Journal of Scientific and Engineering Research, 2021, 8(2):229-241



Research Article

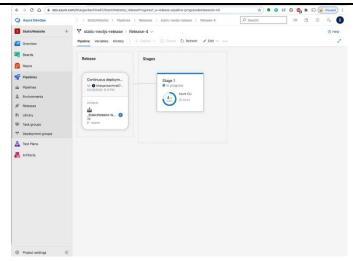
ISSN: 2394-2630 CODEN(USA): JSERBR

Implementing CI/CD Pipelines for Python Azure Functions with Azure DevOps

Bhargav Bachina

Abstract Azure Functions provide a seamless solution for deploying small code segments in the cloud, supporting various programming languages. This paper explores deployment strategies, focusing on Azure DevOps for function app deployment. Additionally, it demonstrates creating a Python REST API using Azure Functions. Key sections include an introduction to Azure Functions and an example project showcasing Azure DevOps deployment. Through this exploration, developers gain insights into leveraging serverless computing for efficient application deployment and development.

Keywords Azure, Cloud Computing, Programming, Python, DevOps



AN AZURE Function is a simple way of running small pieces of code in the cloud. You don't have to worry about the infrastructure required to host that code. You can write the Function in C#, Java, JavaScript, PowerShell, Python, or any of the languages that are listed in the Supported languages in the Azure Functions (https://docs.microsoft.com/en-us/azure/functions/supported-languages) article.

There are a lot of deployment strategies when you deploy your Azure functions to production and your deployment strategy entirely depends on your application architecture and the DevOps tools you are using. For example, If you are using Jenkins you can build the build pipeline and deploy the pipeline in Jenkins 2.

Using Azure DevOps, you have a ready-made template to deploy your function app. In this post, we will see how we can deploy an Azure function app using Azure DevOps. In this post, we will go through how to write Python REST API using Azure Functions.

- Introduction
- Example Project



- Prerequisites
- Build Pipeline
- Release Pipeline
- Demo
- Summary
- Conclusion

1. Introduction

The function app contains individual functions in which your function code resides. Each function app contains either one or more than one function serving each endpoint. When the function app's Authorization level is anonymous you can access the function app endpoints directly from the internet. We can directly deploy our Azure functions to function app from VSCode with the Azure Functions extension (https://marketplace.visualstudio.com/items?itemName=ms-azuretools.vscode-azurefunctions) for Visual Studio Code. But, when you are working in a team, deploying directly from the local machine is not a feasible solution. We need to build a pipeline in which the entire team can collaborate.

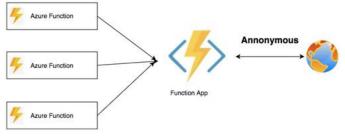


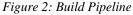
Figure 1: Function App Serving Directly

We must build two pipelines to deploy this application using Azure DevOps.

A. Build Pipeline

This pipeline takes the code from the Azure Repos or any git source and goes through a series of actions such as install, test, build and finally, generate the artifacts ready for the deployment.





B. Release Pipeline

This pipeline takes the artifact and deploys it into the function app. You can have pre-deployment conditions such as approvals, manual only, etc. for the release.



Figure 3: Release Pipeline

2. Example Project

This is a simple Python REST API where you can add tasks, delete tasks, update tasks, and get tasks. Here is an example project you can clone and run on your local machine.

// clone the projectgit clone https://github.com/bbachi/serverless-python-restapi-azure.git

// With Azure Fuctions extension installedfunc start



3. Testing

GET V https://pythorvestap	azurewebsites.net/api/tasks	Params Send 🗡 Sa
uthorization Headers (1) Body	Pre-request Script Tests	Cos
Туре	No Auth 🗸	
ody Cookies Headers (5) 1	603	Batus: 200 OK Time: 504 ms 8
Pretty Raw Preview JSON	× 15	C Q Save I
<pre>style="testpack" testpack" testpack" testpack style="testpack" testpack" testpack" style="testpack" testpack" testpack" style="testpack" testpack" testpack" testpack testpack" testpack testpack" testpack t</pre>	er.	

Figure 4: Testing the API

4. Prerequisites

You need to know a lot of things as prerequisites if you want to write a serverless Python REST API. First, you need to create two accounts: a Github account to store the source code and Microsoft Account to deploy that code using Function App Service. Let's create these accounts by following the below links. You can start both for free.

- Github Account (https://github.com/)
- Microsoft Azure Account (https://azure.microsoft.com/en-us/)
- Login into Azure DevOps (https://azure.microsoft.com/en-us/services/devops/)
- All the API code is written in NodeJS Azure functions. You need to be familiar with the following.
 - Azure Functions (https://azure.microsoft.com/en-us/services/functions/)
 - Azure Functions extension for Visual Studio Code (https://marketplace.visualstudio.com/items?itemName=ms-azuretools.vscode-azurefunctions)
 - Create a Subscription
 - Create a function app

A. Create a Subscription

You should have a Microsoft Azure Account. You can get a free account for one year. You should see the below screen after you login.

• **Azure Account** (https://azure.microsoft.com/en-us/free/)

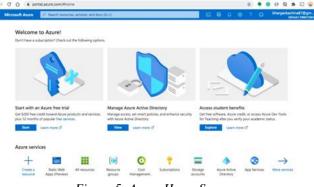
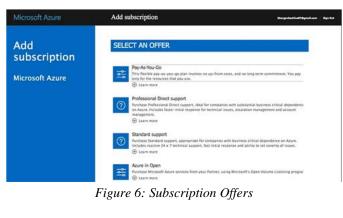


Figure 5: Azure Home Screen

You need to create a subscription for your account. The most common is Pay As You Go subscription.



Subscriptions &)
+ Abl								
View list of subscriptions for whic Showing subscriptions in bachina My role	h you Neve role-based as deectory. Don't see a sul	oni contri (NBAC) permisioni ti actiption? Switch directories		e resources. To view sul Status 🗇	laingitions	for which you have billin	g access, jobs here	
			- V	3 selected				
# selected								
E selected Apply								
Apply.	Show only subscription	selected in the global subscriptio	ra titer (D					
Apply Prowing 1 of 1 subscriptions	Dow only subscription	selected in the global subscriptio	ra liter 🖗					
Apply.	Show only subscription T ₂ Subscript			My role	. 74	Camerit cool	Status	14

Figure 7: Pay-As-You-Go Subscription

You need a subscription to be associated with your tenant so that all the cost is billed to this subscription.

B. Create a function app

Login into the portal and select a function app resource to create and give required fields such as app name, storage account, and resource group, runtime stack, etc.

	re			cs (G+/)
me > New > Funct	ion App >			
Create Func	tion App			
Basics Hosting	Monitoring	Tags Revi	iew + create	
basics Hosting	Monitoring	rags Revi	iew + create	
	ns lets you execute		as a logical unit for easier management, o a serverless environment without having t	
Project Details				
Select a subscription all your resources.	to manage deploye	id resources ar	nd costs. Use resource groups like folders	s to organize and manage
Subscription * ①		Pay-As-Ye	/ou-Go	~
Resource Gro	up * 💿	demo		~
		Create new	N	
Instance Details				
		demo-tas	sks-api	~
Instance Details Function App name *		demo-tas	isks-api	.azurewebsites.ne
		emo-tas		.azurewebsites.ne
Function App name *				.azurewebsites.ne
Function App name * Publish *		Code		→ .azurewebsites.ne → ↓

Figure 8 Creating a function app

Your deployment should be complete after some time.

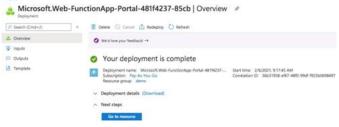


Figure 9: Deployment Completed



Here is the function app created and you can access the APIs with that URL if its authorization level is anonymous.

P Seech Ender5	- Of Broose () Notesh 🗌 Stop () Austart 🖉	1-10 \$ Gets	ulitish profile 🦻 Never publish pri	offer 🕴 Download	app content 👔 Delante 🗢 Send us pi
Dieves	Chill have to access Application Insights for maritaring	and profiling for you	NO. 17		
Activity log	 Inertials 			-	
Access control (AM)	Residure group (change) / demo			105	https://denoinaika.golatumeettaitas.net
Tags	Safer Burning			Operating System	
Clapione and solve problems	Location Central US				ASP-dance Blass (17. D)
	Externation Unanget Pay As Too Ge.				The More
Security				Runtime service	
Elents (preview)		HE140100.30		Autora series	3.0.151820
incline	Tage (change) Cick here to add tage				
functions	Metrics Features (N Notifications (1) Quid	0.00			
Applend	Methods Tearbares (It) Motifications (1) Quic	katart			
Area films					
	Memory working set	\$	Function Execution Count		\$
	Menory working set	s?	Function Execution Count		ø
Pressee		\$	Function Execution Count		ø
Proves		x0	Function Execution Count		ø
Provine oployment Deployment stats	10 10 10 10	*	1		\$
Process gelogeneet Deplogeneet stors Deplogeneet Center (Classic)	8. 8. 8. 8. 8.	\$	1		\$
Presse glogment Deployment stors Deployment Center (Clessic)	10. 10. 10. 10. 10. 10. 10.	\$	1		\$
Prawee playment Deployment sites Deployment Center (Classic) Deployment Center	8. 8. 8. 8. 8.	\$	1		2
Proces playment Daphyment bits Daphyment Center (Dasid) Daphyment Center Mings	10. 10. 10. 10. 10. 10. 10. 10. 10. 10.	\$	1		Ø
Proces playment Daphyment bits Daphyment Center (Dasid) Daphyment Center Mings	8. 8. 9. 9. 9. 9. 9. 9. 9. 9. 9. 9.	\$	1		*
Preses playment Lagtoynent sizes Lagtoynent Center (Cessic) Lagtoynent Center mings	10. 10. 10. 10. 10. 10. 10. 10.		8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8		
App film Proce Proce	98. 39. 39. 39. 39. 39. 49. 49. 49. 49. 49. 49.	\$ 	1	140	9

Figure 10: Function app created

5. Build Pipeline

We need to build pipelines as part of continuous integration first. The way this pipeline works is that the moment you check-in code into GitHub or Azure Repos, this pipeline builds the project, test it, make the built artifact ready for deployment. Let's follow all the steps to build this pipeline.

- 1. Create a Project in Azure DevOps
- 2. Create a Repo and Put your code in Azure Repos
- 3. Create a pipeline that takes it from the source repository.
- 4. Install all dependencies.
- 5. Run the tests.
- 6. Build the code.
- 7. Copy files from source for staging.
- 8. Archive all the copied files.
- 9. Finally, publish the artifact.
- 10. Enable Continuous Integration with triggers.

Let's create a project in your Azure DevOps account. I named it **Tasks API** with visibility of your choice. You can make it public or private.

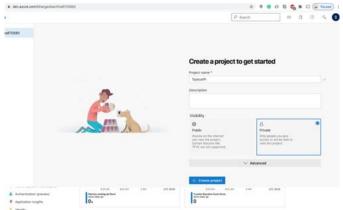


Figure 11: Creating Project

Once you have created this project you can see the dashboard which is empty right now.



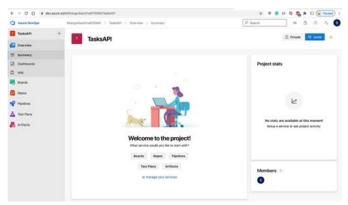


Figure 12: Project Created

A. Put your code in Azure Repos

It's time to create a repo and place all the code from the example project from above and push your code into this repo. I created a repo called **tasks-api** and pushed all the code into this repo.

Aurs Devõps Uningerbacht	attitite / TakaAN / Repai	/ Then / O taxo-apl -		P Seaton III 🖞 🛈
■ Teasanty + ● Teasanty ■ Derivere > Teasanty > Teasanty ■ Teasanty > Teasanty > Teasanty ■ Teasanty > Teasanty > Teasanty > Teasanty ■ Teasanty > Teasanty > Teasanty Teas	s j ss http: pri sour se se se se sp s https:japn	/ The (* Search of ~) Previous ~ (* Search of *) Previous ~ (* Search of *) Previous ~ (* Search of *) Sea	Lan change 3n ago 3n ago 3n ago 2n ago 3n ago 3n ago 3n ago 3n ago 3n ago 3n ago	A Second and a second sec
annes			3m ago	

Figure 13: Azure Repos

You can generate the Git credentials when you click on the clone button on the top right corner. You need these credentials if you want to push the code into this repo later.

Marge-Bartina670580 / %	skality / Regna / Files / @tasks-agi ~		Cione Repository	Ň
tasko-api tesko-api tesko-api tesko-apine	y Dynais o thy / hyperistication Files Contents History	a or fulder	Command line HTTP: SSH https://bhargavibachinal5700808/dex Useoname Vaconame	
) hi todo-get	Name 1	Last sharpe	bhargavbachina67	D
> Bit tado-post	a services	471.000	Password	
> == todo-put	the toolo-delete	477.900	ozśwpkhtatog477cgozztpśśgu3wy746abyłokormuacj	0 D
D altignore	The tudo-get	4m.ago	Comp the parameter toos. We shart share it and you won't be allo	419
D hest, som	Six Sudio-post	4m ago		
D local.settings.json	an todo-out	4m ago	DI	
C proviesjaon	D .pitiprore	Saturday	Clone in VS Code V	
Mi README.md	C rest,jeen	4m ago	Having inclusions authenticating in DIT to solve to get the lenser Or the Windows or nor program for interfal Europe. Android Studies	14+1214 4 5*
🗅 requirements.tet	C tocal.settings.json	4m ago	Windows command line	
	D proxies.json	4m.ago		
	Ms README.md	4m ago		
	C) requirements.txt	ém ago		
	serveriess-python-restapi-			

Figure 14: Git Credentials

B. Create a Build Pipeline

Let's start a pipeline by selecting a source and repository branch and by selecting a classic editor.



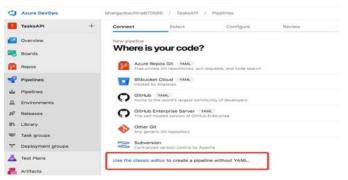


Figure 15: Select Classic Editor

bhargavbachina870580 / TasksAPI / Pipelines		,P Searc	h	= 0	© 8
	Select a source	C GitHub Enterprise Server	M Subversion	Bitbucket Cloud	Other Git
(\Rightarrow)	Team project	Ŷ			
	Repository	Ŷ			
Select your repository	Default branch for manual and sch				
Tell us where your sources are. You can customize how to get these sources from the repository later.	₽ main	~			
	Continue				

Figure 16: Select a source

On the next page, select the azure function for the Python template as Azure provides readymade templates for us.

padachina870580 / TaakaAPI / Pipelnes	P Search	
	Select a template Or start with an da Empty job	P Azure func 🛛 🛛
	Configuration as code	
\frown	VAML Looking for a before separations to configure time applicate using the finant The finance (MAR, pageline condition experiments Labor more	ran.
(\rightarrow)	Others	
<u> </u>	Azure Functions for NET Build and package + MCT bened Azure Functions application in temperature Azure Functions.	
Choose a template	Azure Functions for Node js Initial and participy a Node jo based Azure Functions application t application of Azure Functions.	n. 84
Choose a template that builds your kind of age. Durt worry if it's not an exact match, you can add and tostomory the basis later	Azure Functions for Powershell Institute out package a Foremental based Asure Functions adminuter Institute out Asure Functions.	to bu
	Azure Functions for Python Autor and package a Python based Acure Parentarie application to proceeding on Azure Functions.	Anty

Figure 17: Selecting a template

Once you click on the button Apply you can see the next screen with all these tasks. You can change the pipeline name as you want. If you notice the tasks, it is using the Python version 3.6, install dependencies, turns build script, archive files, and publish the artifact.



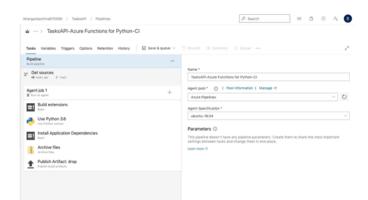


Figure 18: Template Applied

Click on the save and you can save this into any folder. Let's define the trigger.

C. Define the trigger

Let's define the trigger for this build. Edit the pipeline by clicking on the triggers and enable continuous integration. Any commit to the task-api repo main branch triggers this pipeline.

hargavbachina870580 / TasksAPI / Pipelines	🔎 Search 💷 🖻 📀 🖧 📵	ļ
··· > TasksAPI-Azure Functions for Python-CI		
Taska Variables Triggers Options Retention History 🛛 😂 Save & queu Continuous integration		
tasks-api	M tasks-api	
Scheduled + A	Enable continuous integration Batch changes while a build is in progress Branch filters	
Build completion + A	include v 👂 main v 📋	
	+ Add Puth Dans + Add	

Figure 19: Enable Continuous Integration

Let's build the pipeline manually this time by clicking on the Queue.

on TasksAPI-Azure Functions for Python-Cl			
Manually run by 📵 bhargavbachina87@g	mail.com Bachina		View 6 chang
Repository and version	Time started and elapsed	Related	Tests and coverage
tasks-api	C3 Just now	C 0 work items	A Get started
⊁main ≎		131 consumed	
Jobs			
Name		Status Duration	
Agent job 1		Queued	

Figure 20: Job Running

Click on Agent Job 1 to see the results.



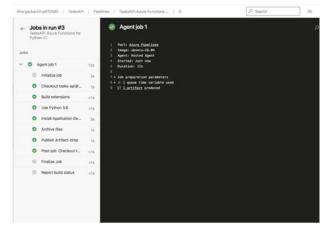


Figure 21: Job Details

Click on the artifact link in the logs to see the artifact details. This creates the Zip file.

hargavbachina870580 / TasksAPI / Pipelines / TasksAPI-Azure Functions / 3 / Published artifacts	P Search
← Artifacts	
Published Consumed	
Name	Size
Name	Size 8 MB

Figure 22: Zip Artifact

If you download the zip file and open you can see the Azure Functions project structure.

Today	Today	Today
3 3.zip Snip20210211_2014 Snip20210211_2014 Snip20210211_2024 Snip20210211_2034 Snip20210211_2045 Snip20210211_2045 Snip20210211_2046 Snip20210211_2046 Snip20210211_2046 Snip20210211_2046 Snip20210211_2049 Snip20210211_2049 Snip20210211_204 Snip20210211_2049 Snip20210211_214 Snip20210211_214 Snip20210211_214 Snip20210211_214	KRADME.md requirements.txt requirements.txt requirements.txt for todo-delete todo-get todo-post todo-post todo-put worker_venv	Linitpy function.json sample.dat

Figure 23: Zip file Extract

Here is the complete YAML for this build pipeline.

https://gist.github.com/bbachi/82288061db154706c20ccb67cf1c90ab#file-template-yaml You can get the above YAML from this edit pipeline screen.

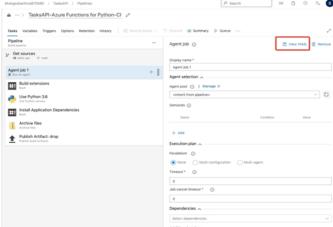


Figure 24: Getting YAML



6. Release Pipeline

We are done with the Continuous integration part and let's build a release pipeline for continuous delivery. Click on the releases and new release pipeline and follow these steps.

- A. Define the artifacts such as project, source, default version, etc
- B. Define a Stage for the deployment.

A. Define the Artifact

The first step is to define the artifact so that it takes that artifact after the build pipeline is completed. Make sure you have a trigger placed.

When you click on the releases you see an empty page with a new pipeline button. Click on it to create a new release pipeline. Select a template called *Deploy a function app to Azure Functions*.

eline Tasks - Variable	n Relevitor, Options History	Select a template Or start with an the Empty job	P function app X
Artifacts 1 + And	Stages) + Ans -	Others Openion a function app to Azure Deploy a function app to Azure Deploy a function app to Azure	
	A Stage 1 R Desist a transiers R	Deploy a function app to Azure with dot Deploy are function app to Azure Deploy and function are to a staging to so another to enstantian. Featured	
O Schenich Satur		Azure App Service deployment besty your sociation to Azure App tom the App on Workers, Lines, In Age, or Web-Net	Sarvice, Chotne

Figure 25: Selecting a pipeline

You can name the stage name. I named it Development.

rgedactival70580 / TaiksAPi / Panines / Releases	P Search III (2) (3) Pa
Microsofines > 7 New release pipeline getime © Taiks - Vasades Retention Options History	E Lies of Cherry Hands III Very Margare
Artifacts + Add + add an + add an Artifacts + Add Artifacts + Add Arti	Stage Diverganeet ⊒ Popperties ∧ Stage name Divergament
Solveste	Stage mover Of thergonetactivest?digmat.com Succines X

Figure 26: Stage Name

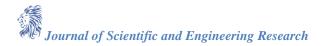
Click on the Job task section of the stage to define the tasks.

_			
Artifacts + Add	Stages 1 + Add V	Stage Delete 3 Mo Development	ve ~
		且 Properties ∧ Name and owners of the stage	
		Stage name	
+ Add an artitlact	A O 1 job, 1 task A	Development	
		Stage owner C bhargavbachina87@gmail.com Bachina	
Schedule not set			
on not set			

Figure 27: Job

B. Define a Stage for the deployment.

To deploy the function app, you need a subscription and create a function app in the Azure portal as we discussed in the prerequisites section. We need to select the subscription and give the function app name that we created in the prerequisite section.



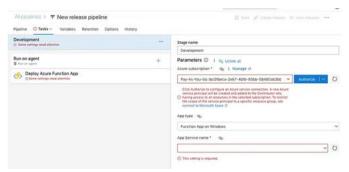


Figure 28: Subscription and App Service Name

Once it is Authorized this is how it looks like

Al pipeline > 🐨 New release pipeline		Save of Child Street TH Day stream		
Ppeline Tasks Variables Relention Options Histor	×			
Development	-	Stage name		
		Development		
Run on agent	+	Parameters () og Unins all Abure subscription * og Mahage it		
Deploy Azure Function App Azure Texestore		Pay-As-You-Gs (https://www.astation.org/abs/s6480ab30b30)	¥.	3
		O Second to adversarian Par-Ja-Par-Jar		
		App type: qp		
		Punction App on Lihux		
		App Service name * 1 %		
		demo-tasks-adi		9

Figure 29: Release Pipeline

We need to add an artifact by selecting the source pipeline

rgavbachina870580 / Tasks	sAPI / Pipelines / Releases		
Upipelines > 7 News	release pipeline	Add an artifact	
peline Tasks - Variables	s Retention Options History	Source type	
Artifacts + Add	Stages + Add ~	Dulit Asver Inpo CID-LO PYC	
		5 more artifact types ~	
		Project *	
Mi Add an artifact	B Development R	TasksAPI	\sim
		Source (build pipeline) *	
		TasksAPI-Azure Functions for Python-C	×
Schedule not set		Default version *	
		Latest	v
		Source alas *	
		_TasksAPI-Azure Functions for Python-Cl	
		The artifacts published by each version will be available for deployment in release pipelines. The latest successful build of TasksAPI-Azare Functions for Python-Cl published the following artifacts: drap.	
		Add	

Figure 30: Adding an Artifact

Once it is added, you can have a trigger defined. Click on the flash icon

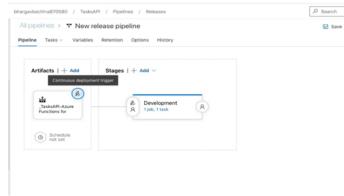


Figure 31: Enable Trigger

Once it is enabled, don't forget to save the pipeline.

Journal of Scientific and Engineering Research

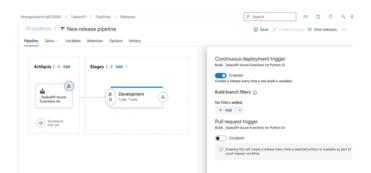


Figure 32: Continuous deployment trigger

7. Demo

It's time for the demo. As soon as you check-in the code to mine it triggers the build pipeline. Usually, you don't check in the code directly into main instead you create a pull request. I just used a main here for simplicity.

A. Build Pipeline Demo

I just added one more task to the array and push it to the main branch. As soon as we pushed the code, the build pipeline triggered.

https://miro.medium.com/v2/resize:fit:1400/1*0RVkc6t1-GyvxA5LVmjW2g.gif

B. Release Pipeline Demo

As soon as the building is completed and a new artifact will trigger the release pipeline. <u>https://miro.medium.com/v2/resize:fit:1400/1*5xbT_9lE-nCaGtzAdWhJIA.gif</u>

Once it is succeeded, you can go to the function app in the portal and verify the functions.

Home > demo-tasks-api {/i} demo-tasks-api Fu Function App	inctions			×
	+ Add 🖒 Rafresh 🖹 Delete			
 Overview Activity log 	Lefting functions in the Asure portal is not supported for Linu	e Consumption Function Apps.		
 Access control (AM) Tags 	P filter by name.			
🖉 Diagnose and solve problems	Name 1:	Trigger 1.	Status 12	
Security	D todo-delete	HTTP	Enabled	
👂 Events (preview)	C sudo-gat	HTTP	Enabled	
Functions	tode-post	HITP	Enabled	
(A) Functions	todo-put	HTTP	Enabled	
📍 App keys				
App files				
 Provies 				
Deployment Toployment slots				

Figure 33: Functions Deployed

You can test the functions either directly in the portal or you can the function URL and hit it in the browser or in the postman. You can see the 5 tasks in the array response.

×

JI Search (Cmd+/)) * El tare X Storet. D Antent. □ Settillar 1 Upload	Charles and Charles an
(H) Overview	Earling functions in the particle and segmented for Since Consemption Function Appl.	Piput Delput
Oeveloper	demo-tasks-api \ todo-get \ivitpy 🗸	HTTP response code
Cole Tot Support Supp	<pre>Interf loging_ine in Interf loging_ine inter Prime structure.https://www.imit.gov/imit.g</pre>	2009
	The program of the second sec	MP. 5. "Marine": "Samplere 1000, "Status": "sampleted" 1

Figure 34: Testing the function

You can hit the below URL in the browser and see the response as below.

https://demo-tasks-api.azurewebsites.net/api/tasks



Figure 35: Testing the function

8. Summary

- One way of building Python REST API is to use the Azure function app.
- There are a lot of deployment strategies when you deploy your Azure functions to production and your
- deployment strategy entirely depends on your application architecture and the DevOps tools you are using.
- Using Azure DevOps, you have a ready-made template to deploy your function app.
- We must build two pipelines to deploy this application using Azure DevOps: Build pipeline and Release pipeline.
- The build pipeline generates the artifact as soon as there is a commit in the source repository.
- The Release pipeline takes the artifact and releases it to the appropriate environment.
- Authentication should be made before you can use pipelines to access your Repos and upload files to the Azure subscription account.
- You can use task groups to put the common tasks in one place across the environments.
- You can use Azure key vaults to store the access keys for your storage account.
- You can make use of Pipeline Variables while creating tasks. It's a convenient way to get key bits of data into various parts of the pipeline.

9. Conclusion

In conclusion, this paper explored the process of building a Python REST API using Azure Functions, emphasizing deployment strategies with Azure DevOps. The discussion highlighted the significance of tailoring deployment strategies to match the application architecture and DevOps tooling. Leveraging Azure DevOps, developers can utilize pre-configured templates for function app deployment, streamlining the deployment process. Additionally, the paper outlined the necessity of creating two pipelines—Build and Release—to automate artifact generation and deployment across environments. Authentication measures were underscored as essential for accessing repositories and Azure subscription accounts. Furthermore, the use of task groups and Azure Key Vaults was recommended to enhance efficiency and security in deployment workflows. Lastly, the utilization of Pipeline Variables was highlighted as a practical approach to managing crucial data within the deployment pipeline, ensuring smooth and reliable deployment processes.

References

- [1] Azure Cloud Documentation https://azure.microsoft.com/en-us
- [2] Azure DevOps Documentation https://azure.microsoft.com/en-us/products/devops/boards/
- [3] Azure Functions https://azure.microsoft.com/en-us/products/functions
- [4] Python Documentation https://docs.python.org/3/