Run tasks in parallel	Static website
	Retries and error handling	Transform requests and responses

Function Apps

Function Apps



- Unit of deployment
- Share common configuration Scale together
- Logically related

Hands-On

Hands-On

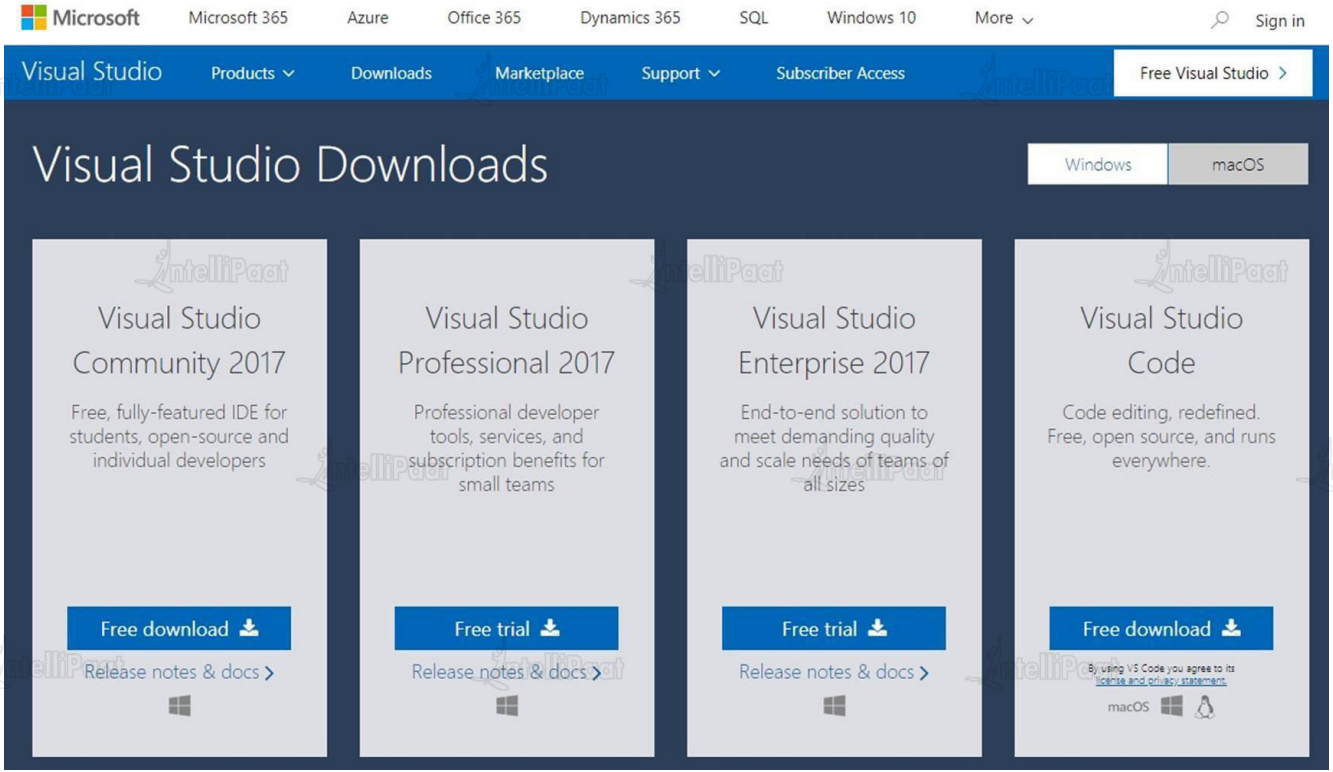
- Create an Azure Function App in the Azure portal
- Create a function in the Azure portal
- Test a function in the Azure portal
- View function invocation history
- Understand function.json

Create Azure Functions in Visual Studio

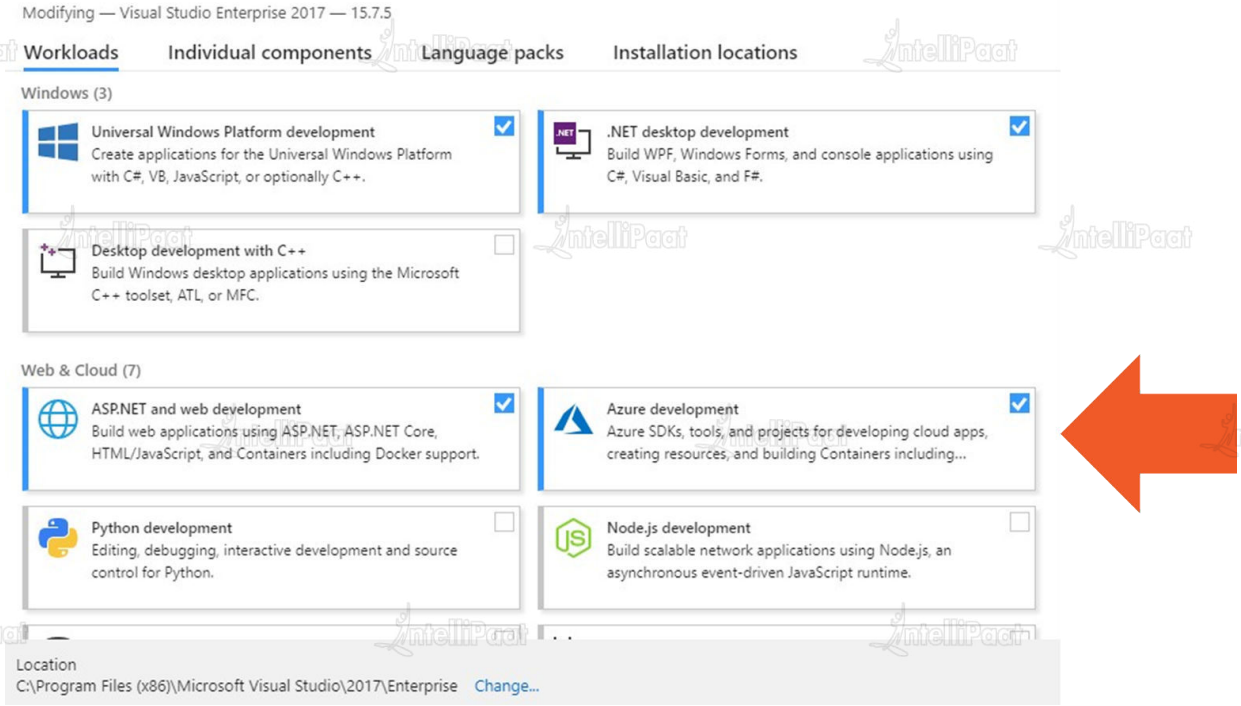
Visual Studio 2017

- Supports creating Azure Function apps
  - C#, JavaScript, F#
- IntelliSense Debugging

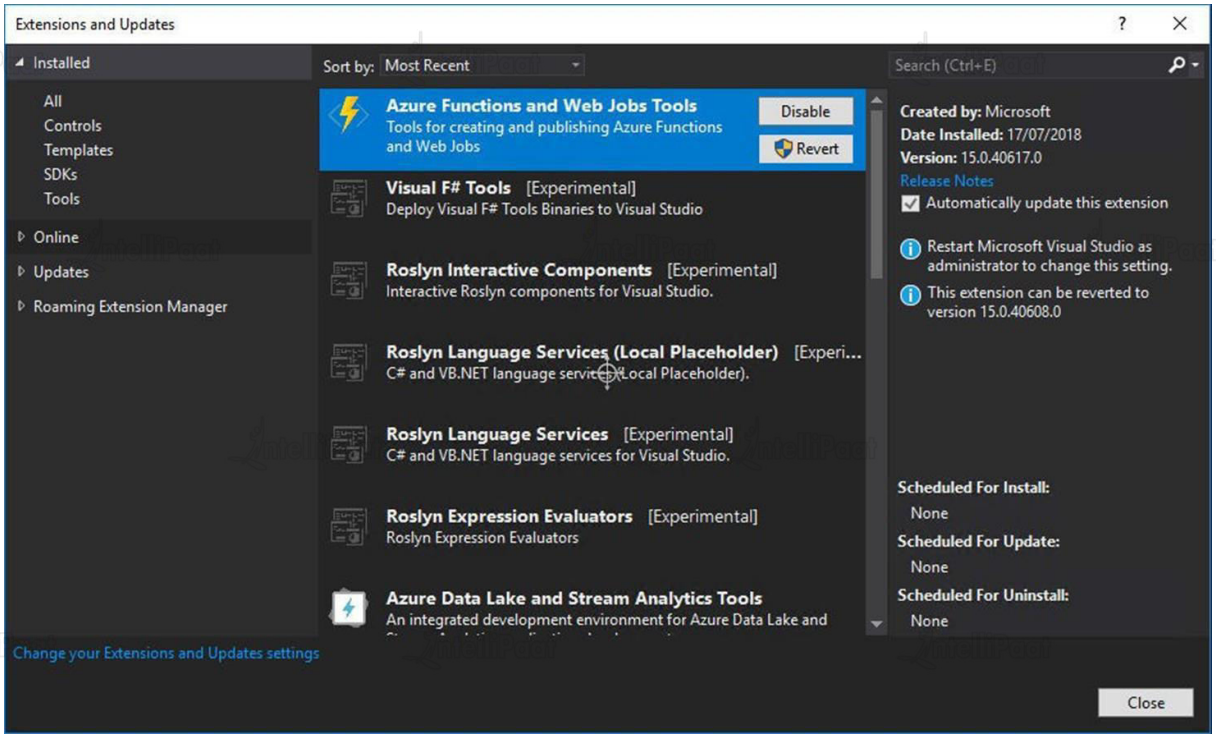
Visual Studio 2017



Visual Studio 2017



Visual Studio 2017



Hands-On

- Create an Azure FunctionApp in Visual Studio 2017
- Add a new function to a FunctionApp
- Test a function in Visual Studio

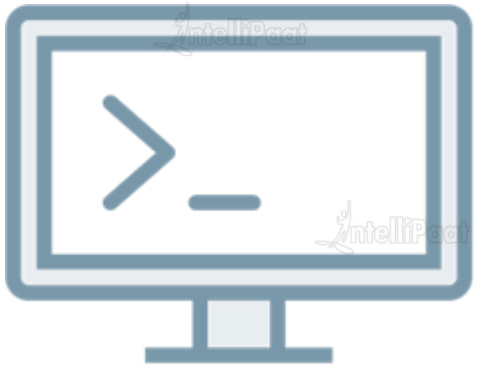
Create Azure Functions in Visual Code

Visual Studio Code

- Visual Studio 2017 is Windows only
- Visual Studio Codeis cross-platform
- Many extensions
- Azure support

Azure Functions Core Tools

Azure Functions Core Tools



- Command line tools
- Creating new FunctionApp
- Adding functions
- Deploying to Azure
- Includes the Azure Functions runtime
  Azure Storage Emulator
- Windows only
- Use a real Storage Account instead

Hands-On

- Create an Azure Function App with the Azure Functions Core Tools
- Work with Azure Functions in Visual Studio Code

Building a REST API in Azure Functions

Default HTTP Triggered Function Routes

http://localhost:7071/api/MyFunctionName

https://myfunctionapp.azurewebsites.net/api/MyFunctionName

How can we customize the route?

Demo Scenario

Demo Scenario

REST API for a Todo List Application

Operation	Default Route	REST Route
Get all Todo items	api/GetAllTodos	GET api/todo
Get Todo item by id	api/GetTodoById	GET api/todo/{id}
Create new Todo item	api/CreateTodo	POST api/todo
Update a Todo item	api/UpdateTodo	PUT api/todo/{id}
Delete a Todo item	api/DeleteTodo	DELETE api/todo/{id}

Hands-On

- Get a Todo item by id
  - GET /api/todo/{id}
- Delete a Todo item
  - DELETE /api/todo/{id}
- Test the API with Postman
- Calling the API from a web page
  - Configure CORS

Working with Triggers and Bindings

Triggers

HTTP Request

Timer

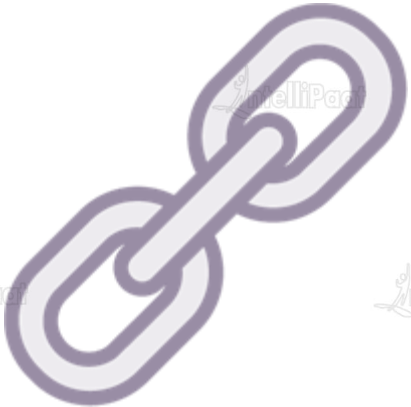
Blob

Queue

Cosmos DB

Service Bus

Bindings



- Input bindings
- e.g. Blob, Cosmos DB
- Output bindings
- e.g. Queue, SendGrid
- Defined in function.json Several built-in bindings
- More available as extensions

Choosing a Database

Choosing a Database

Note: Azure Functions can connect to any database

Cosmos DB

- Highly scalable
- Globally distributed
- Document database
- Multiple programming models
- Not as cheap as other options
- Maybe overkill for this API

Table Storage

- Extremely cheap
- Limited querying capabilities
- Emulated by the Storage Emulator
- Every Function App already has a Storage Account in Azure
- Good choice for initial prototype
- Can migrate to Cosmos DB later

Every Table Storage row has a *PartitionKey* and a *RowKey*

Hands-On

- Use the Table Storage bindings
- Testing TableStorage
  - Azure Storage Explorer
- Timer triggered function
- Queue output binding
- Queue trigger
- Blob input binding

Running Functions in Azure

Deploying Function Apps

Manual Deploy	Git	Zip
Visual Studio or VS Code	Automated deployment	Kudu API
Experiments	Configure in portal	Azure Functions Core Tools
Publish to an existing Function App	GitHub or VSTS	Azure CLI

Azure Functions Proxies

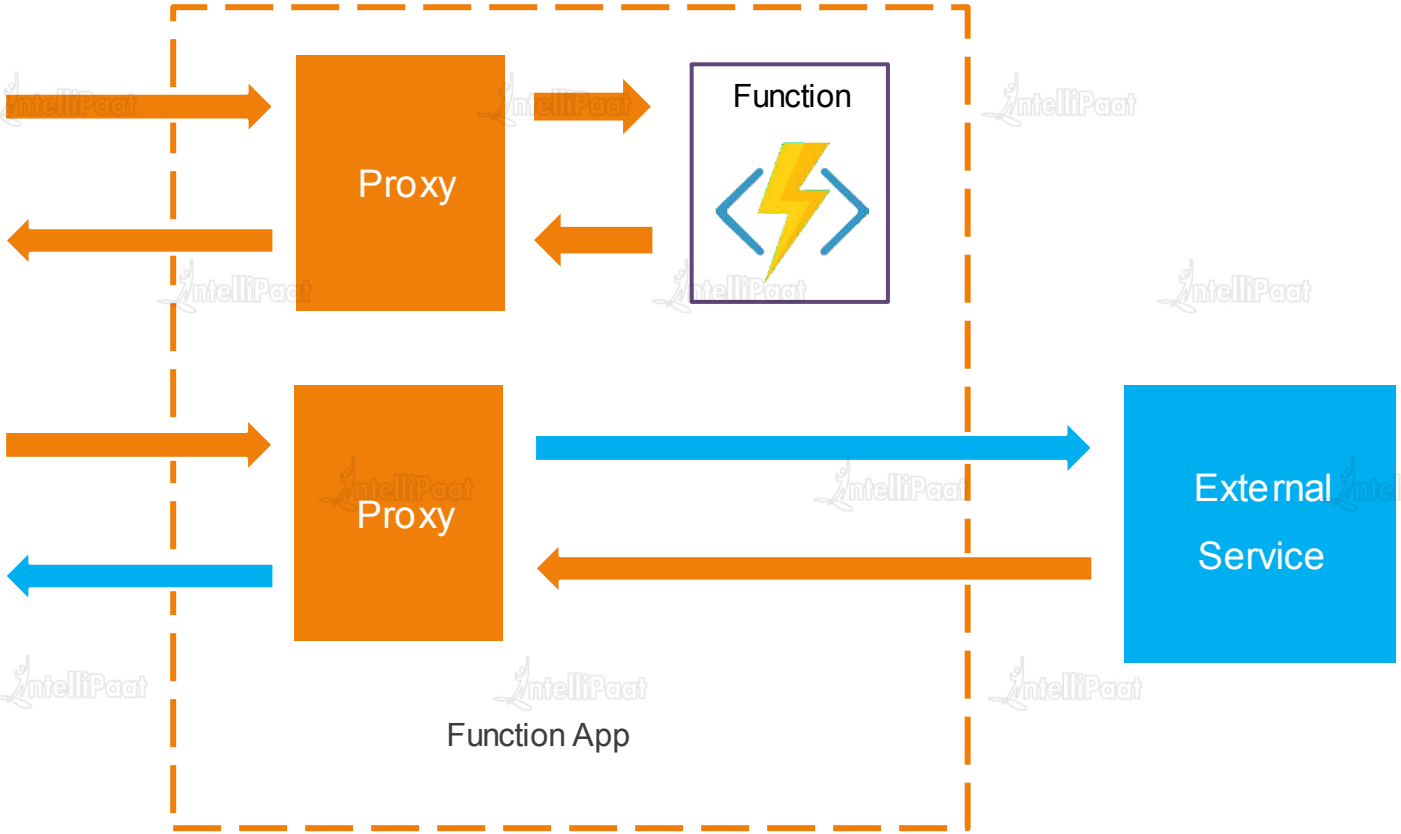
Publishing a Zip with Azure CLI


```
az functionapp deployment source config -g MyResGroup -n MyFuncApp –src MyZip.zip
```

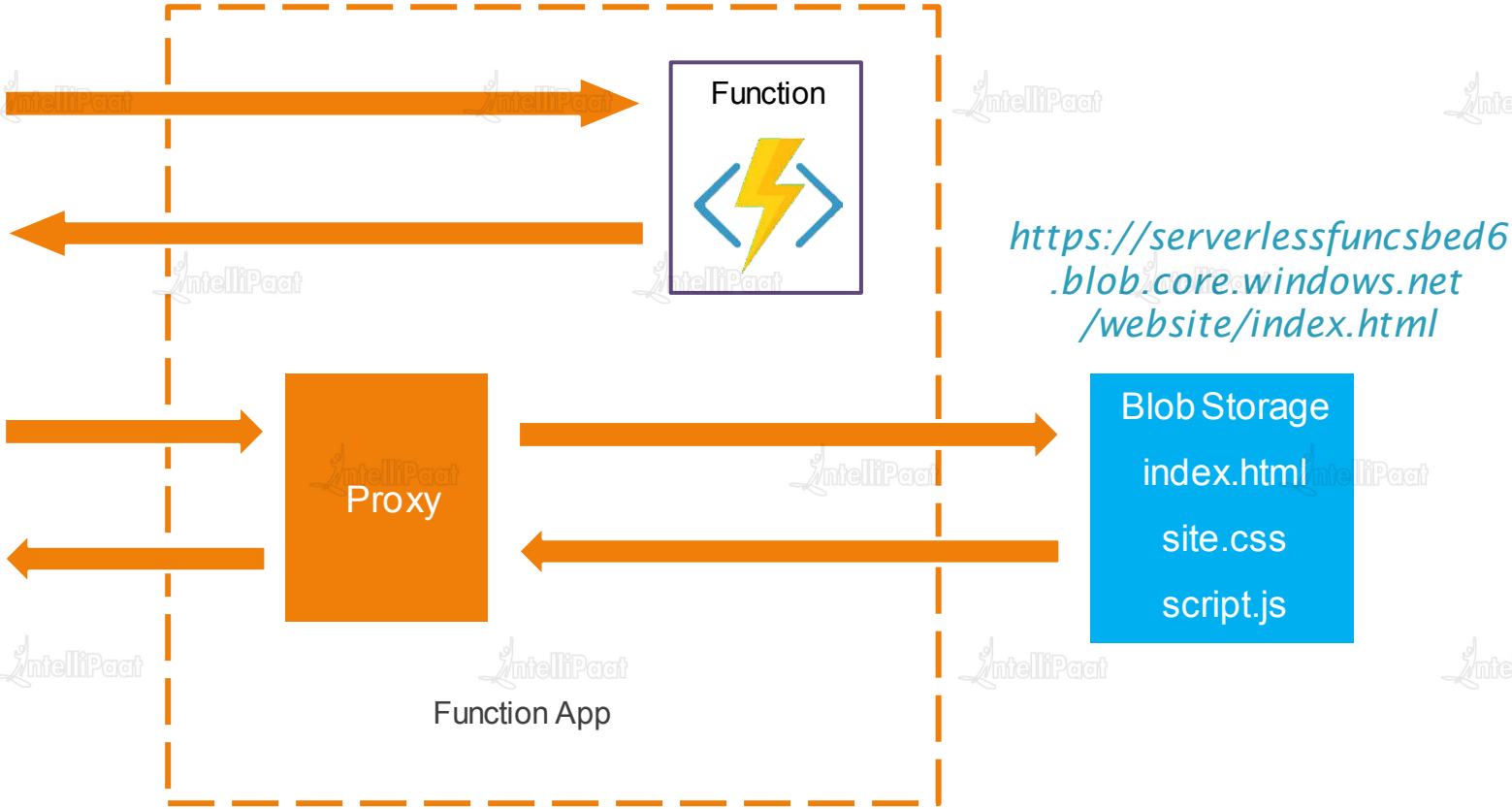
Hands-On

- Publish from Visual Studio
- Testing a FunctionApp
  - Configure CORS

Azure Functions Proxies



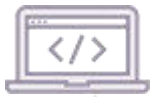
Azure Functions Proxies



Running Functions in Containers

Hosting Function Apps

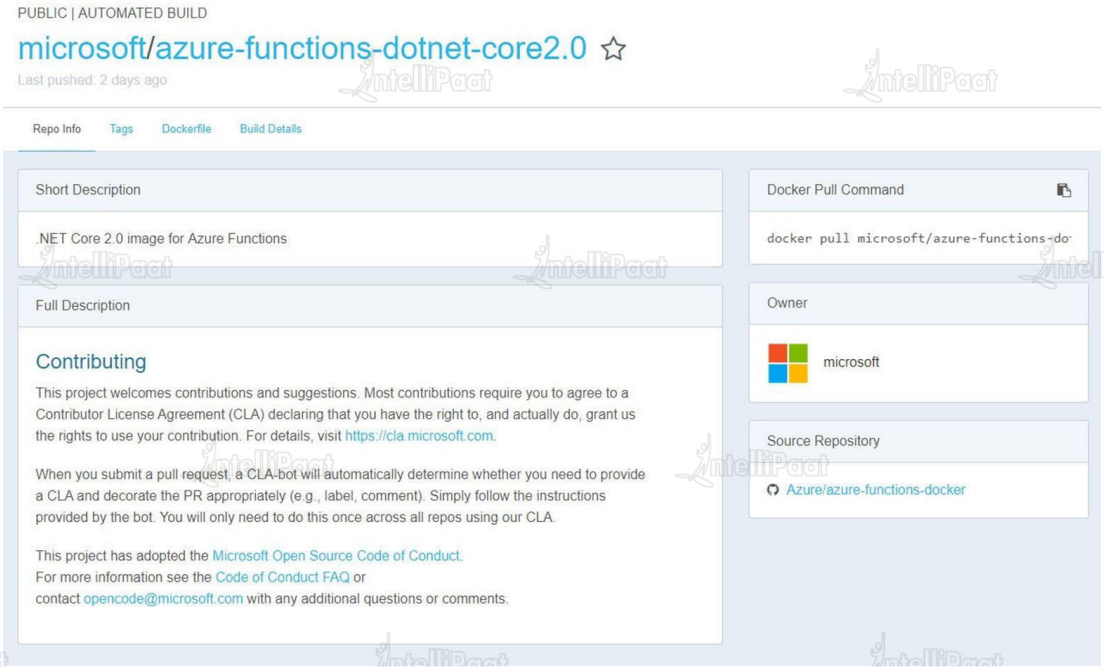
App Service plan in Azure



Locally for development (cross-platform)

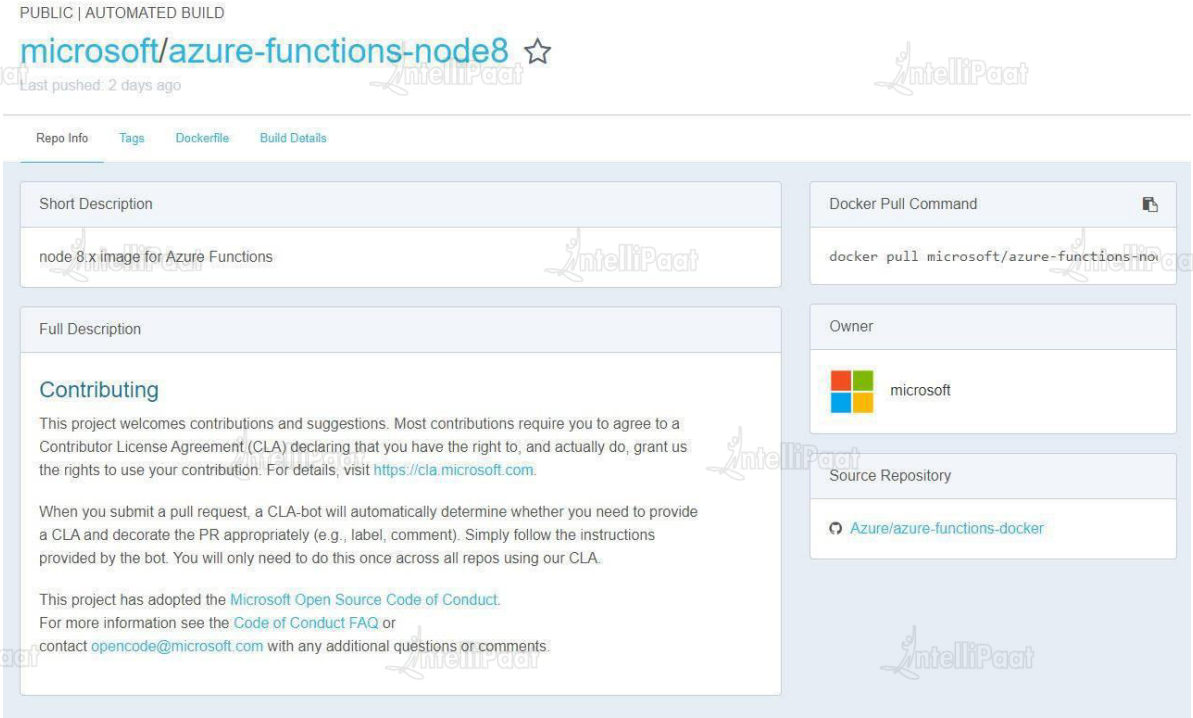
In a Docker container

Hosting Function Apps



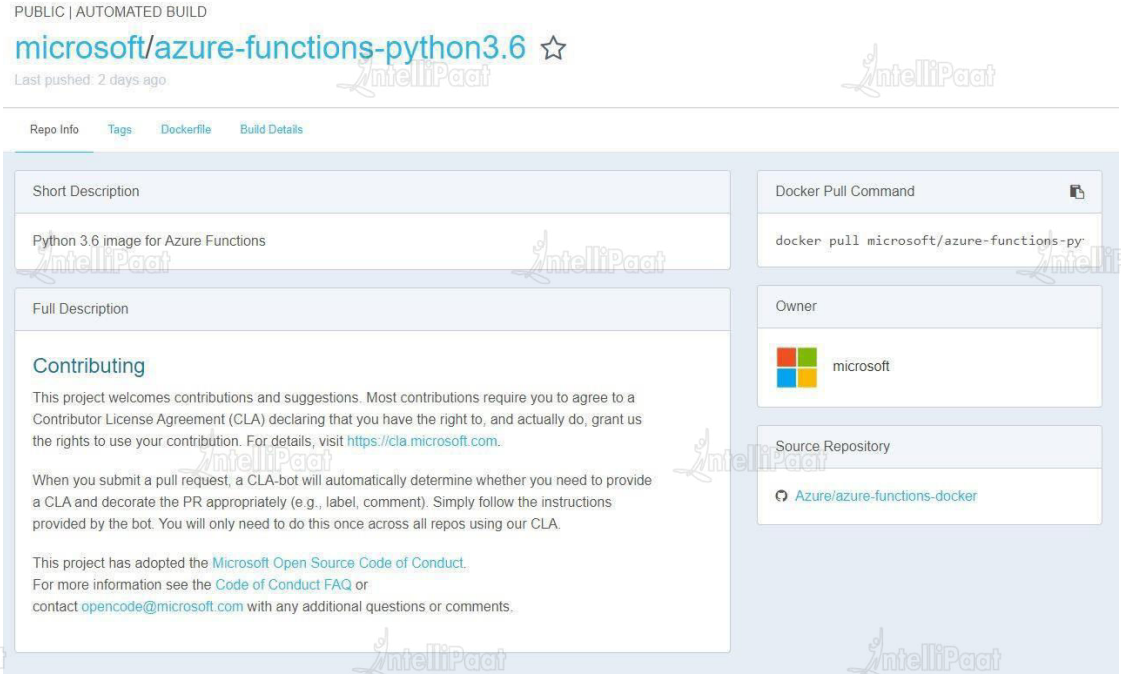
https://hub.docker.com/r/microsoft/azure-functions-dotnet-core2.0/

Hosting Function Apps



https://hub.docker.com/r/microsoft/azure-functions-node8/

Hosting Function Apps



https://hub.docker.com/r/microsoft/azure-functions-python3.6/

Limitations

Limitations

- Consumption plan benefits
  - Automatic scaling
  - Pay only while your functions run
- Connected resources
  - Azure Storage Accounts
  - Azure Application Insights

India: +91-7847955955

US: 1-800-216-8930 (TOLL FREE)

sales@intellipaat.com

24/7 Chat with Our Course Advisor