



[DELPHICON 2023]

 Embarcadero®

When Delphi reaches the Cloud!

by Dion Mai



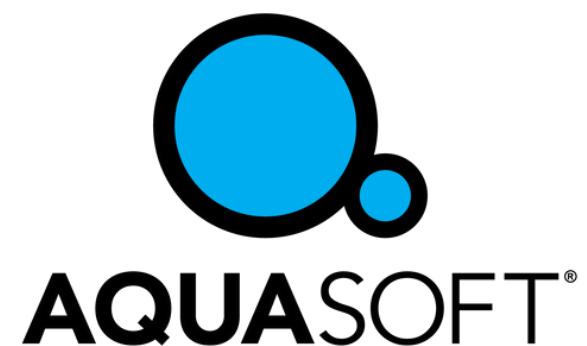
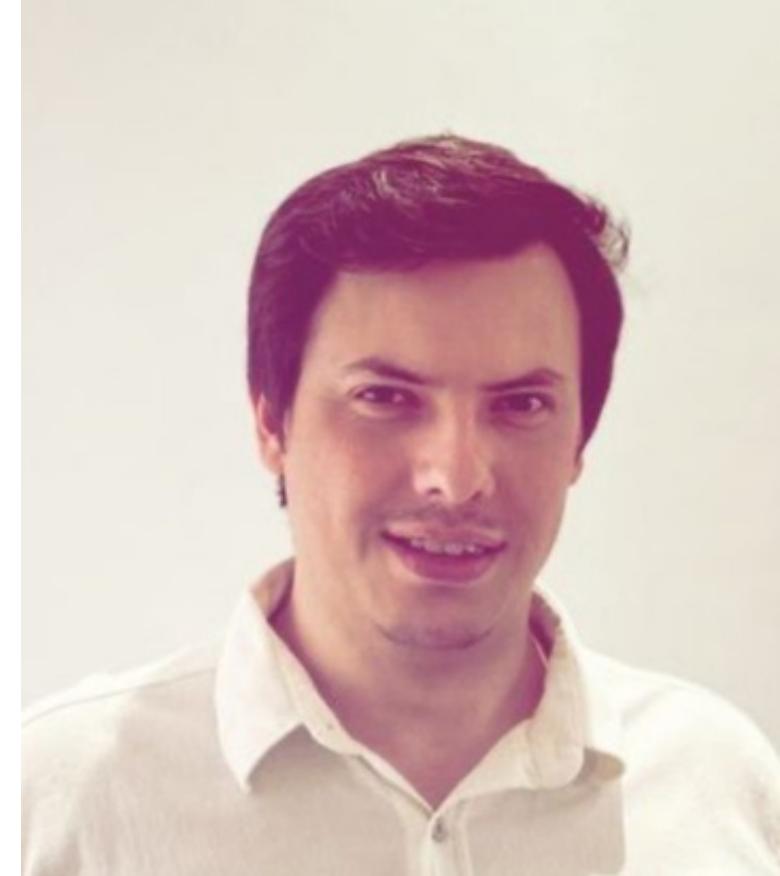
Agenda

- What is serverless?
- How to implement the back-end?
- How to implement the front-end?
- Advantages of using serverless

About me

Dion Mai

- Development Manager at Aquasoft
- MVP Embarcadero Delphi & C++ Builder
- Delphi Master Certified
- Bachelor of Engineering in Control and Automation





What is Serverless?



APPLAIED FAAS

*Serverless is applied over FaaS - Funcion as a Service;
but server "Event Triggered".*

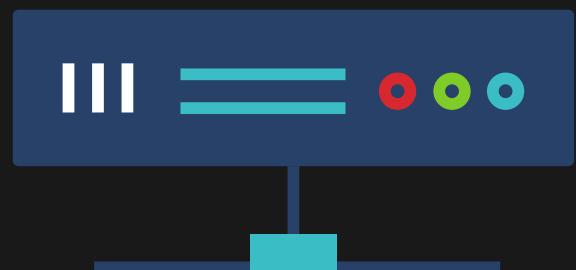
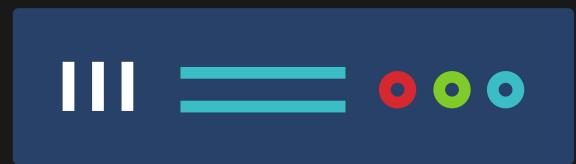
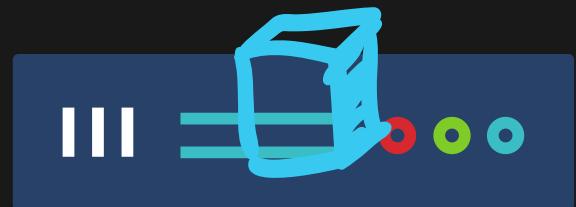


What is Serverless?

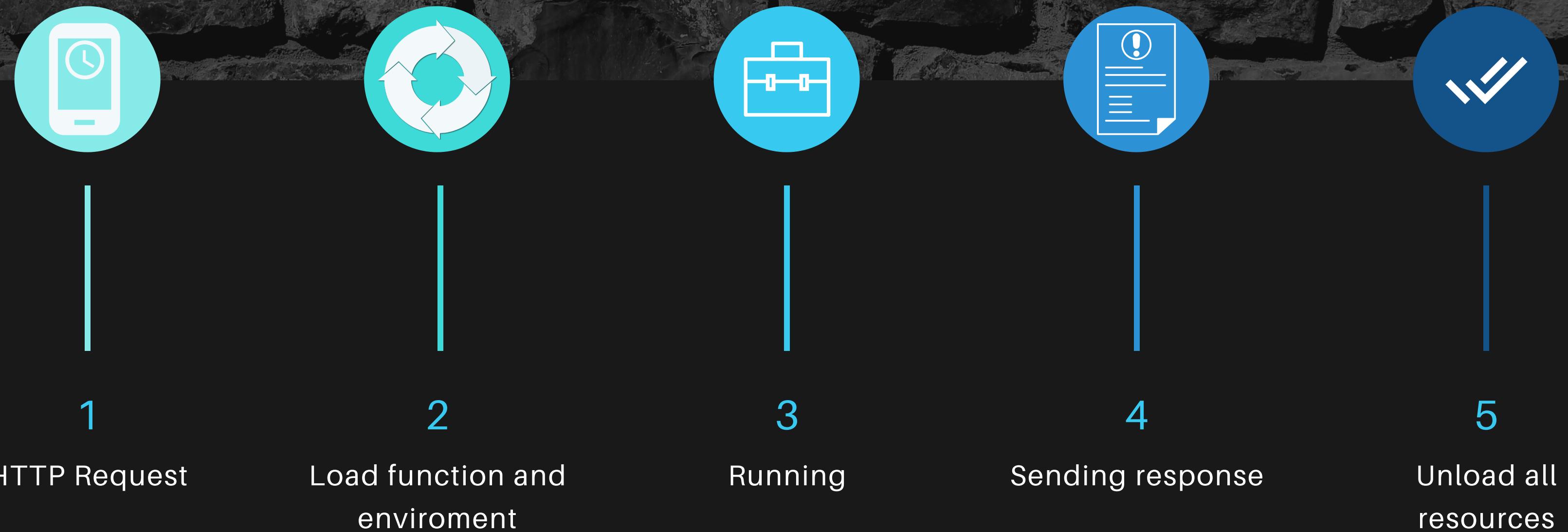


SERVERLESS HAS A SERVER?!

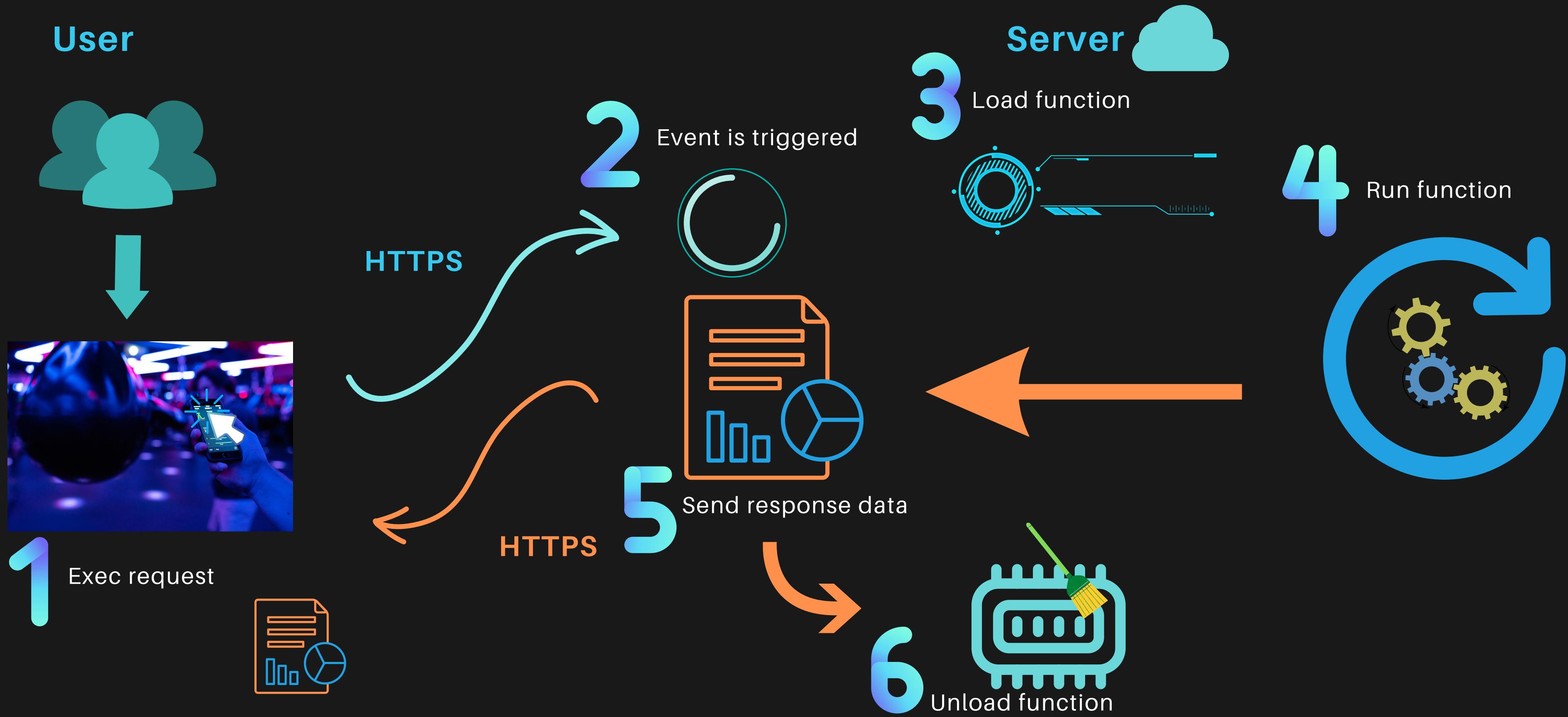
Maybe they put the wrong name...



Serverless - at work!

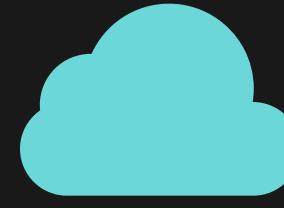


Serverless - once again...



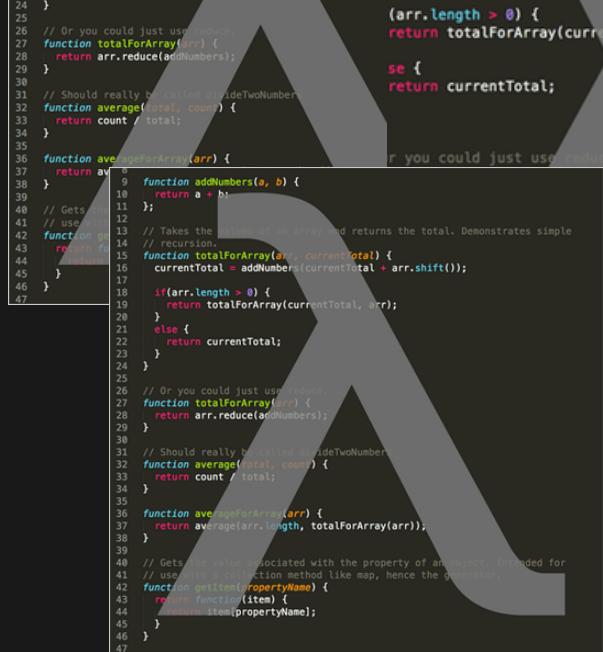


So Serverless?

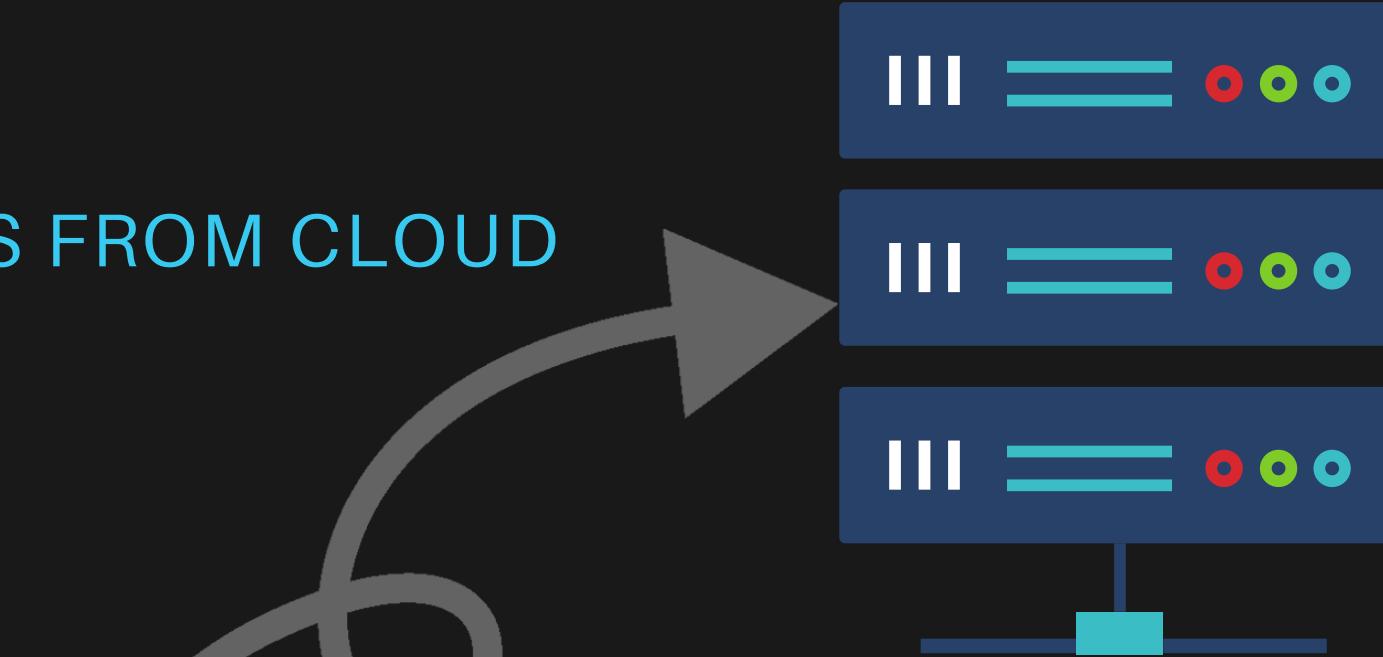
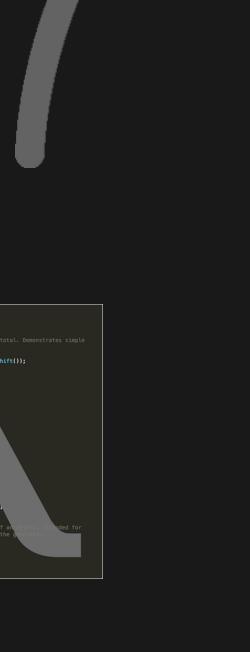
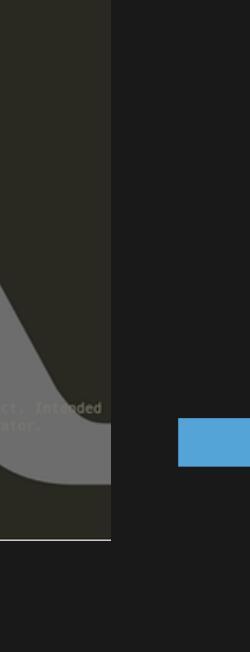


WE CALL ISOLATED FUNCTIONS FROM CLOUD

Any system with "infinite functions"



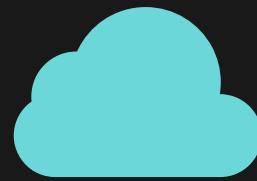
```
9 function addNumbers(a, b) {
10    return a + b;
11}
12
13 // Takes the values of an array and returns the total. Demonstrates simple recursion.
14 function totalForArray(arr, currentTotal) {
15    currentTotal = addNumbers(currentTotal + arr.shift());
16
17    if(arr.length > 0) {
18        return totalForArray(arr, currentTotal);
19    }
20    else {
21        return currentTotal;
22    }
23}
24
25 // Or you could just use reduce.
26 function totalForArray(arr) {
27    return arr.reduce(addNumbers);
28}
29
30 // Should really be called divideTwoNumber.
31 function average(total, count) {
32    return count / total;
33}
34
35
36 function averageForArray(arr) {
37    return average(...arr);
38}
39
40 // Gets the average of an array and returns the total. Demonstrates simple recursion.
41 function averageForArray(arr) {
42    if(arr.length > 0) {
43        return totalForArray(arr, currentTotal);
44    }
45    else {
46        return currentTotal;
47    }
48}
49
50 // Or you could just use reduce.
51 function averageForArray(arr) {
52    return arr.reduce(addNumbers);
53}
54
55 // Should really be called divideTwoNumber.
56 function average(count, total) {
57    return count / total;
58}
59
60 function averageForObject(arr) {
61    return average(arr.length, totalForArray(arr));
62}
63
64 // Gets the value associated with the property of an object. Intended for use with a collection method like map, hence the generator.
65 function getitem(propertyName) {
66    return function(item) {
67        return item[propertyName];
68    }
69}
```



*DDD(Domain Driven Design)
Model to help build systems
with functions



Serverless - implementing the backend



STEPS TO BUILD

- Select one cloud provider: Azure, AWS, GoogleCloud, IBM Cloud, Alibaba cloud
- Create your account, yes with a credit card on it
- Choose your backend technology
 - Use a native language
 - Use containers, small VMs to run
- Write your function
- Create a deploy environment
- Publish your function
- Other configurations
 - Function execution limit
 - Regions with access
 - Security
 - Logs



Serverless - Lets see Azure sample!



ONLINE MANUAL

Microsoft has an updated online manual:

<https://learn.microsoft.com/en-us/azure/azure-functions/create-first-function-vs-code-other?tabs=go%2Cwindows>



The screenshot shows a Microsoft Learn page for Azure Functions. At the top, there's a navigation bar with links for Microsoft, Learn, Documentation, Training, Certifications, Q&A, Code Samples, Shows, and Events. Below that is a secondary navigation bar for Azure with links for Product documentation, Architecture, Learn Azure, Develop, and Resources. On the left, there's a sidebar with a 'Filter by title' input field and a list of documentation sections: Functions Documentation, Overview, Quickstarts (with 'Create your first function' and 'Visual Studio Code' sub-items), and a 'Learn' section. The main content area displays the title 'Quickstart: Create a Go or Rust function in Azure using Visual Studio Code'. The URL in the browser address bar is <https://learn.microsoft.com/en-us/azure/azure-functions/create-first-function-vs-code-other?tabs=go%2Cwindows>. The page includes standard Microsoft UI elements like a back arrow, a search bar, and a top navigation bar.



Serverless - Azure



1 - CREATE ACCOUNT

- 30 days of credit to test
- Some resources are *always free* based on consumption
- Yes, here comes your credit card!

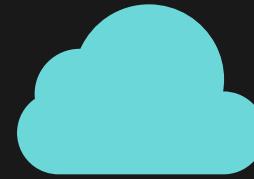
<https://portal.azure.com/>

The screenshot shows the Microsoft Azure portal interface. At the top, there's a browser-like header with a tab labeled "Home - Microsoft Azure". Below it is a toolbar with standard navigation icons (back, forward, refresh) and a search bar containing the URL "https://portal.azure.com/#home". The main content area has a blue header bar with the "Microsoft Azure" logo and a search bar that says "Search resources, services, and docs (G+/)". Below this is a section titled "Azure services" featuring several icons and links:

- Create a resource** (plus sign icon)
- Function App** (lightning bolt icon)
- Subscriptions** (key icon)
- Quickstart Center** (rocket icon)
- Virtual machines** (monitor icon)
- App Services** (globe icon)



Serverless - Azure Sample



1 - ACCOUNT DETAILS

- 1 million free by month
- Even for 5 million execs, the price can be around US\$ 1.00
- A virtual machine is many times more expensive

<https://azure.microsoft.com/en-us/pricing/calculator/>

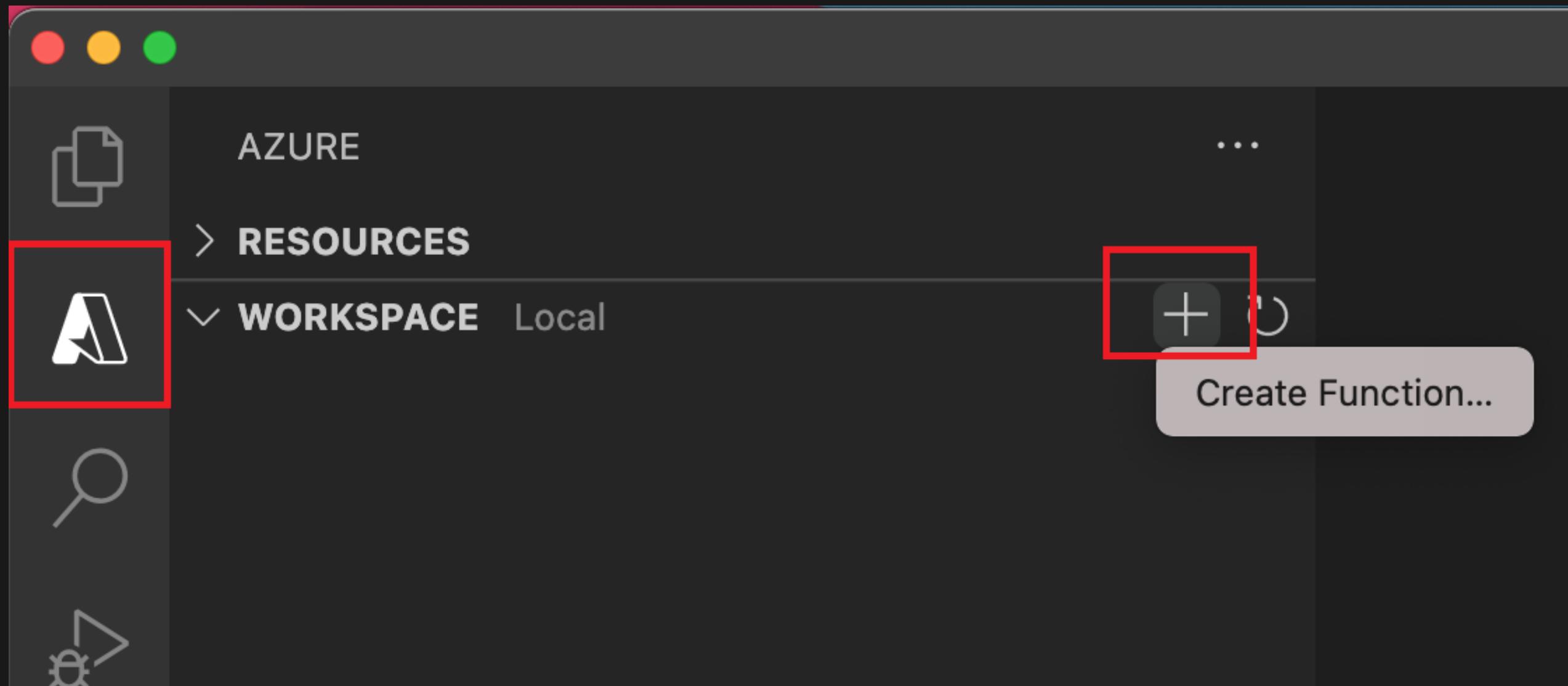


Serverless - Azure Sample



2 - CONFIGURING DEVELOPMENT ENVIRONMENT

- Install Visual Studio Code/Visual Studio
- Install Azure Functions Core Tool
- Create the first function locally





Serverless - Azure Sample



3 - DEFINING FUNCTION TYPE

Functions can be written in the native language or published in the form of executables or containers

Differences:

- Small delay in runtime
- Can be more expensive because of the extra load of data and resources

Function for non native
language(Delphi!)

Prompt	Selection
Select a language for your function project	Choose Custom Handler .

Function in native language

Prompt	Selection
Select a language	Choose C# .
Select a .NET runtime	Select .NET 6 .



Serverless - Azure Sample



4 - HOW TO CONFIGURE

The screenshot shows the Visual Studio Code interface with the following details:

- EXPLORADOR**: Shows the project structure:
 - EDITORES ABERTOS**: host.json, function.json (selected), .vscode, AZUREDELPHI, .funcignore, .gitignore, functionDelphiAzure.exe, host.json, local.settings.json, logExec.txt.
 - AZUREDELPHI**: .vscode, HttpExample (selected), function.json, .funcignore, .gitignore, functionDelphiAzure.exe, host.json, local.settings.json, logExec.txt.
- function.json** file content (selected in the Explorer):

```
1  {
2   "bindings": [
3     {
4       "authLevel": "anonymous",
5       "type": "httpTrigger",
6       "direction": "in",
7       "name": "req",
8       "methods": [
9         "post"
10      ]
11    },
12    {
13      "type": "http",
14      "direction": "out",
15      "name": "res"
16    }
17  ]
18 }
```

The code editor highlights the "methods" array with a blue rounded rectangle.



Serverless - Azure Sample



4 - CONFIGURE FOR EXECUTABLES

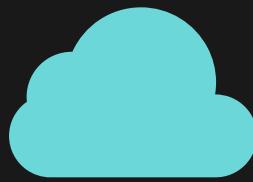
The screenshot shows the VS Code interface with the Explorer sidebar open, displaying files in the 'AZUREDELPHI' folder. The 'host.json' file is selected in the Explorer and is also the active tab in the main editor area. The code in the editor is as follows:

```
{
  "version": "2.0",
  "logging": {
    "applicationInsights": {
      "samplingSettings": {
        "isEnabled": true,
        "excludedTypes": "Request"
      }
    }
  },
  "extensionBundle": {
    "id": "Microsoft.Azure.Functions.ExtensionBundle",
    "version": "[3.*, 4.0.0)"
  },
  "customHandler": {
    "enableForwardingHttpRequest": true,
    "description": {
      "defaultExecutablePath": "functionDelphiAzure.exe",
      "workingDirectory": "",
      "arguments": []
    }
  }
}
```

A red rounded rectangle highlights the 'arguments': [] section of the 'customHandler' configuration.



Serverless - Azure Sample



4 - CREATING THE EXECUTABLE IN DELPHI

```
procedure RunServer(APort: Integer);
var
  LServer: TIIdHTTPWebBrokerBridge;
  LResponse: string;
begin
  LServer := TIIdHTTPWebBrokerBridge.Create(nil);
  try
    var portEnvVarStr: string := GetEnvironmentVariable('FUNCTIONS_CUSTOMHANDLER_PORT');
    var portEnvVar: Integer;
    if TryStrToInt(portEnvVarStr, portEnvVar) then
    begin
      LServer.DefaultPort := portEnvVar;
    end else begin
      LServer.DefaultPort := APort;
    end;

    StartServer(LServer);
    while True do
    begin
      // 
    end;

  finally
    StopServer(LServer);
    LServer.Free;
  end;
end.
```



Serverless - Azure Sample



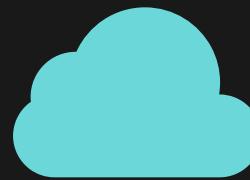
4 - IMPLEMENTING IT NATIVELY

The screenshot shows the Visual Studio interface with the following details:

- Code Editor:** The main window displays the `FunctionNovidadesERP.cs` file. The code implements an Azure Function named `FunctionNovidadesERP` that handles HTTP requests. It includes logic to validate an app key and return a JSON response.
- Solution Explorer:** The right-hand pane shows the solution structure for the project `NovidadesERP`, which includes files like `host.json` and `local.settings.json`.
- Properties:** A smaller pane below the Solution Explorer shows the properties for the selected item.
- Status Bar:** The bottom status bar indicates the current line (Ln: 21), column (Car: 6), and encoding (SPC CRLF).
- Bottom Navigation:** Buttons for saving changes, adding to source control, and selecting a repository are visible at the bottom.



Serverless - Azure Sample



5 - LOCAL RUN AND TEST!

```
PS C:\Users\...> func start --verbose
    %%%%%%
    %%%%
    @  %%%%  @
    @@ %%%%  @@
    @@@ %%%%%%% @@@
    @@ %%%%%%% @@@
    @@ %%%  @@
    @@ %%  @@
    @@ %  @@
    %

Azure Functions Core Tools
Core Tools Version:      4.0.4785 Commit hash: N/A (64-bit)
Function Runtime Version: 4.10.4.19213

[2022-10-24T17:55:20.255Z] Building host: startup suppressed: 'False', configuration suppressed: 'False', startup operation id
30f089'
[2022-10-24T17:55:20.263Z] Reading host configuration file 'C:\Users\...\'CloudFunct
[2022-10-24T17:55:20.266Z] Host configuration file read:
```



Serverless - Azure Sample



5 - LOCAL RUNNING

Functions:

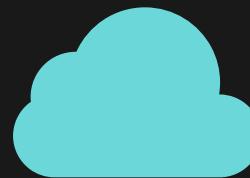
HttpExample: [POST] http://localhost:7071/api/HttpExample

[2022-10-24T17:55:22.345Z] Worker process started and initialized.

[2022-10-24T17:55:26.554Z] Host lock lease acquired by instance ID '000000000000000000000000442FC398'.



Serverless - Azure Sample



6 - CLIENT CALLING THE FUNCTION LOCALLY

The screenshot shows a Postman request configuration and its corresponding response.

Request Headers:

KEY	VALUE	DESCRIPTION
app-key	[REDACTED]	Description
Key	Value	Description

Response Body (Pretty JSON):

```
1  {
2      "atualizacoes": [
3          "boletos": "Funcionalidade de pagamento por PIX",
4          "dashboards": "Consolidação dos totalizadores filtrados",
5          "relatórios": "Adicionados múltiplos filtros"
6      ],
7      "licençaAtiva": true,
8      "sistema": "Controle de Updates by Azure + Delphi",
9      "status": "1 nova atualização disponível"
10 }
```



Serverless - Azure Sample



6 - CLIENT CALLING IT FROM DELPHI

Object Inspector Data.TestCloudFunctions X

RESTClientAzure TRESTClient

Properties Events

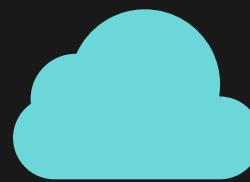
Accept	
AcceptCharset	
AcceptEncoding	
AllowCookies	<input checked="" type="checkbox"/> True
Authenticator	
AutoCreateParams	<input checked="" type="checkbox"/> True
BaseUrl	http://localhost:7031/api/Novidade
> BindSource	RESTClientAzure.BindSource
ConnectTimeout	30000
ContentType	
FallbackCharsetEncoder	utf-8
HandleRedirects	<input checked="" type="checkbox"/> True
> LiveBindings Designer	LiveBindings Designer
Name	RESTClientAzure
Params	(TRESTRequestParameterList)
ProxyPassword	
ProxyPort	0
ProxyServer	
ProxyUsername	
RaiseExceptionOn500	<input checked="" type="checkbox"/> True
ReadTimeout	30000
> RedirectsWithGET	[Post301,Post302,Post303,Put303,Delete303]
> SecureProtocols	[]

Diagram View:

```
graph LR; RESTClientAzure[RESTClientAzure] --> RESTResponseAzure[RESTResponseAzure]; RESTClientAzure --> TRequestAzure[TRequestAzure]; RESTResponseAzure --> TRequestAzure;
```



Serverless - Azure Sample



6 - CLIENT CALLING IT FROM DELPHI

```
• procedure TdmdTestCloudFunctions.GetAzureVersionUpdates;
• begin
5 RESTRequestAzure.Execute;
• var jsonContent: string := RESTResponseAzure.Content;
•
• if jsonContent <> '' then
• begin
0
• end else begin
•   dmdNotifications.Send('Sem resposta!', 'Não foi possível verificar novidades da versão');
• end;
• end;
```

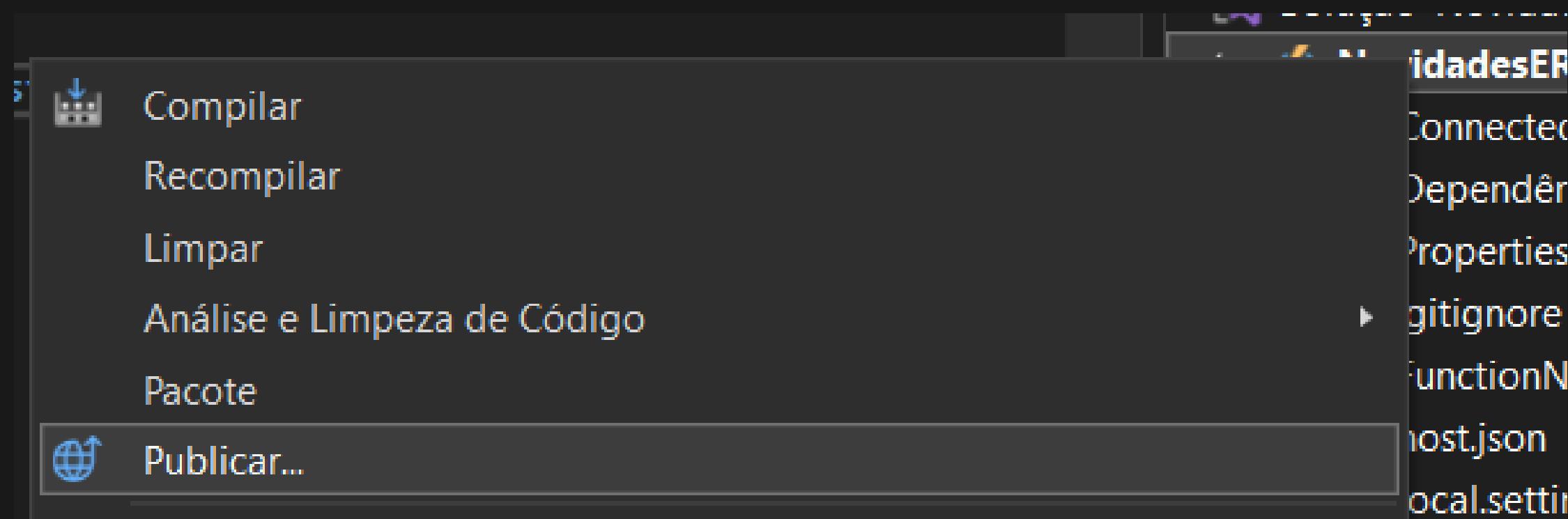


Serverless - Azure Sample



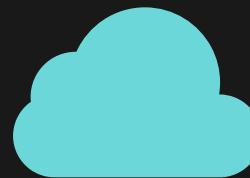
7 - HOW TO DEPLOY

- We need to configure the MS tools
- Once published, the clients can immediately call the function





Serverless - Azure Sample



7 - DEPLOY

The screenshot shows the Azure portal interface for deploying an Azure Function app. At the top, there's a navigation bar with a cloud icon, the text 'AzureOnTheFly - Zip Deploy.pubxml ▾', and a 'Publicar' button. A large green arrow points from the navigation bar towards the 'Publicar' button. Below this, a message box displays a green checkmark and the text 'Publicação com êxito em 05/10/2022 às 17:12'. There are 'Novo' and 'Mais ações ▾' buttons below the message. The main content area has a dark background with light-colored text. It shows 'Configurações' with 'Configuração' and 'Release' buttons, and 'Tempo de Execução de Destino' with 'Portátil' and edit icons. A link 'Mostrar todas as configurações' is also present. The bottom section is titled 'Hospedagem' and lists 'Assinatura' (redacted), 'Grupo de recursos' (NovidadesERP20221006115124ResourceGroup), 'Nome do recurso' (AzureOnTheFly), 'Nome de usuário' (redacted), and 'Senha' (*****). A 'Site:' link with a copy icon is at the bottom.

AzureOnTheFly - Zip Deploy.pubxml ▾

Aplicativo de Funções do Azure (Windows)

Publicar

+ Novo Mais ações ▾

✓ Publicação com êxito em 05/10/2022 às 17:12.

Abrir o site

Configurações

Configuração Release

Tempo de Execução de Destino Portátil

Mostrar todas as configurações

Hospedagem

Assinatura [redacted]

Grupo de recursos NovidadesERP20221006115124ResourceGroup

Nome do recurso AzureOnTheFly

Nome de usuário [redacted]

Senha *****

Site: <https://azureonthefly.azurewebsites.net>



Serverless - Azure Sample



8 - MONITORING THE FUNCTIONS DASHBOARD

Home > AzureOnTheFly > AzureOnTheFly - Application Insights >

AzureOnTheFly Application Insights

Search Application Dashboard Getting started Search Logs Monitor resource group Feedback Favorites Rename Delete

Overview Activity log Access control (IAM) Tags Diagnose and solve problems

Investigate Application map Smart detection Live metrics Transaction search Availability Failures Performance Troubleshooting guides (preview)

Monitoring Alerts

Essentials

Resource group (move) : NovidadesERP20221006115124ResourceGroup
Location : Brazil South
Subscription (move) : Azure subscription 1
Subscription ID : 7a1adf12-55dc-450b-835f-123d27d44a70
Tags (edit) : Click here to add tags

Instrumentation Key : XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX
Connection String : InstrumentationKey=XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX
Workspace : DefaultWorkspace-7a1adf12-99de-430b-893f-123d27d44a70

Show data for last: 30 minutes 1 hour 6 hours 12 hours 1 day 3 days 7 days 30 days

Failed requests

Failed requests (Count) azureonthefly

0

3:20 PM 3:25 PM 3:30 PM 3:35 PM 3:40 PM UTC-03:00 Oct 24

Server response time

Server response time (Avg) azureonthefly

8.04 ms

25ms
20ms
15ms
10ms
5ms
0ms

3:20 PM 3:25 PM 3:30 PM 3:35 PM 3:40 PM UTC-03:00 Oct 24

Server requests

Server requests (Count) azureonthefly

6

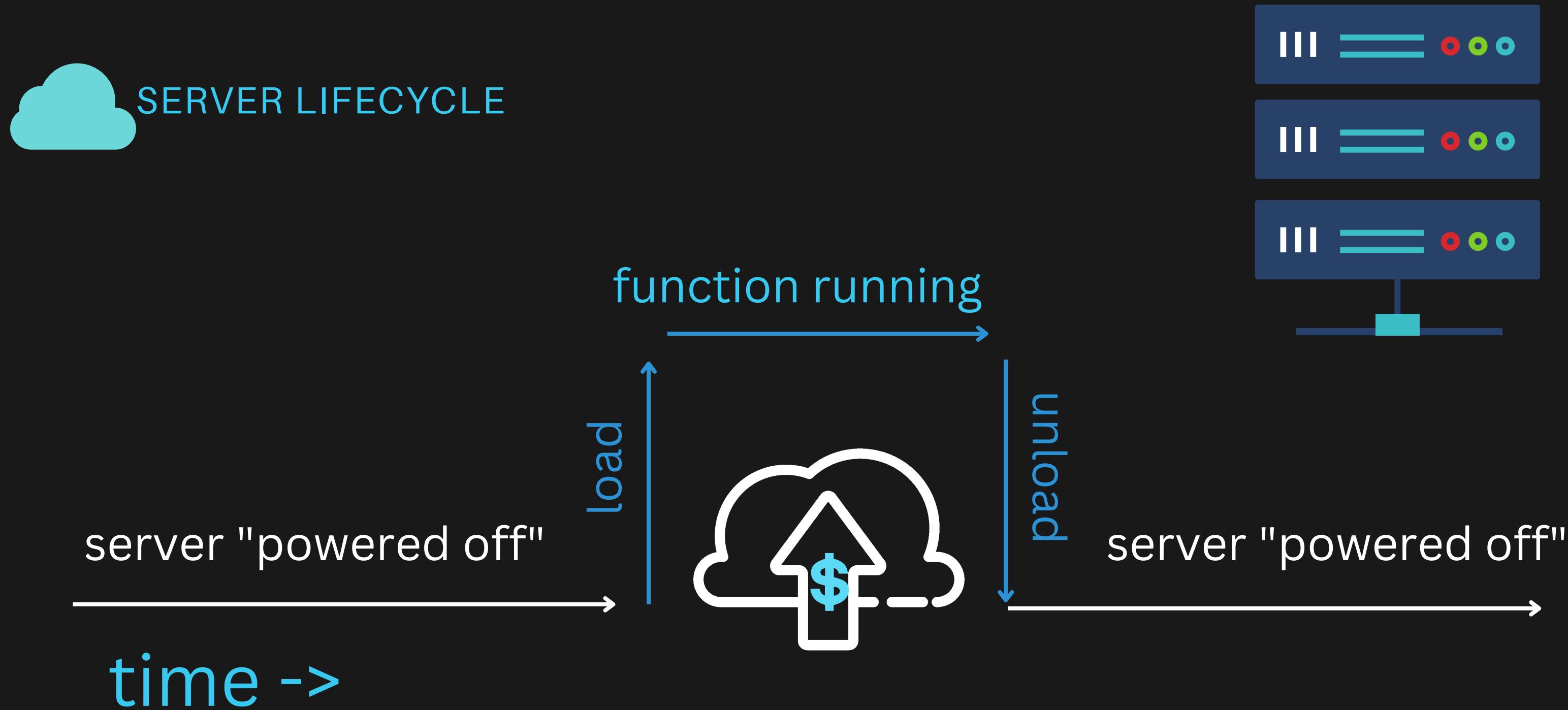
3
2
1
0

3:20 PM 3:25 PM 3:30 PM 3:35 PM 3:40 PM UTC-03:00 Oct 24

Diagram annotations: A blue arrow points from the 'Instrumentation Key' field in the Essentials section to the 'InstrumentationKey' value in the workspace details. Another blue arrow points from the 'Instrumentation Key' field to the 'InstrumentationKey' value in the connection string.

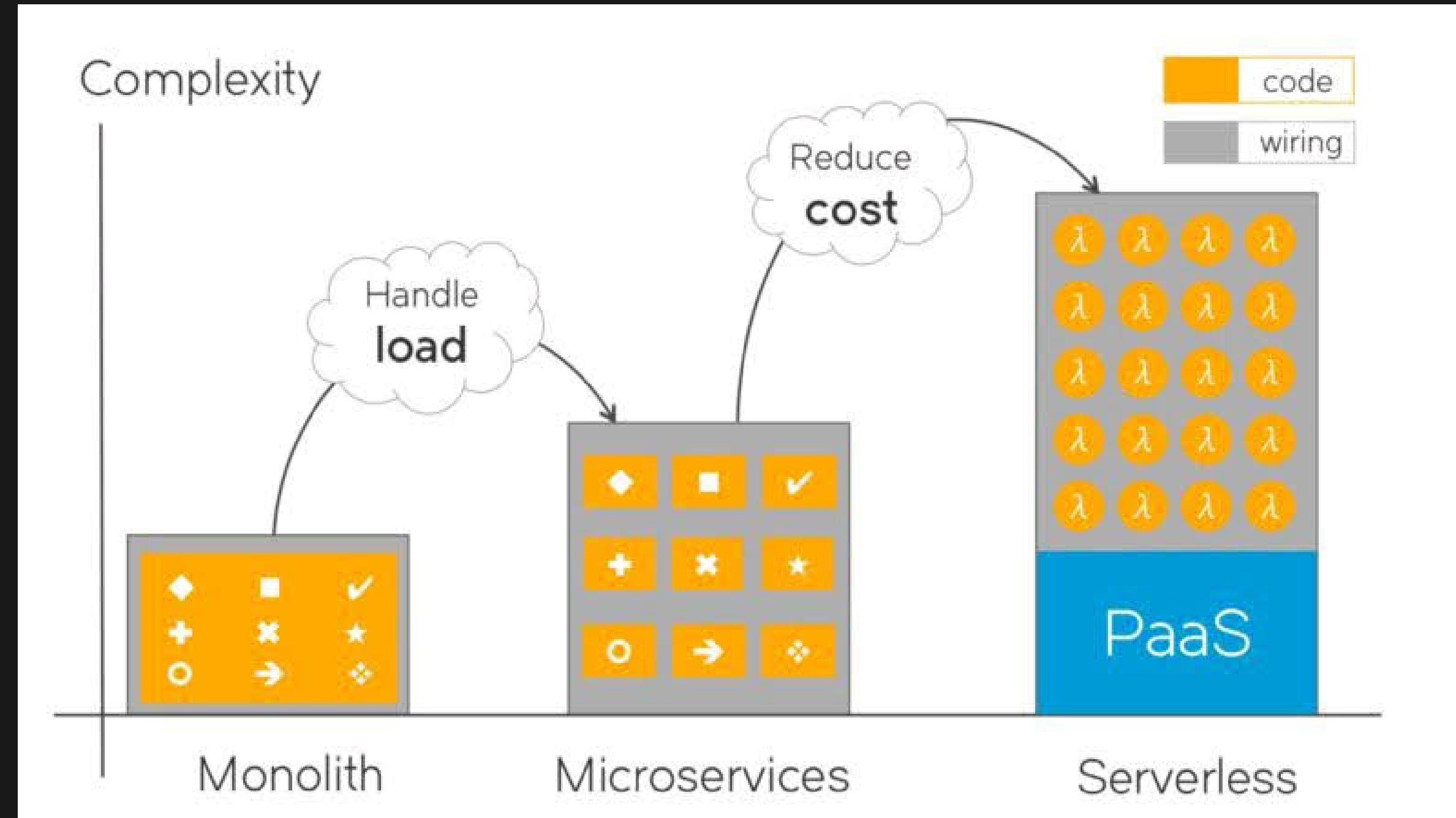


Serverless - A small recap





Serverless - Architecture





Serverless - Architecture



Questions / Challenges on that Journey

From Monolith



**Where to start?
What to break out?
Dependencies?**

To Services



**Works as expected?
Users happy?
Does it scale?
Does it perform?**

To Functions



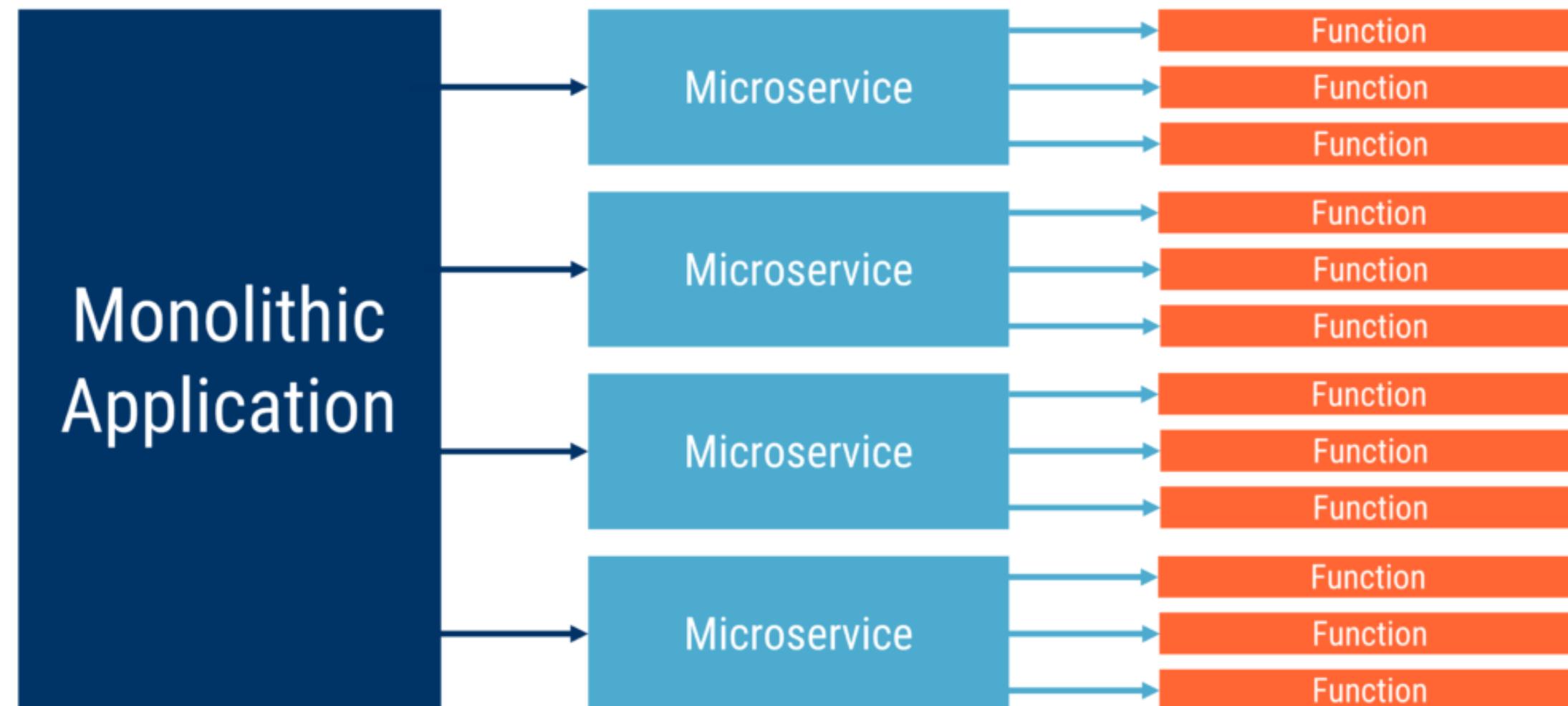
**Works as expected?
Users happy?
How to optimize?
How to automate?**



Serverless - Architecture

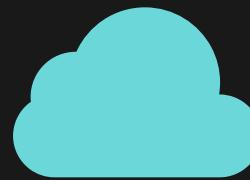


Functions provide further application granularity





Serverless - Advantages of using

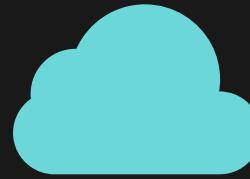


THE BENEFITS

- Cost
- Speed to "go live"
- Monitoring Tools
- Increase in source reuse



Serverless - Disadvantages

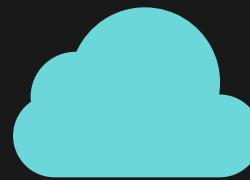


THE DOWNSIDE

- Increase in complexity
- Full dependency on a cloud provider
- Data lock in case of absence of payment
- Mistakes in source or even malicius calls can generate great costs
- Change of cloud provider could demand changes in source



Serverless - Some differences in providers



DIFFERENCES

- Cost
- Access to native languages
- Environment to run the functions
- Need of extra libs to access
- Amount of free calls monthly
- Additional tools



Serverless - Related tools

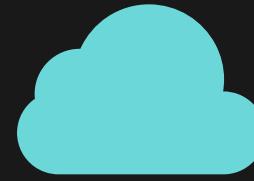


OTHER TOOLS FROM CLOUD

- Message services
- Logs
- "Functions Apps"/"Logical Apps"
- Application Insight
- Monitors



Serverless - Links



SOME TUTORIALS

From Embarcadero

- <https://www.youtube.com/watch?v=NihVuXeW5Qo>

Custom Handlers(Delphi)

- <https://learn.microsoft.com/en-us/azure/azure-functions/create-first-function-vs-code-other?tabs=go%2Cwindows>

Native functions

- <https://learn.microsoft.com/en-us/azure/azure-functions/functions-create-your-first-function-visual-studio?source=recommendations&tabs=in-process>

「 Thank You! 」

All samples on GitHub:

