

OSS (Open Source Software) | Specialist

“Our mission is to empower every person and every organization on the planet to achieve more.”

© 2021 Microsoft Corporation. All rights reserved. Last modified: October 1, 2021

This training package content is proprietary and confidential and is intended only for users described in the training materials. Some elements of this document are subject to change. This document is for informational purposes only. MICROSOFT MAKES NO WARRANTIES, EXPRESS OR IMPLIED, IN THIS SUMMARY.

This content and information is provided to you under a Non-Disclosure Agreement and cannot be distributed. Copying or disclosing all or any portion of the content and/or information included in this package is strictly prohibited. Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in, or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

The example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted herein are fictitious. No association with any real company, organization, product, domain name, e-mail address, logo, person, place, or event is intended or should be inferred.

The names of actual companies and products mentioned herein may be the trademarks of their respective owners.

Microsoft and the Microsoft products and services listed are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

Participant Guide

Table of contents

[About this module 3](#_Toc1550795665)

[Node.js Fundamentals 4](#_Toc1286502732)

[Overview 5](#_Toc446972293)

[Node.js Introduction 5](#_Toc71400820)

[Node.js Architecture 5](#_Toc801583876)

[Event Loop and Thread Pool 7](#_Toc383736843)

[Most Common IDEs 8](#_Toc600994987)

[Different ways to install Node.js 8](#_Toc1428567521)

[Hands On: Install Azure CLI, Visual Code and Node.js 9](#_Toc2136323992)

[Package Managers 9](#_Toc1208073605)

[Hands On: Node.js/JavaScript Frameworks 10](#_Toc1605737340)

[Dockerize a Node.js application 12](#_Toc2049799870)

[Hands On: Dockerize a Node.js application 12](#_Toc1109879414)

[Connect Databases 13](#_Toc873647104)

[Hands On: Connecting a REST API with MongoDB 14](#_Toc1017321080)

[Node.js on Linux 14](#_Toc502365426)

[Overview 15](#_Toc277135104)

[Create Node.js Apps 15](#_Toc793228524)

[Deploy Node.js Apps 19](#_Toc1326234215)

[Hands On: GitHub with Oryx (Kudu Build Service) 21](#_Toc1560021359)

[Hands On: GitHub Actions 22](#_Toc1783527788)

[Hands On: Local Git with Oryx (Kudu Build Service) 23](#_Toc265973216)

[Hands On: ZipDeploy with Oryx 23](#_Toc1265729617)

[Run from Package 24](#_Toc1067373778)

[Hands On: Using Run From Package with Oryx 24](#_Toc312030561)

[Hands On: Deploying with Azure DevOps pipelines 25](#_Toc574249116)

[Hands On: Deploy and Troubleshoot Next with TypeScript app 26](#_Toc1902463291)

[Hands On: Deploy an Express app that connects to PostgreSQL database 26](#_Toc1951912163)

[Application Logging 26](#_Toc1855350421)

[Node.js Memory Management 28](#_Toc745291124)

[Hands On: Hands On: Profiling Samples 35](#_Toc128987977)

[Installing and Using PM2 35](#_Toc513827350)

[Overview 36](#_Toc1829893915)

[Installation 36](#_Toc163172334)

[Hands-on: Installing, Configurating and Monitoring PM2 40](#_Toc1905148498)

[PM2 in Azure Web Apps Linux 40](#_Toc566455446)

[Hands-on: Using PM2 to Serve Angular Web Apps 41](#_Toc1435110768)

[How to Install & Configure Nginx 41](#_Toc1473915706)

[Prerequisites 42](#_Toc2009427198)

[Overview 42](#_Toc34862983)

[Accessing Content 42](#_Toc1494169969)

[Nginx File Structure 43](#_Toc1913895825)

[Nginx Configuration 44](#_Toc288413989)

[Setting up Nginx Virtual Hosts (Server Blocks) 49](#_Toc1466194680)

[Hands-on: Installing & Setting up Nginx with new Domain and Reverse Proxy for Node.js 54](#_Toc642428621)

[Configuring Web.config for NodeJS Apps on Windows 54](#_Toc983319056)

[Overview 55](#_Toc1135183862)

[Sample Web.config 55](#_Toc960992438)

[ExpressJS 57](#_Toc1018874884)

[NestJS 57](#_Toc1257083288)

[Production Build 57](#_Toc487788571)

[IISNode.yml 60](#_Toc2059028531)

[Hands-on: Deploying Node.js Frameworks on App Service Windows 61](#_Toc178031346)

[Troubleshooting Performance Issues for NodeJS on Windows 61](#_Toc228922695)

[Overview 62](#_Toc1955125104)

[Finding Memory Leaks and CPU Usage in Azure Node.js Web App 62](#_Toc570486207)

[IISNODE Configuration 63](#_Toc1107764990)

[Scenarios and Recommendations/Troubleshooting 65](#_Toc1676861772)

[Hands-on: Profiling samples - App Service Windows 66](#_Toc1640948852)

# About this module

|  |  |  |  |
| --- | --- | --- | --- |
| A close up of a logo  Description automatically generated | This participant guide was created for the Azure App Service Support team as Support Engineers or other roles. | | |
| A picture containing drawing  Description automatically generated | It will take approximately 16 hours to review content. | | |
| A picture containing drawing, monitor  Description automatically generated | This module provides information for how to create, configure, deploy, and troubleshoot Open Source technologies supported by OSS Support Engineers. | | |
|  | | | |
|  | Icons are used throughout this guide to direct you to types of information: | | |
| A picture containing drawing  Description automatically generated | The time a lesson or activity will take. | A picture containing drawing, window  Description automatically generated | An individual activity. |
| A picture containing drawing  Description automatically generated | Resource lookup information. | A picture containing drawing  Description automatically generated | An activity for partners. |
| A picture containing drawing, light  Description automatically generated | Additional note information. | A picture containing drawing  Description automatically generated | A group activity. |
|  | | | |

# Node.js Fundamentals

|  |  |
| --- | --- |
|  |  |
| A picture containing drawing  Description automatically generated | The time to complete this lesson, including exercises, is 300 minutes. |
|  |  |
| Imagen que contiene imágenes prediseñadas  Descripción generada automáticamente | After this lesson, you will be able to:   * Understand what the Node.js JavaScript runtime is. * What components are part of the Node.js Architecture and how the Event Loop works. * How to install Node.js in Windows and Linux * Most Common IDEs used for Node.js * Different options for package managers and how to install dependencies. * Paths and Files used in Node.js * Compare different frameworks, template engines and tools. * How to configure local dev web servers for node.js * How to dockerize your nodejs application. |
|  |  |

### Overview

This lesson introduces you to the Fundamentals of Node.js, understanding the architecture and how node.js works, the different package managers and how to implement different frameworks.

### Node.js Introduction

Node.js is an open-source, cross-platform, JavaScript environment that executes JavaScript code outside of your typical browser and it is built on Chrome’s V8 JavaScript engine developed by Google.

|  |  |
| --- | --- |
| Imagen que contiene dibujo  Descripción generada automáticamente | **Resource Lookup Note:**  For information related to *What Node.js is*, review:  <https://supportability.visualstudio.com/AzureAppService/_wiki/wikis/AzureAppService/540669/Azure-App-Service-Linux-OSS-Node-Development-Fundamentals?anchor=what-is-node.js%3F> |

### Node.js Architecture

Node.js handles concurrent requests with Single-Threaded model and it is built on Chrome's V8 JavaScript engine. Its main components are displayed in the image below:

Graphical user interface, text, application, chat or text message

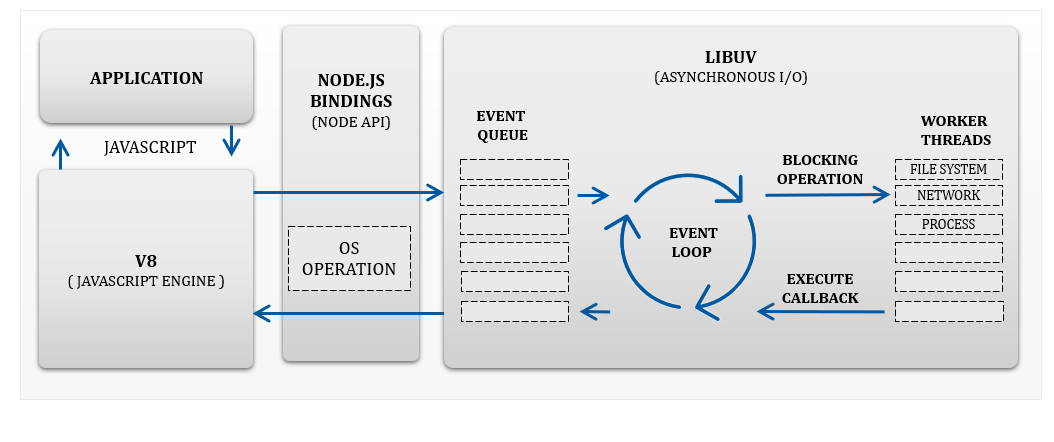
Description automatically generated

* ***NodeJS App***: Refers to your nodejs application.
* [***Node.js API***](https://nodejs.org/api/) : Exposed JavaScript API to be used by applications.
* [***Node.js Standard Library - "Internal Core Modules"***](https://github.com/nodejs/node/tree/master/lib/internal) : Consists of libraries operating system related functions for Timers setTimeout, File System fs, Network Calls http.
* [*V8 JavaScript Engine*](https://github.com/nodejs/node/tree/master/deps/v8): Written in C++ and handles Memory Heap, Call Stack, Garbage Collector and converts JavaScript code into machine code of given OS with just-in-time (JIT) compilation to speed up the execution.
* [*LibUv*](http://docs.libuv.org/en/v1.x/design.html): Consists of Thread Pool and handles Event Loop, Event Queue. It’s multi-platform C library focusing on asynchronous I/O operations. A Node.js application runs on single thread and the event loop also runs on the same thread. Node.js internally uses the libuv library which is responsible for handling operating system related tasks, like asynchronous I/O based operation systems, networking, concurrency, etc.
* [*NodeJS Bindings*](https://github.com/nodejs/node/tree/master/src#binding-functions): Wrapper libraries for other C++ components. Node.js provides various helpers for building JS classes in C++ and/or attaching C++ functions to the exports of a built-in module.
* [***C++ addons***](https://nodejs.org/dist/latest/docs/api/addons.html) : Addons are dynamically-linked shared objects written in C++. Addons provide an interface between JavaScript and C/C++ libraries and are implemented using Node-API, nan, or direct use of internal V8, libuv and Node.js libraries.
* ***Components/Dependencies***
  + [***llhttp***](https://github.com/nodejs/node/tree/master/deps/llhttp) : parsing HTTP request/response (Previously http-parse used)
  + [***c-ares***](https://github.com/nodejs/node/tree/master/deps/cares) : C library for async DNS request used in dns module.
  + [***open-ssl***](https://github.com/nodejs/node/tree/master/deps/openssl) : Cryptographic functions used in tls (ssl), crypto modules.
  + [***zlib***](https://github.com/nodejs/node/tree/master/deps/zlib): Interface to compress and decompress by sync, async and streaming.
  + [***npm***](https://github.com/nodejs/node/tree/master/deps/npm) : Default JavaScript package manager for node.

|  |  |
| --- | --- |
| Imagen que contiene dibujo  Descripción generada automáticamente | **Resource Lookup Note:**  For information related to Node.js Architecture, review:  <https://supportability.visualstudio.com/AzureAppService/_wiki/wikis/AzureAppService/540669/Azure-App-Service-Linux-OSS-Node-Development-Fundamentals?anchor=general-architecture> |

### Event Loop and Thread Pool

Node.js uses a small number of threads to handle many clients. In Node.js there are two types of threads: one *Event Loop* (aka the main loop, main thread, event thread, etc.), and a pool of k Workers in a *Worker Pool* (aka the thread pool). Review the diagram below to understand how it works:



The event loop is what allows Node.js to perform non-blocking I/O operations — even though JavaScript is single-threaded — by offloading operations to the system kernel whenever possible.

The Event Loop is responsible for JavaScript callbacks and non-blocking I/O, and a Worker executes tasks corresponding to C++ code that completes an asynchronous request, including blocking I/O and CPU-intensive work. Both types of threads work on no more than one activity at a time.

Request Flow - Requests arrive and are set in the event queue, then passes to an event loop, then libuv will decide if the request has non-blocking operations such as I/O polling, it will serve the request, if this request requires like computation, database, external resources, it will assign it into a thread pool, which by default is 4 threads to offload asynchronous work to.

|  |  |
| --- | --- |
| Imagen que contiene dibujo  Descripción generada automáticamente | **Resource Lookup Note:**  For information related to *Request Flow with LibUV (EventLoop and ThreadPool)*, review:  <https://supportability.visualstudio.com/AzureAppService/_wiki/wikis/AzureAppService/540669/Azure-App-Service-Linux-OSS-Node-Development-Fundamentals?anchor=request-flow---event-loop-and-threadpool> |

### Most Common IDEs

Most common Integrated Development Environment when developing Node.js applications are:

* Visual Studio Code
* Sublime Text
* Atom
* WebStorm
* Codeshare
* In General, any text editor of your preference.

### Different ways to install Node.js

|  |  |
| --- | --- |
| Type | Description |
| Official Packages | *Either downloading from source code or pre-built installers (Windows, macOS, Linux).* |
| Package Managers | *Depending in the Linux Distributions. These are maintained and supported by their respective packagers (Debian/Ubuntu, Arch Linux, FreeBSD, etc)*  *Other package managers:*   * *brew - macOS* * *n - macOS/Linux* * *nodenv/node-build - macOS/Linux* * *nvm - Windows/macOS/Linux* * *nvs - Windows/macOS/Linux* * *fnm - macOS, Windows, Linux* |

|  |  |
| --- | --- |
| Imagen que contiene dibujo  Descripción generada automáticamente | **Resource Lookup Note:**  For information related to the *different ways to install Node.js*, review the following article:  <https://supportability.visualstudio.com/AzureAppService/_wiki/wikis/AzureAppService/540669/Azure-App-Service-Linux-OSS-Node-Development-Fundamentals?anchor=installation> |

Hands On: Install Azure CLI, Visual Code and Node.js

|  |  |
| --- | --- |
|  |  |
| Imagen que contiene dibujo, luz  Descripción generada automáticamente | Individual activity. Duration: 15 minutes.  During this activity participants will practice how to install *Azure CLI*, *Visual Code* and *Node.js*.  Follow the instructions in the article below:  <https://supportability.visualstudio.com/AzureAppService/_wiki/wikis/AzureAppService?pageId=540708>  **Note**: Choose **nvm** from the list to install nodejs on Windows/Linux.  For further installation details review all different ways to install Nodejs in this article: <https://supportability.visualstudio.com/AzureAppService/_wiki/wikis/AzureAppService/540669/Azure-App-Service-Linux-OSS-Node-Development-Fundamentals?anchor=installation> |

### Package Managers

You can use different package managers in order install your node.js libraries, the most common ones are the following:

* npm
* yarn
* bower
* rush

|  |  |
| --- | --- |
| Imagen que contiene dibujo  Descripción generada automáticamente | **Resource Lookup Note:**  For information related to the *different package managers*, review the following article:  <https://supportability.visualstudio.com/AzureAppService/_wiki/wikis/AzureAppService/540669/Azure-App-Service-Linux-OSS-Node-Development-Fundamentals?anchor=package-management-and-installing-dependencies> |

* pnpm

Node.js and JavaScript Frameworks

There is an extensive list of available frameworks and tools in the market, that those can be divided in different categories.

Graphical user interface

Description automatically generated

|  |  |
| --- | --- |
| Imagen que contiene dibujo  Descripción generada automáticamente | **Resource Lookup Note:**  For information related to *Node.js frameworks*, review  <https://supportability.visualstudio.com/AzureAppService/_wiki/wikis/AzureAppService/540669/Azure-App-Service-Linux-OSS-Node-Development-Fundamentals?anchor=nodejs-frameworks> |

Hands On: Node.js/JavaScript Frameworks

|  |  |
| --- | --- |
|  |  |
| Imagen que contiene dibujo, luz  Descripción generada automáticamente | Individual activity. Duration: 90-120 minutes.  During this activity participants will practice how to create applications using different Node.js/JavaScript frameworks. Follow the instructions in the article below:  <https://supportability.visualstudio.com/AzureAppService/_wiki/wikis/AzureAppService/540669/Azure-App-Service-Linux-OSS-Node-Development-Fundamentals?anchor=hands-on-labs-with-different-frameworks> |

Http Web Servers (Dev/Production)

There are several http web servers that you can use for development or production, here are the most common ones:

|  |  |
| --- | --- |
| Type | Description |
| Development | Most of the node.js/JavaScript frameworks have already built-in webservers to run the application in development mode. The usual way to do that is using npm start or node <script>.js, in some scenarios there is an executable as meteor, sails lift, ng serve, etc. But you can implement some other web servers or tools to monitor for changes or restart the application if there are crashes in development as:   1. [*nodemon*](https://github.com/remy/nodemon)- A tool that helps develop node.js based applications by automatically restarting the node application when file changes in the directory are detected. 2. [*supervisor*](https://github.com/petruisfan/node-supervisor)- It is used to restart programs when they crash. It can also be used to restart programs when a \*.js file changes. |
| Production | Majority of the best scenarios it is to use a process manager which facilitates deployment, provides high availability, and enables you to manage the application at runtime. Here is a list of products divided by category:  *Process Managers*:   * [*PM2*](https://pm2.keymetrics.io/)- A production process manager for Node.js applications that has a built-in load balancer. PM2 enables you to keep applications alive forever, reloads them without downtime, helps you to manage application logging, monitoring, and clustering. * [*StrongLoop Process Manager (Strong-PM)*](https://strong-pm.io/)- A production process manager for Node.js applications with built-in load balancing, monitoring, and multi-host deployment. Includes a CLI to build, package, and deploy Node.js applications to a local or remote system. * [*Forever*](https://github.com/foreversd/forever) - A simple command-line interface tool to ensure that a script runs continuously (forever). Forever’s simple interface makes it ideal for running smaller deployments of Node.js apps and scripts. Similar as nodemon for Development.   *Static HTTP servers*:   * + [*http-server*](https://github.com/http-party/http-server)- It is a simple, zero-configuration command-line static HTTP server.   + [*serve*](https://github.com/vercel/serve)- Developed by Vercel, designed for static sites, and provides a neat interface for listing the directory's contents.   *General Web Servers*:   * + *IIS (iisnode or HttpPlatformHandler)*- You can host node.js applications with IIS using [*iisnode*](https://github.com/Azure/iisnode)or [*httpPlatformHandler*](https://docs.microsoft.com/en-us/iis/extensions/httpplatformhandler/httpplatformhandler-configuration-reference) handler.   + [*Nginx*](https://www.nginx.com/)- You can configure Nginx to serve static content or use it as a reverse proxy. |

|  |  |
| --- | --- |
| Imagen que contiene dibujo  Descripción generada automáticamente | **Resource Lookup Note:**  For information related to the *different Http Web Servers for Node.js applications*, review the following article:  <https://supportability.visualstudio.com/AzureAppService/_wiki/wikis/AzureAppService/540669/Azure-App-Service-Linux-OSS-Node-Development-Fundamentals?anchor=http-web-servers-for-development-and-production> |

### Dockerize a Node.js application

Docker is a layered filesystem for shipping images and allows organizations to abstract their applications away from their infrastructure. You can find steps and recommendations for a simple node.js application below in the Resource Lookup Note below.

|  |  |
| --- | --- |
| Imagen que contiene dibujo  Descripción generada automáticamente | **Resource Lookup Note:**  For information related to *how to dockerize Node.js applications*, review the following article:  <https://supportability.visualstudio.com/AzureAppService/_wiki/wikis/AzureAppService/540673/Azure-App-Service-Linux-OSS-Node-Development-Docker?anchor=dockerize-node.js-apps>  For information related to *best practices when creating Docker files for Node.js*, review the following article:  <https://supportability.visualstudio.com/AzureAppService/_wiki/wikis/AzureAppService/540673/Azure-App-Service-Linux-OSS-Node-Development-Docker?anchor=docker-best-practices-for-node.js-applications> |

Hands On: Dockerize a Node.js application

|  |  |
| --- | --- |
|  |  |
| Imagen que contiene dibujo, luz  Descripción generada automáticamente | Individual activity. Duration: 20 minutes.  During this activity participants will practice how to create a Dockerfile and use best practices for containerizing Node.js applications. Follow the instructions in the article below:  <https://supportability.visualstudio.com/AzureAppService/_wiki/wikis/AzureAppService/540673/Azure-App-Service-Linux-OSS-Node-Development-Docker?anchor=hands-on-labs-dockerize-node.js-applications> |

### Connect Databases

Node.js supports all kinds of databases, including *Relational* or *NoSQL Databases*. The following table list the *most common* Relational or NoSQL Databases and how to install them:

|  |  |  |  |
| --- | --- | --- | --- |
| Kind | Type | Driver | Installation |
| Relational | MS SQL Server | mssql (Tedious/msnodesqlv8) | npm install mssql |
| Relational | MySQL | mysql | npm install mysql |
| Relational | MariaDB | mariadb | npm install mariadb |
| Relational | Oracle | oracledb | npm install oracledb |
| Relational | PostgreSQL | pg | npm install pg |
| Relational | SQLite | node-sqlite3 | npm install node-sqlite |
| NoSQL | MongoDB | mongodb | npm install mongodb |
| NoSQL | CosmosDB | cosmos | npm install @azure/cosmos |
| NoSQL | Cassandra | cassandra-driver | npm install cassandra-driver |
| NoSQL | Redis | redis | npm install redis |
| NoSQL | Apache CouchDB | nano | npm install nano |
| NoSQL | RethinkDB | rethinkdb | npm install rethinkdb |
| NoSQL | OrientDB | orientjs | npm install orientjs |
| NoSQL | MarkLogic | marklogic | npm install marklogic |
| NoSQL | ArangoDB | arangojs | npm install arangojs@5 |

#### Installation Drivers

|  |  |
| --- | --- |
| Imagen que contiene dibujo  Descripción generada automáticamente | **Resource Lookup Note:**  For information about *Installing Drivers*, review  <https://supportability.visualstudio.com/AzureAppService/_wiki/wikis/AzureAppService/507667/Azure-App-Service-Linux-OSS-Node-Data-Sources?anchor=installing-drivers> |

Proceed with the following instructions to install a database driver for Node.js

1. Use the command npm install driver@version.
2. Import the module in node as followed var driver = require('driver');

Hands On: Connecting a REST API with MongoDB

|  |  |
| --- | --- |
|  |  |
| Imagen que contiene dibujo, luz  Descripción generada automáticamente | Individual activity. Duration: 20 minutes.  During this activity participants will practice how to connect a REST API with a local MongoDB.  Proceed with the steps in the article below:  <https://supportability.visualstudio.com/AzureAppService/_wiki/wikis/AzureAppService/507667/Azure-App-Service-Linux-OSS-Node-Data-Sources?anchor=hands-on-labs---connecting-a-rest-api-with-mongodb> |

# Node.js on Linux

|  |  |
| --- | --- |
|  |  |
| A picture containing drawing  Description automatically generated | The time to complete this lesson, including exercises, is 180 minutes. |
|  |  |
| Imagen que contiene imágenes prediseñadas  Descripción generada automáticamente | After this lesson, you will be able to:   * Use different tools to create and configure Node.js applications for App Service Linux. * Compare different options to enable, access and review Application Logs. * Manage issues related to Deployment, High Memory and CPU for Node.js applications running on App Service Linux. |
|  |  |

### Overview

This lesson introduces you to the *creation and deployment* of *Node.js applications* to Azure and provides information about troubleshooting issues related to *Deployment*, *High Memory* and *CPU* for *Node.js* *applications* running on *App Service Linux*.

### Create Node.js Apps

|  |  |
| --- | --- |
| Imagen que contiene dibujo  Descripción generada automáticamente | **Resource Lookup Note:**  For information related to *Package Manager*, review  <https://supportability.visualstudio.com/AzureAppService/_wiki/wikis/AzureAppService/619311/Azure-App-Service-Linux-OSS-Node-Deployment-Oryx-for-Node?anchor=package-manager> |

Nodejs applications can be created by using the following tools: *Azure Portal*, *Visual Code*, *Azure CLI*, or *ARM Templates*.

#### Azure Portal

Proceed with steps below:

1. Navigate to [Azure Portal](https://portal.azure.com/%23create/Microsoft.WebSite).
2. Create or select an existing “*Resource Group”*.

|  |  |
| --- | --- |
| A picture containing drawing, light  Description automatically generated | **Additional Information Note:** You cannot create a *Linux Web App* in an *App Service Plan* already hosting *Non-Linux Web Apps*. Based on a current limitation, for the same *Resource Group, Windows* and *Linux* apps cannot be mixed in the same region. |

1. Fill up the field “*Name”* and select “*Code”* as publish option.
2. Select the “*Runtime stack”* considering the following Node.js versions: 12 LTS, 14 LTS.
3. Select the “*Region”* to deploy.
4. Create a new “*App Service Plan”* or select an existing one.
5. Enable “*Application Insights”* or skip this step.
6. Press on “*Review + create”*.

#### Visual Code

|  |  |
| --- | --- |
| Imagen que contiene dibujo  Descripción generada automáticamente | **Resource Lookup Note:**  For more information about *Node.js applications using Visual Code*, review  <https://docs.microsoft.com/en-us/azure/app-service/app-service-web-get-started-nodejs#clone-and-run-a-local-nodejs-application> |

Next instructions are provided to create a Node.js application for Linux:

1. In the *VS Code Activity Bar*, select the *Azure Logo* to display the “*Azure App Service”* explorer as displayed in the following image:

Captura de pantalla de un celular

Descripción generada automáticamente

1. Select *“Sign in”* to Azure and follow the instructions.
2. Review the name of your *Azure subscription* asdisplayed in the next image:

Captura de pantalla de un celular

Descripción generada automáticamente

1. In the “*Azure App Service”* explorer of *VS Code*, select the blue up arrow icon () to deploy your app to Azure. To invoke the same command from the Command Palette (Ctrl+Shift+P), type *deploy to web app* and choose *Azure App Service: Deploy to Web App*.

Captura de pantalla de un celular

Descripción generada automáticamente

1. Choose your *application folder*.
2. Choose “*Create new Web App... Advanced”*.
3. Type a globally *unique* name for your web app using *only* alphanumeric characters ('A-Z', 'a-z', and '0-9') and hyphens ('-').
4. Create “*New Resource Group”* and “*Linux App Service Plan”* or select existing ones.
5. Select an “*LTS Node.js version”* when prompted, as displayed in the image below:

Captura de pantalla de un celular

Descripción generada automáticamente

1. Select *“Yes”* when prompted with *“Update your configuration”* to run npm install on the target Linux server.
2. Select “Yes” when prompted with *“Always deploy the workspace [yourapplication] to [app name]”* to *automatically* target the same *App Service Web App* with subsequent deployments.
3. Select *Browse Website* in the prompt to view your freshly deployed web app once deployment is complete.

#### Azure CLI

|  |  |
| --- | --- |
| Imagen que contiene dibujo  Descripción generada automáticamente | **Resource Lookup Note:**  For more information about *az webapp up*, review  <https://docs.microsoft.com/en-us/cli/azure/webapp?view=azure-cli-latest#az-webapp-up>  For more information about *az webapp browse*, review  <https://docs.microsoft.com/en-us/cli/azure/webapp?view=azure-cli-latest#az-webapp-browse> |

There are two different options to create a Node.js applications once you have logged to Azure using the command az login.

For *option A*, proceed with the next step:

1. Navigate to the application folder.
2. Type and run the command az webapp up --sku F1 -n <app-name> -l <location>.

This command will recognize the stack of your app, create a default resource group, create a default *App Service Plan*, create an app with the specified name, and zip deploy files from the current working directory to the app.

For *option B*, follow the next instructions:

1. Type and run the command az web app create.
2. Set up the deployment option.
3. Review the current runtimes for *“node”* by using the command az webapp list-runtimes as displayed in the image below:

Imagen que contiene pájaro, ave, flor

Descripción generada automáticamente

#### ARM Templates

To create a web site using ARM and deploy it with *PowerShell*, *Azure CLI*, or *Azure Portal*, you will need a *template* and *parameters json files*. For using *Azure Portal*:

1. Navigate to [Create MS Template](https://ms.portal.azure.com/#create/Microsoft.Template) in *Azure Portal*.
2. Click on “*Build your own template in the editor”*: You can use an online editor or upload files to create the template. The following are samples for templates and parameters:
   1. Navigate to [GitHub](https://github.com/Azure/azure-quickstart-templates/tree/master/101-webapp-linux-node) to review a sample as a reference for the template.

|  |  |
| --- | --- |
| A picture containing drawing, light  Description automatically generated | **Additional Information Note:** Take care for this setting *linuxFxVersion* and review specific *stack* is correct, otherwise it will create *Windows App*. |

1. Navigate to [Template.json](https://raw.githubusercontent.com/azureossd/arm-templates-samples/master/nodejs/template.json) to visualize another *template sample* from *GitHub*.
2. Navigate to [Parameter.json](https://raw.githubusercontent.com/azureossd/arm-templates-samples/master/nodejs/parameters.json) to visualize another *parameter sample* from *GitHub*.
3. Check *Agree Terms and Conditions* and press on *Purchase*.

### Deploy Node.js Apps

There are several *deployment options* for *Node.js* on *Azure Web App Linux* as listed below:

|  |  |
| --- | --- |
| Type | Description |
| CD *(Kudu Build Service or Azure Pipelines, GitHub Actions)* | *GitHub*, *BitBucket*, and *Azure Repos repositories* by pulling in the latest updates. |
| Local Git *(Kudu Build Service or Azure Pipelines)* | *Git repository* on your local computer. |
| ZipDeploy *(Kudu Service)* | *ZIP File* deployment through Kudu. |
| Run from Package *(Kudu Service)* | *ZIP Package File* deployment as read-only. |
| FTP | *Manual Deployment* to copy files through FTP and FTPS endpoints. |

#### Oryx for Node.js

|  |  |
| --- | --- |
| Imagen que contiene dibujo  Descripción generada automáticamente | **Resource Lookup Note:**  For information related to Oryx, review  <https://github.com/microsoft/Oryx/blob/main/doc/hosts/appservice.md>  <https://github.com/microsoft/Oryx/blob/main/doc/architecture.md>  <https://github.com/microsoft/Oryx/blob/main/doc/runtimes/nodejs.md> |

*App Service Linux* uses *Oryx* to build and install packages for *Node.js* by default. There are *three main phases* to deploy a *Node.js application*, which will be explained below:

1. *Oryx Node Detection:* It runs in KuduLite when the following conditions are met:
2. One of these files is found in the root of the repo: package.json, package-lock.json, or yarn.lock.
3. One of these files is found in the root of the repo: server.js, or app.js.
4. *Oryx Build*: It runs in KuduLite and applies the following process for each build:
5. Run custom script if specified by PRE\_BUILD\_SCRIPT\_PATH.
6. Run npm install without any flags, which includes npm preinstall and postinstall scripts and also installs devDependencies.
7. Run npm run build if a build script is specified in your package.json.
8. Run npm run build:azure if a build:azure script is specified in your package.json.
9. Run custom script if specified by POST\_BUILD\_SCRIPT\_PATH.

|  |  |
| --- | --- |
| A picture containing drawing, light  Description automatically generated | **Additional Information Note:** Scripts named prebuild and postbuild will run *before* and *after* build respectively if specified; and scripts named preinstall and postinstall will run *before* and *after* install. |

1. *Run and Startup for Node.js*: It runs in a *Node.js container*, and applies the following process to determine how to start an app:
2. Run npm start if a start script is specified.
3. Else, if a script is specified in package.json main field, run that.
4. Run the *first script* found in the root of the repo, which can be one of the following: bin/www, server.js, app.js, index.js, or hostingstart.js.

#### Node Versions

To configure & review Node version information, you can use Azure Portal, Azure CLI.

Proceed with steps below to set the Node version to 10.14:

1. Run the following command from Azure CLI: az webapp list-runtimes --linux | grep NODE, to review the current Node versions available.
2. Then run the following command to set the value: az webapp config set --resource-group <resource-group-name> --name <app-name> --linux-fx-version “NODE|10.14”
3. Display the current Runtime version.**az webapp config show --resource-group <resource-group-name> --name <app-name> --query linuxFxVersion**

Runtime Support & Patching

Depreciations will be announced & once a version reaches end of life, it will no longer be available as an option for new deployments. Existing apps will show as "Depreciated" from the Portal dropdown.

|  |  |  |
| --- | --- | --- |
| Version | Status | End |
| Node.js 9.x | EOL | 6/30/19 |
| Node.js 10.x | EOL | 4/30/21 |
| Node.js 12.x | LTS | 4/30/22 |
| Node.js 14.x | LTS | 4/30/23 |

#### Continuous Deployment (Kudu Build Service, GitHub Actions or Azure Pipelines)

|  |  |
| --- | --- |
| Imagen que contiene dibujo  Descripción generada automáticamente | **Resource Lookup Note:**  For information related to *CD with Kudu Build Service or GitHub provider*, review  <https://docs.microsoft.com/en-us/azure/app-service/deploy-continuous-deployment?tabs=github>  For information about *CD with Azure Pipelines*, review  <https://docs.microsoft.com/en-us/azure/app-service/deploy-continuous-deployment?tabs=repos>  For information about *CD with Bitbucket and Kudu Build Service*, review  <https://docs.microsoft.com/en-us/azure/app-service/deploy-continuous-deployment?tabs=bitbucket> |

You can enable *Continuous Deployment (CD)* from *GitHub*, *Bitbucket* and *Azure Repos* repositories and *automatically* get the latest updates by following the next instructions:

1. Ensure that your *repository root* has the *correct* files in your project, such as: server.js, app.js, or package.json with a start script.
2. If you are deploying for the first time, select Authorize and follow the authorization prompts. If you want to deploy from a different user's repository, select Change Account.
3. Select a build server, you have the following options (it varies from the Source you select):

a) *Kudu App Service build server*. Kudu Lite uses Oryx as a build system to detect, install and deploy your app. Steps can be found on this Doc

b) *Azure Pipelines*. DevOps will use a pipeline with agent to build and deploy your app. Steps can be found on this Doc

c) *GitHub Actions*. When GitHub Actions is the chosen build provider, you can select the workflow file you want with the Runtime stack and Version dropdowns. Azure commits this workflow file into your selected GitHub repository to handle build and deploy tasks.

Hands On: GitHub with Oryx (Kudu Build Service)

|  |  |
| --- | --- |
|  |  |
| A picture containing drawing, window  Description automatically generated | Individual activity. Duration: 15 minutes.  During this activity participants will practice how to deploy a *Node.js App* to *Azure Web App* from a forked repository on *GitHub*.  Proceed with the steps from the article below:  <https://supportability.visualstudio.com/AzureAppService/_wiki/wikis/AzureAppService/579503/Azure-App-Service-Linux-OSS-Node-Deployment-GitHub-Actions?anchor=github-with-oryx> |

#### Troubleshooting Guidelines

|  |  |
| --- | --- |
| Imagen que contiene dibujo  Descripción generada automáticamente | **Resource Lookup Note:**  For information related to *Troubleshooting CD with Oryx*, review  <https://supportability.visualstudio.com/AzureAppService/_wiki/wikis/AzureAppService/579503/Azure-App-Service-Linux-OSS-Node-Deployment-GitHub-Actions?anchor=troubleshooting-continuous-deployment-with-oryx-build> |

Hands On: GitHub Actions

|  |  |
| --- | --- |
|  |  |
| Imagen que contiene dibujo, luz  Descripción generada automáticamente | Individual activity. Duration: 15 minutes.  During this activity participants will practice how to configure and deploy a *Node.js* application with GitHub Actions.  Follow the steps in the article below:  <https://supportability.visualstudio.com/AzureAppService/_wiki/wikis/AzureAppService/579503/Azure-App-Service-Linux-OSS-Node-Deployment-GitHub-Actions?anchor=set-up-github-actions-from-the-portal> |

#### Troubleshooting Guidelines

|  |  |
| --- | --- |
| Imagen que contiene dibujo  Descripción generada automáticamente | **Resource Lookup Note:**  For information related to *Troubleshooting GitHub Actions deployments*, review  <https://supportability.visualstudio.com/AzureAppService/_wiki/wikis/AzureAppService/579503/Azure-App-Service-Linux-OSS-Node-Deployment-GitHub-Actions?anchor=troubleshooting-continuous-deployment-with-oryx-build>  For information related to *Troubleshooting Node.js Frameworks*, review  <https://supportability.visualstudio.com/AzureAppService/_wiki/wikis/AzureAppService/540665/Azure-App-Service-Linux-OSS-Node-Deployment-Frameworks> |

#### Local Git (Kudu Build Service)

|  |  |
| --- | --- |
| Imagen que contiene dibujo  Descripción generada automáticamente | **Resource Lookup Note:**  For information related to *Configure Deployment Credential*, review  <https://docs.microsoft.com/en-us/azure/app-service/deploy-configure-credentials>  For information about *Deploy with Azure CLI*, review  <https://docs.microsoft.com/en-us/azure/app-service/deploy-local-git#deploy-with-kudu-build-server> |

You can enable *Local Git* and keep your web app code stored in a local git repository, then deploy it to Azure by pushing to a remote repository. Ensure that your *repository root* has the *correct* files in your project, such as: *server.js*, *app.js*, or *package.json* with a start script.

Hands On: Local Git with Oryx (Kudu Build Service)

|  |  |
| --- | --- |
|  |  |
| A picture containing drawing, window  Description automatically generated | Individual activity. Duration: 15 minutes.  During this activity participants will practice how to create a site and deploy it using local git using *Azure Portal* and *Git Commands*.  Proceed with the steps from the article below:  <https://supportability.visualstudio.com/AzureAppService/_wiki/wikis/AzureAppService/619309/Azure-App-Service-Linux-OSS-Node-Deployment-Local-Git?anchor=hands-on-labs---deploy-a-git-repository-with-local-git-and-oryx> |

#### Troubleshooting Guidelines

|  |  |
| --- | --- |
| Imagen que contiene dibujo  Descripción generada automáticamente | **Resource Lookup Note:**  For information about *Troubleshooting* *Local Git*, review  <https://supportability.visualstudio.com/AzureAppService/_wiki/wikis/AzureAppService/619309/Azure-App-Service-Linux-OSS-Node-Deployment-Local-Git?anchor=troubleshooting-local-git-with-oryx-build> |

Hands On: ZipDeploy with Oryx

|  |  |
| --- | --- |
|  |  |
| A picture containing drawing, window  Description automatically generated | Individual activity. Duration: 20 minutes.  During this activity participants will practice how to deploy a *Node.js* application without the node\_modules included and have *Oryx* build the packages for you.  Proceed with the steps in the article below:  <https://supportability.visualstudio.com/AzureAppService/_wiki/wikis/AzureAppService/619307/Azure-App-Service-Linux-OSS-Node-Deployment-Zip-Deploy-with-Oryx?anchor=hands-on-labs---using-zipdeploy-with-oryx-build> |

#### Troubleshooting Guidelines

|  |  |
| --- | --- |
| Imagen que contiene dibujo  Descripción generada automáticamente | **Resource Lookup Note:**  For information related to *Troubleshooting ZipDeploy with Oryx*, review  <https://supportability.visualstudio.com/AzureAppService/_wiki/wikis/AzureAppService/619307/Azure-App-Service-Linux-OSS-Node-Deployment-Zip-Deploy-with-Oryx?anchor=troubleshooting-zipdeploy-with-oryx-build> |

### Run from Package

It refers to deploy a *Zipped App* without having to unpack the contents, and this process can be applied for Windows and Linux.

* *Windows*: The *Zipped App* is stored in D:\home\data\SitePackages.
* *Linux*: The *Zipped App* is stored at /home/data/SitePackages.

The *Zip* is then mounted to /home/site/wwwroot in read-only.

This process has many benefits such as: prevents file locking when scaled out to multiple instances, reduces cold start times as no files need to be unzipped, and you can run the file from storage.

Hands On: Using Run From Package with Oryx

|  |  |
| --- | --- |
|  |  |
| Imagen que contiene dibujo, luz  Descripción generada automáticamente | Individual activity. Duration: 15 minutes.  During this activity participants will practice how to configure and deploy a *Node.js* application that *Run from Package*.  Follow the steps in the article below:  <https://supportability.visualstudio.com/AzureAppService/_wiki/wikis/AzureAppService/619305/Azure-App-Services-Linux-OSS-Node-Deployment-Run-From-Package?anchor=hands-on-labs---using-run-from-package-with-oryx-build> |

#### Troubleshooting Guidelines

|  |  |
| --- | --- |
| Imagen que contiene dibujo  Descripción generada automáticamente | **Resource Lookup Note:**  For information related to *Troubleshooting Run From Package*, review  <https://supportability.visualstudio.com/AzureAppService/_wiki/wikis/AzureAppService/619305/Azure-App-Services-Linux-OSS-Node-Deployment-Run-From-Package?anchor=troubleshooting-run-from-package>  For information related to *Troubleshooting Node.js Frameworks*, review  <https://supportability.visualstudio.com/AzureAppService/_wiki/wikis/AzureAppService/540665/Azure-App-Service-Linux-OSS-Node-Deployment-Frameworks> |

Hands On: Deploying with Azure DevOps pipelines

|  |  |
| --- | --- |
|  |  |
| Imagen que contiene dibujo, luz  Descripción generada automáticamente | Individual activity. Duration: 15 minutes.  During this activity participants will practice how to configure and deploy a *Node.js* application that *deploys from Azure DevOps pipelines*.  Follow the steps in the article below:  <https://supportability.visualstudio.com/AzureAppService/_wiki/wikis/AzureAppService/621140/Azure-App-Service-Linux-OSS-Node-Deployment-Azure-DevOps-Pipelines?anchor=hands-on-labs---devops-pipelines---deploy-a-node-app> |

#### Troubleshooting Guidelines

|  |  |
| --- | --- |
| Imagen que contiene dibujo  Descripción generada automáticamente | **Resource Lookup Note:**  For information related to *Troubleshooting DevOps deployments*, review  <https://supportability.visualstudio.com/AzureAppService/_wiki/wikis/AzureAppService/621140/Azure-App-Service-Linux-OSS-Node-Deployment-Azure-DevOps-Pipelines?anchor=troubleshooting>  For information related to *Troubleshooting Node.js Frameworks*, review  <https://supportability.visualstudio.com/AzureAppService/_wiki/wikis/AzureAppService/540665/Azure-App-Service-Linux-OSS-Node-Deployment-Frameworks> |

Hands On: Deploy and Troubleshoot Next with TypeScript app

|  |  |
| --- | --- |
|  |  |
| Imagen que contiene dibujo, luz  Descripción generada automáticamente | Individual activity. Duration: 30 minutes.  During this activity participants will practice how to deploy Next with TypeScript app, and how to troubleshoot issues related to this deployment.  Follow the steps in the article below:  <https://supportability.visualstudio.com/AzureAppService/_wiki/wikis/AzureAppService/540665/Azure-App-Service-Linux-OSS-Node-Deployment-Frameworks?anchor=hands-on-labs---deploy-and-troubleshoot-next-with-typescript-app> |

Hands On: Deploy an Express app that connects to PostgreSQL database

|  |  |
| --- | --- |
|  |  |
| Imagen que contiene dibujo, luz  Descripción generada automáticamente | Individual activity. Duration: 20 minutes.  During this activity participants will practice how to deploy an Express Node.js with Azure PostgreSQL Database.  Proceed with the steps in the article below:  <https://supportability.visualstudio.com/AzureAppService/_wiki/wikis/AzureAppService/507667/Azure-App-Service-Linux-OSS-Node-Data-Sources?anchor=hands-on-labs---running-a-postgresql-sample> |

### Application Logging

Application Logging can be enabled from *Azure Portal* or *Azure CLI*.

To enable it from *Azure Portal*, proceed with following steps:

1. Navigate to your app and select *App Service Logs*.
2. In *Application logging*, select *File System*.
3. In *Quota (MB)*, specify the disk quota for the application logs.
4. In *Retention Period (Days)*, set the number of days the logs should be retained.
5. When finished, select *Save*.

To enable it from *Azure CLI*, follow steps below:

1. Install the [Azure CLI](https://docs.microsoft.com/en-us/cli/azure/install-azure-cli?view=azure-cli-latest).
2. Run the command az login.
3. Run the command az webapp log config --name <app-name> --resource-group <yourResourceGroup> --docker-container-logging filesystem.

Docker logs are enabled by default in *Azure Web App on Linux* (Blessed images and custom containers).

#### Access Logs

The current set of logs can be accessed from a *Customer Side* or *CSS Side*.

##### Customer Side

Logs are accessible from the *Azure Portal*, *Kudu Site* or *Azure CLI*:

* *Azure Portal*: By using Log Stream blade to find helpful information during live troubleshooting or debugging.
* *Kudu Site*: By proceeding with the next steps:
  1. Download the *ZIP file* in the browser requesting: **Error! Hyperlink reference not valid.**.
  2. Navigate to *Kudu Site* to browse these logs on the following url: **Error! Hyperlink reference not valid.**, copy the link for any *specific Dockerfile* and open it in a new tab to view the logs.
  3. Use *Kudu Bash/SSH* and navigate through /home/LogFiles.
* *Azure CLI*: Running the command az webapp log tail --name <app-name> --resource-group <yourResourceGroup>.

##### CSS Side

You can use *AppLens* and navigate to *Application Logs Detector*:

* If the customer has enabled the *Application Logging* previously from Azure Portal, then the application logs will be generated, otherwise you will find just the Docker Logs.
* If you are troubleshooting, you can request customer to enable *Application Logs* and refresh the detector, request the *ZIP File*, or use *Log Stream*.

|  |  |
| --- | --- |
| A picture containing drawing, light  Description automatically generated | **Additional Information Note:** It is always recommended to request the logs *Zip File* using one of the following *URLs*: Error! Hyperlink reference not valid.; or Error! Hyperlink reference not valid.. This algorithm generates a Zip File with deployment logs and LogFiles (docker/application logs). |

#### Read Logs

The *Logfiles directory* contains a list of files with logs displayed in the following format:

A picture containing knife

Description automatically generated

In the command above, \_default\_docker.log refers to your application logs, while \_docker.log refers to the overall Docker log.

Find information about each command below:

* From this log year\_month\_day\_roleinstance\_default\_docker.log you can find not only Node.js errors, but also output from PM2 and other components logging to *stdout* and *stderr* as *Oryx startup output*. If your Node.js application is sending console.log() or console.err() as examples, these messages will appear in the application logs.
* From this log year\_month\_day\_roleinstance\_docker.log you can find docker run commands as well as warmup requests to the container and if the container starts successfully or was failing during startup with timeouts or application crashes.

|  |  |
| --- | --- |
| A picture containing drawing, light  Description automatically generated | **Additional Information Note:** If using *Log Stream* or *Azure CLI*, you will find the content of both *Logs* and *Kudu Trace* or *WebSSH* output. |

### Node.js Memory Management

Review the next overview of memory management components and the processes related to troubleshooting *High Memory* and *CPU* issues for *Node.js applications* running on *App Service Linux*.

#### Memory Management Components

There are important concepts related to memory management components such as stack, heap, resident set size (RSS), working set size (WSS), virtual memory, virtual set size, garbage collector (GC).

##### Stack

It is used for *static memory* allocation and it is a Last in First Out (LIFO) stack. It stores only *finite* and *static* data, such as *local variables*, *functions frames* and *pointers* or *references to objects*. There is a *limit on the size of value* that can be stored on the stack for most languages. *Overflow errors* usually occur in the stack as the size of it is limited compared to the Heap.

##### Heap

It is used for *dynamic memory* allocation and the application or program needs to look up the data in heap using pointers. It is slower than stack, but it can store more data that the stack. It stores data with *dynamic* size such as *global variables*, *reference types* like *objects*, *strings*, *maps*, and other *complex data structures*. Heap is shared among threads of an application. *Memory errors* usually occur if the application tries to use more memory than the allocated heap.

|  |  |
| --- | --- |
| A picture containing drawing, light  Description automatically generated | **Additional Information Note:** There are older languages that do not manage memory for your application by default, it is up to you to allocate and free memory for the objects you create. If you are using *C and C++*, you can find methods (*malloc*, *realloc*, *calloc* and *free*) to manage the memory and it is up to the developer to allocate and free heap memory in the program. |

##### Resident Set Size (RSS)

It is the portion of memory occupied by a process that is held in main memory (RAM - Physically Resident Memory), excluding memory that is swapped out. It includes memory from shared libraries if the pages from those libraries are in memory, as well as the code itself, the stack, and the heap memory. The value is typically specified in bytes or pages. If the full amount of space required by a process exceeds the RSS, the remaining portion is typically stored in swap.

##### Working Set Size (WSS)

It is the amount of memory that a process requires in a time interval. If your application has populated 1 GB of main memory, but only uses 30 MBs each second to do its job, that is the working set size.

##### Virtual Memory

It is also known as virtual storage and it uses both hardware and software. It maps memory addresses used by a program called virtual addresses into physical addresses.

##### Virtual Set Size

It is the memory size assigned to a process during the initial execution. It is a number of how much memory a process has available for its execution, including memory that is in swap and all shared libraries.

##### Garbage Collector (GC)

It is the process of reclaiming the memory occupied by objects that are no longer in use by the application. It is the most common memory management in languages and the process often runs at certain intervals. An object is a candidate for GC when it is unreachable from the root node, so not referenced by the root object or any other active objects.

#### V8 Memory Management

There are some processes to consider before troubleshooting memory issues on V8 such as: Generational Heap Layout, Spaces and V8 Triggers, Heap and Stack Sizes.

##### Generational Heap Layout

It is divided in generations and ages of objects, and it uses different GC, as displayed in the following image:

Captura de pantalla de un celular

Descripción generada automáticamente

There are two *generations*:

* *Young generation:* It is also called New Space and it includes *nursery* and *intermediate*. *Nursery* refers to the space in which new objects are allocated. It is fast to collect garbage here and has a size of ~1-8 MBs. *Intermediate* refers to the space that keep the objects that were not removed from the Nursery age updating their references.
* *Old generation:* It is also called Old Space and it contains objects that were not removed from the *young generation*, and integrates multiple parts like *Old Space*, *Map Space*, *Large Object Space*.

There are three *ages of objects*:

* *Nursery age*: When you *first* allocate an object, it is going into this age.
* Intermediate age: If this object survives *one* GC, then it is moved to this age.
* *Old generation*: If this object survives *another* GC, then it is moved to this age.

There are two *GC*:

* *Minor GC (Scavenger)*: It collects garbage in the young generation, surviving objects are always evacuated to a new page.
* *Major GC (Mark-Compact)*: It collects garbage from the whole heap.

##### Garbage Collection Flow Between Spaces

The *Young Generation* is also divided in two equal-sized semi spaces (*to-space* and *from-space*). V8 uses semi-space just designed for this generation. This means that *half* of the total space is always *empty*, to allow this for evacuation step. This space is managed by *Minor GC (Scavenger) process* which consists in:

* Copying the object from the *Nursery area* called *From-Space* (*aka New Space*) to the intermediate area called *To-Space* after *one GC*. This process also updates the pointers that reference the original objects. *V8* will call *GC* when it runs *out* of space in the *new space*.
* In the *second GC*, the objects that were not cleaned in the *Young Generation* and survived are moved (evacuation-compaction) to the *Old Generation* area called *Old-Space*.

The *Old Generation* is divided into *Old-Pointer Space* and *Old-Data Space*. This space is managed up by the *Major GC (Mark-Sweep & Mark-Compact) process* which consists in:

* *Marking*: It is the process by which reachable objects are found, checking from the known object’s pointers called root set. GC will follow each pointer to the object and marks it as reachable.
* *Sweeping*: It is the process where the memory addresses of objects are not marked alive. These objects are added into a data structure called a free-list.
* *Compaction*: It is the process of moving all survived objects into other pages using the free-list and compact those.

The *Old Generation* may contains pointers to *New Space*, maps objects in the *Old Generation* are moved to *Map Space* and promoted large objects exceeding size limits of other spaces are move to *Large Object Space* which are other parts of the *Heap*.

Heap and Stack Size

The size of heap for V8 is *hard limit* and can vary between *Node.js versions* and *architecture*:

|  |  |  |
| --- | --- | --- |
| Version | Architecture | Size |
| Node.js 14,15,16 | 64 bits | 4 GB |
| Node.js 12 | 64 bits | 2 GB |
| Node.js 12/14 | 32 bits | 1 GB |
| Node.js 9,10,11 | 64 bits | 1.4 GB |

Find important commands related to memory:

|  |  |
| --- | --- |
| Command | Description |
| **node --v8-options** | Find all the variables that you can set in v8. |
| **--min\_semi\_space\_size(Initial)** | Control the initial size of the new space |
| **--max\_semi\_space\_size(Max)** | Control the maximum size of the new space |
| **--initial\_old\_space\_size(Initial)** | Control the initial size of the old space |
| **--max\_old\_space\_size(Max)** | Control the maximum size of the old space |

The default size of stack region v8 allowed to use is 984 KB.

#### Troubleshooting High Memory and CPU

Review the following steps for troubleshooting *High Memory* and *CPU* issues.

##### High Memory

|  |  |
| --- | --- |
| Imagen que contiene dibujo  Descripción generada automáticamente | **Resource Lookup Note:**  For information about *High Memory*, review  <https://supportability.visualstudio.com/AzureAppService/_wiki/wikis/AzureAppService/258599/Azure-App-Service-Linux-OSS-Node-Performance?anchor=high-memory> |

To assist customers that report issues related to high memory on the Azure Portal, site is slow, or HTTP 500 level errors, follow the instructions below:

1. Check information related to [High Memory Linux Performance](https://supportability.visualstudio.com/AzureAppService/_wiki/wikis/AzureAppService/188479/Azure-App-Service-Linux-Perf-Memory?anchor=high-memory-overview).
2. Review with *CSS internal tools* (AppLens detectors, Kusto Tables, top command) to detect high memory issues. If the application process is consuming high memory, the next step will be to use memory profilers.

##### Tracing

|  |  |
| --- | --- |
| Imagen que contiene dibujo  Descripción generada automáticamente | **Resource Lookup Note:**  For information about *Tracing Garbage Collector*, review  <https://supportability.visualstudio.com/AzureAppService/_wiki/wikis/AzureAppService/258599/Azure-App-Service-Linux-OSS-Node-Performance?anchor=tracing> |

The *Tracing Collector Garbage* is a tool used to trace the amount of memory allocated, comparing the heapUsed before and heapUsed. It displays two types of cycle entries:

* *Scavenge (new space Garbage Collection)*: It is a fast GC responsible for cleaning up small objects from the new space on the heap.
* *Mark-sweep (old space Garbage Collection)*: It is a GC event which occurs less often because it takes longer to detect which objects can be cleaned up.

##### High CPU

|  |  |
| --- | --- |
| Imagen que contiene dibujo  Descripción generada automáticamente | **Resource Lookup Note:**  For information related *High CPU*, review  <https://supportability.visualstudio.com/AzureAppService/_wiki/wikis/AzureAppService/540630/Azure-App-Service-Linux-OSS-Node-Performance-High-CPU> |

To assist customers that report issues related to high CPU on the Azure Portal, site is slow, or HTTP 500 level errors, follow the instructions below:

1. Check information related to [High CPU Linux Performance](https://supportability.visualstudio.com/AzureAppService/_wiki/wikis/AzureAppService/188477/Azure-App-Service-Linux-Perf-CPU).
2. Review with *CSS internal tools* (AppLens detectors, Kusto Tables, top command) to detect the *High CPU* and the process that is consuming most of CPU percentage. If the process is related to the application, the next step will be to use profilers.

##### Reading Profile Traces

Since Chrome uses the same JavaScript runtime (V8 engine), profiler traces can be read using DevTools for Node. To find memory and profiler tabs, follow the next steps:

1. Navigate to *Chrome Browser*.
2. Type *chrome://inspect/*.
3. Press in *Open dedicated DevTools for Node*.

###### Memory profiles

The *memory profiles* facilitate the analysis and identification of the highest percentage in Retained Size comparing as well with Shallow Size, as displayed in the image below:

Captura de pantalla de un celular

Descripción generada automáticamente

* *Shallow Size*: It indicates the size of memory that is held by the object itself. Usually, only arrays and strings can have a significant shallow size.
* *Retained Size*: It displays the size of memory that is freed once the object itself is deleted due it becomes unreachable from GC roots. Held by object implicitly.

###### CPU profiles

|  |  |
| --- | --- |
| Imagen que contiene dibujo  Descripción generada automáticamente | **Resource Lookup Note:**  For information related to *Profiles taken by --prof*, review  <https://supportability.visualstudio.com/AzureAppService/_wiki/wikis/AzureAppService/540630/Azure-App-Service-Linux-OSS-Node-Performance-High-CPU?anchor=profiles-taken-by---prof> |

The *CPU profiles* facilitate the analysis and identification the function that has high self-time because it is called many times for execution. In Dev tools this will be seen as **Profiler** or **Java Script Profiler** as displayed in the image below:

Captura de pantalla de un celular

Descripción generada automáticamente

* *Self Time:* It represents the *amount of time spent* in the function at the *current level of a call tree*.
* *Total Time*: It refers to the *self time + the amount of time* it took to execute the *code in functions* that the current level calls.

##### Debuggers

|  |  |
| --- | --- |
| Imagen que contiene dibujo  Descripción generada automáticamente | **Resource Lookup Note:**  For information about *Debuggers*, review  <https://supportability.visualstudio.com/AzureAppService/_wiki/wikis/AzureAppService/258599/Azure-App-Service-Linux-OSS-Node-Performance?anchor=debuggers> |

To debug remotely with *Azure Web App Linux*, proceed with the following instructions:

1. Check information related to [Debug Remotely](https://docs.microsoft.com/en-us/azure/app-service/containers/configure-language-nodejs#debug-remotely).

Hands On: Hands On: Profiling Samples

|  |  |
| --- | --- |
|  |  |
| Imagen que contiene dibujo, luz  Descripción generada automáticamente | Individual activity. Duration: 20 minutes.  During this activity participants will practice how to profile *Memory* and *CPU* using *v8 Profiler* or *built-in profilers.*  Proceed with the steps in the article below:  Memory:  <https://supportability.visualstudio.com/AzureAppService/_wiki/wikis/AzureAppService/540631/Azure-App-Service-Linux-OSS-Node-Performance-High-Memory?anchor=hands-on-labs---profiling-memory>  CPU:  <https://supportability.visualstudio.com/AzureAppService/_wiki/wikis/AzureAppService/540630/Azure-App-Service-Linux-OSS-Node-Performance-High-CPU?anchor=hands-on-labs---profiling-cpu> |

##### Application Performance Monitoring (APM) Tool

There are *third-party tools* that customers can also use, such as *New Relic*, *AppDynamics*, *Dynatrace*, *Elastic*, *ClinicJs*, *SignalFx*, and *Stackimpact*. Most of them are not free, but they are specific designed to gather extra information and present it in graphs for Memory and CPU.

Installing and Using PM2

|  |  |
| --- | --- |
|  |  |
| A picture containing drawing  Description automatically generated | The time to complete this lesson, including exercises, is 30 minutes. |
|  |  |
| A picture containing drawing, light  Description automatically generated | After this lesson you will be able to:   * Understand how to install and configure PM2 * How to monitor and review logs with apps running under PM2 * How to configure PM2 Azure Web Apps Linux and serve static files |

### Overview

PM2 is an advanced, production process manager for Node.js, it is known as daemon process manager that helps you manage and keep your application online. It is a built-in Load Balancer that implements auto-restarting across crashes and machine restarts.

### Installation

You can install pm2 globally with the following command:

npm install pm2 -g

Or using Yarn:

yarn global add pm2

Configuration and startup

To start a nodejs application you can use the following command:

pm2 start server.js

To start a nodejs application passing node arguments, you can use the following command:

pm2 start server.js --node-args="--max\_old\_space\_size=2048"

You can use other options to pass through the CLI as followed:

--watch Watch and Restart app when files change.

--max-memory-restart <200MB>Set memory threshold for app reload

--log <log\_path> Specify log file

--no-daemon Use to run in docker container to prevent exit.

You can also use a configuration file called Ecosystem File (ecosystem.config.js) or with yaml file (<name>.yml) to set specific parameters:

pm2 start ecosystem.config.js

Using yml file:

pm2 start process.yml

A quick example of an ecosystem file:

module.exports = {

apps : [

{

name: "myapp1",

script: "./server.js",

watch: false,

error\_file:'./error.log',

out\_file:'./output.log',

env: {

"NODE\_ENV": "development",

"PORT": 3000,

},

env\_production : {

"NODE\_ENV": "production",

"PORT": 8080,

}

}

]

}

#### Monitoring and Logs

To list the status of all applications managed by PM2, you can use the following command:

pm2 [list|ls|status]

To review logs from PM2, it will tail the last 15 lines for all process, but you can also modify this parameter, you can use the following command:

pm2 logs <app\_name|all>

Output example:

edison@edisgavm:$ pm2 logs

[TAILING] Tailing last 15 lines for [all] processes (change the value with --lines option)

/home/edison/.pm2/pm2.log last 15 lines:

PM2 | 2020-05-31T17:43:14: PM2 log: App [server:0] starting in -fork mode-

PM2 | 2020-05-31T17:43:14: PM2 log: App [server:0] online

PM2 | 2020-05-31T17:43:14: PM2 log: App [server:0] exited with code [1] via signal [SIGINT]

PM2 | 2020-05-31T17:43:14: PM2 log: App [server:0] starting in -fork mode-

PM2 | 2020-05-31T17:43:14: PM2 log: App [server:0] online

PM2 | 2020-05-31T17:43:14: PM2 log: App [server:0] exited with code [1] via signal [SIGINT]

PM2 | 2020-05-31T17:43:14: PM2 log: App [server:0] starting in -fork mode-

PM2 | 2020-05-31T17:43:14: PM2 log: App [server:0] online

PM2 | 2020-05-31T17:43:14: PM2 log: App [server:0] exited with code [1] via signal [SIGINT]

PM2 | 2020-05-31T17:43:14: PM2 log: App [server:0] starting in -fork mode-

PM2 | 2020-05-31T17:43:14: PM2 log: App [server:0] online

PM2 | 2020-05-31T17:43:14: PM2 log: App [server:0] exited with code [1] via signal [SIGINT]

PM2 | 2020-05-31T17:43:14: PM2 log: Script /home/edison/codes/nodejs/nodejs-helloworld-processversion/server.js had too many unstable restarts (16). Stopped. "errored"

PM2 | 2020-05-31T17:43:51: PM2 log: Stopping app:server id:0

PM2 | 2020-05-31T17:43:51: PM2 error: app=server id=0 does not have a pid

/home/edison/.pm2/logs/server-out.log last 15 lines:

/home/edison/.pm2/logs/server-error.log last 15 lines:

0|server | at Function.Module.\_resolveFilename (internal/modules/cjs/loader.js:625:15)

0|server | at Module.Hook.\_require.Module.require (/home/edison/.nvm/versions/node/v12.5.0/lib/node\_modules/pm2/node\_modules/require-in-the-middle/index.js:61:29)

0|server | at require (internal/modules/cjs/helpers.js:16:16)

0|server | at Object.<anonymous> (/home/edison/codes/nodejs/nodejs-helloworld-processversion/server.js:1:17)

0|server | at Module.\_compile (internal/modules/cjs/loader.js:776:30)

0|server | at Object.Module.\_extensions..js (internal/modules/cjs/loader.js:787:10)

0|server | at Module.load (internal/modules/cjs/loader.js:643:32)

0|server | at Function.Module.\_load (internal/modules/cjs/loader.js:556:12)

0|server | at Object.<anonymous> (/home/edison/.nvm/versions/node/v12.5.0/lib/node\_modules/pm2/lib/ProcessContainerFork.js:32:23)

0|server | at Module.\_compile (internal/modules/cjs/loader.js:776:30) {

0|server | code: 'MODULE\_NOT\_FOUND',

0|server | requireStack: [

0|server | '/home/edison/codes/nodejs/nodejs-helloworld-processversion/server.js'

0|server | ]

0|server | }

To have a realtime dashboard for checking CPU, Memory, Crashes, etc , you can use the following command:

pm2 monit

|  |  |
| --- | --- |
|  |  |
| A picture containing drawing  Description automatically generated | Note: Pm2 monit has some limitations using WebSSH Azure Web App Linux. |
|  |  |

A screenshot of a cell phone

Description automatically generated

Hands-on: Installing, Configurating and Monitoring PM2

|  |  |
| --- | --- |
|  |  |
| A picture containing drawing, light  Description automatically generated | Individual activity. Duration: 20 minutes.  During this activity participants will follow the steps to install, configure and monitor PM2. If students run into issues, please allow time for them to ask questions.  Please follow the steps described in the article below:  <https://supportability.visualstudio.com/AzureAppService/_wiki/wikis/AzureAppService/493414/PM2?anchor=hands-on-labs---configuring-and-monitoring-with-pm2> |
|  |  |

### PM2 in Azure Web Apps Linux

|  |  |
| --- | --- |
|  |  |
| A picture containing drawing  Description automatically generated | Note: You can review more information about PM2 in Azure Web Linux through the link below: <https://supportability.visualstudio.com/AzureAppService/_wiki/wikis/AzureAppService/493414/PM2> |
|  |  |

Hands-on: Using PM2 to Serve Angular Web Apps

|  |  |
| --- | --- |
|  |  |
| A picture containing drawing, light  Description automatically generated | Individual activity. Duration: 15 minutes.  During this activity participants will follow the steps use PM2 to serve Angular Web Apps. If students run into issues, please allow time for them to ask questions.  Please follow the steps described in the article below:  <https://supportability.visualstudio.com/AzureAppService/_wiki/wikis/AzureAppService/493414/PM2?anchor=hands-on-labs---using-pm2-to-serve-angular-web-apps> |
|  |  |

How to Install & Configure Nginx

|  |  |
| --- | --- |
|  |  |
| clock icon | The time to complete this lesson, including exercises, is 60 minutes. |
|  |  |
| target icon | After this lesson you will be able to:   * Install Nginx as a web server & understand the default file structure. * Configure Nginx with Virtual Hosts * Configured Nginx as a Reverse Proxy with Node Express |
|  |  |

|  |  |
| --- | --- |
|  |  |
| A picture containing drawing  Description automatically generated | Resource Lookup Note: For more information about Understand how to configure Nginx, review <https://supportability.visualstudio.com/AzureAppService/_wiki/wikis/AzureAppService/334046/Nginx?anchor=nginx-configuration> |
|  |  |

### Prerequisites

To prepare for this lesson you will need the following:

|  |  |
| --- | --- |
|  | Azure CLI |
|  | Ubuntu Virtual Machine with the following virtual configuration:   * SSH Access - Port 22 * Default Server Access - Port 80, 8080 & 3000 |
|  |  |

### Overview

Nginx is a widely used web server known for its versatility & efficiency. While it operates as an HTTP Web Server it can also function as a reverse proxy, mail proxy server & TCP/UDP proxy server. Similar web servers like Apache may be resource-heavy creating new processes/threads to handle each new connection. Instead, Nginx has a master process operating single-threaded & asynchronously; this process controls multiple workers to handle concurrent connections which are known to scale well under high load in a non-blocking manner.

Accessing Content

While we see that nginx is running, we can request our server IP address via the browser or curl to view the default welcome page as confirmation.

All content served by Nginx by default is located within "/var/www" which is set by the configuration in "/etc/nginx/sites-enabled/default"

A screenshot of a cell phone

Description automatically generated

Nginx File Structure

All default configuration for nginx is within "/etc/ngnix/". We'll be walking through the common sections but below is a snippet of what is expected to be seen within the default directory after installation on Ubuntu & Debian.

$ ls /etc/nginx

conf.d/

fastcgi\_params

koi-win

modules-available/

nginx.conf

scgi\_params

sites-enabled/

uwsgi\_params

fastcgi.conf

koi-utf

mime.types

modules-enabled/

proxy\_params

sites-available/

snippets/

win-utf

Structure Definitions

|  |  |
| --- | --- |
| Path | Description |
| conf.d/ | If hosting multiple websites with Nginx each would have a configuration file in this directory similar to the nginx.conf |
| fastcgi.conf/fastcgi\_params | contains parameters referenced when using FastCGI to proxy request often used with php-fpm |
| koi-win/utf/win-utf | Character map used for to convert legacy characters |
| modules-avaliable/enabled/ | Modules are used to allow nginx to perform new functions & allow additional features. When a module is installed they are first seen within "/modules-available" & once enabled symlinks are created from "/mods-enabled". |
| sites-avaliable/enabled/ | Contains the server files for new domain. All server files will be available in "/sites-available" & once enabled symlinks are created from "/sites-enabled". |
| scgi-params | Commonly configured directives when passing a request to SCGI server |
| mime.types | Links file extensions to MIME types of responses |
| proxy\_params | Commonly configured directives when Nginx is used as a reverse proxy |
| snippets/ | Contains snippets that can be referenced in configurations |
| uwsgi\_params | Commonly configured directives when proxying request with a uwsgi application. |

Nginx Configuration

Nginx.conf Overview

This is the core global configuration file responsible instructing Nginx on what to do.

The default config file is broken up into a few sections addressed as contexts. These are a hierarchy of configurations blocks to provide structure. They consist of nested directives & arguments that are specific to certain contexts.

While there is a wide array of directives available for configuration, we'll walk through the basic context types which are also known as top-level directives or blocks.

Default Nginx conf

user www-data;

worker\_processes auto;

pid /run/nginx.pid;

include /etc/nginx/modules-enabled/\*.conf;

events {

worker\_connections 768;

}

http {

###### Basic Settings

sendfile on;

tcp\_nopush on;

tcp\_nodelay on;

keepalive\_timeout 65;

types\_hash\_max\_size 2048;

include /etc/nginx/mime.types;

default\_type application/octet-stream;

###### SSL Settings

ssl\_protocols TLSv1 TLSv1.1 TLSv1.2; # Dropping SSLv3, ref: POODLE

ssl\_prefer\_server\_ciphers on;

###### Logging Settings

access\_log /var/log/nginx/access.log;

error\_log /var/log/nginx/error.log warn;

###### Gzip Settings

gzip on;

# Virtual Host Configs

include /etc/nginx/conf.d/\*.conf;

include /etc/nginx/sites-enabled/\*;

server {

# configuration of HTTP virtual server 1

location /one {

# configuration for processing URIs starting with '/one'

}

}

Main Context

Any content outside of any other contexts is considered the main as it is the only context not using curly braces. The main context is placed at the start of the configuration file & will apply any directives to the entire environment & can't be overridden in any other context level.

user www-data;

worker\_processes auto;

pid /run/nginx.pid;

include /etc/nginx/modules-enabled/\*.conf;

|  |  |
| --- | --- |
| Directive | Description |
| User | sets the default user for the webserver to use; Nginx will access anything that "www-data" can access. |
| worker\_processes | number of Nginx worker processes; auto will gauge the amount to spawn based on the system load. |
| Pid | defines the location of the Nginx process |

Events Context

Anything defined within the event context determines how the worker process will manage connections. Similar to the main context, these directives are applied at the global level. "Worker\_connections" determine how many client connections will be served per worker.

events {

worker\_connections 768;

}

HTTP Context

Any directive in relation to handling HTTP or HTTPS traffic is contained within the HTTP Context & will be inherited to all virtual hosts declared.

http {

###### Basic Settings

sendfile on;

tcp\_nopush on;

tcp\_nodelay on;

keepalive\_timeout 65;

types\_hash\_max\_size 2048;

include /etc/nginx/mime.types;

default\_type application/octet-stream;

###### SSL Settings

ssl\_protocols TLSv1 TLSv1.1 TLSv1.2; # Dropping SSLv3, ref: POODLE

ssl\_prefer\_server\_ciphers on;

###### Logging Settings

access\_log /var/log/nginx/access.log;

error\_log /var/log/nginx/error.log warn;

###### Gzip Settings

gzip on;

# Virtual Host Configs

include /etc/nginx/conf.d/\*.conf;

include /etc/nginx/sites-enabled/\*;

}

|  |  |
| --- | --- |
| Directive | Description |
| Sendfile | When on, it omits the use of the buffer when copying data for transmission. It will transfer data from a file descriptor directly to kernel space to save resources. |
| tcp\_nopush | Optimizes the amount of data sent at once. Response headers will be sent in one packet. |
| keepalive\_timeout | Seconds a client connection will stay open on the server until timeout |
| types\_hash\_max\_size | Sets size of hash table nginx references for accessing data sets. |
| include /etc/nginx/mime.types; | Maps the default allowed mime types. |
| default\_type application/octet-stream; | Defines the default mime type when serving a file. If the file extension is not known it will default to serve with the type of "octet-stream" which is binary data |
| ssl\_protocols | Used to enable the needed security protocols |
| ssl\_prefer\_server\_ciphers | Specifies the preferred client cipher to be used during encryption |
| access\_log | specifices the location of the access logs which contains client request details. |
| error\_log | specifices the location of the error logs & the logging level. Warn, error, crit, alert, and emerg are the available levels. |
| Gzip | Gzipping of responses will compress responses before sending responses to clients to reduce the size of transmitted data. |
| include /etc/nginx/conf.d/\*.conf; | Used to included the configuration of other websites hosted & enabled within nginx |

Server & Location Context

These contexts are used when hosting multiple domains on a single server Nginx server. These contexts are defined within the HTTP context & will inherit its baseline configuration.

Below we have an example of a Wordpress server & Magento server handled to separate locations where each block can specify their own custom directives.

http {

...

...

server {

listen 80;

server\_name test-wordpress.com;

root /var/www/html/wordpress;

...

location /some\_url {

# configuration for processing URIs starting with /some\_url

}

location /another\_url {

# configuration for processing URIs starting with /another\_url

}

}

server {

listen 80;

server\_name test-magento.com;

root /var/www/html/magento;

...

location /some\_url {

# configuration for processing URIs starting with /some\_url

}

location /some\_other\_url {

# configuration for processing URIs starting with /some\_other\_url

}

}

}

### Setting up Nginx Virtual Hosts (Server Blocks)

We'll walk through setting a new domain with new Server & Location context blocks. This is common if a user is hosting multiple domains from a single server similar to Apache Virtual Hosts.

By default, all static content is hosted within "/var/www" but we can create multiple directories when serving multiple domains but we'll walk through the steps of using “[testnginx.centralus.cloudapp.azure.com](http://testnginx.centralus.cloudapp.azure.com/)" as new virtual host.

In this example I’ve configured my VM to with the DNS name of “testnginx” but you will use the dns name you’ve assigned to your Azure VM.

#### Creating a Directory with Permissions

We're making a new directory in our content root with the proper permissions.

sudo mkdir -p /var/www/testnginx.centralus.cloudapp.azure.com

sudo chown -R $USER:$USER /var/www/testnginx.centralus.cloudapp.azure.com

sudo chmod -R 755 /var/www/testnginx.centralus.cloudapp.azure.com

#### Creating Content

A small static page is created using nano for our test using the command & HTML below

nano /var/www/testnginx.centralus.cloudapp.azure.com/index.html

<html>

<head>

<title>Hello Test!</title>

</head>

<body>

<h1>Hello Test.com Server!</h1>

</body>

</html>

#### Creating & Enabling the Server Block

With our directory & static content created, we'll now make our new config file as seen below.

sudo nano /etc/nginx/sites-available/testnginx.centralus.cloudapp.azure.com

server {

listen 80;

listen [::]:80;

root /var/www/testnginx.centralus.cloudapp.azure.com;

index index.html index.htm index.nginx-debian.html;

server\_name testnginx.centralus.cloudapp.azure.com www.testnginx.centralus.cloudapp.azure.com;

location / {

try\_files $uri $uri/ =404;

}

}

Once saved, we'll create our symlink within site-enabled.

sudo ln -s /etc/nginx/sites-available/testnginx.centralus.cloudapp.azure.com /etc/nginx/sites-enabled/

#### Testing the Domain

Once saved we'll run "nginx -t" to test our configuration updates. Once it's returned as successful, we'll restart Nginx to enable our changes.

sudo nginx -t

sudo nginx systemctl restart nginx

We can now visit our domain & confirm our request with our custom domain!

A screenshot of a cell phone

Description automatically generated

Setting Up Nginx as a Reverse Proxy for Node.js

We'll walk through setting a new domain with new Server & Location context blocks. This is common if a user is hosting multiple domains from a single server like Apache Virtual Hosts.

|  |  |
| --- | --- |
|  |  |
| A picture containing drawing  Description automatically generated | Resource Lookup Note: For more information about Understand how to configure Nginx, review  [https://supportability.visualstudio.com/AzureAppService/\_wiki/wikis/AzureAppService/334046/Nginx?anchor=\*\*setting-up-nginx-as-a-reverse-proxy-for-node-js\*\*](https://supportability.visualstudio.com/AzureAppService/_wiki/wikis/AzureAppService/334046/Nginx?anchor=**setting-up-nginx-as-a-reverse-proxy-for-node-js**)  <https://supportability.visualstudio.com/AzureAppService/_wiki/wikis/AzureAppService/334046/Nginx?anchor=nginx-reverse-proxy> |
|  |  |

The purpose of a reverse proxy is to act as a middleman on behalf of your backend server. They are positioned as a front end for your web servers to efficiently & safely handle client requests since it has the ability to load balance, cache & filter all traffic without having to directly expose the backend server.

Reverse proxies are even seen built into the Azure Service Fabric & App Service Infrastructure but we'll walk through a simple example with of proxying to a Node Express app listening on port 3000.

A screenshot of a cell phone

Description automatically generated

#### Installing Nodejs & NPM

Node.JS is a lightweight open-source server & allows one to use JavaScript & leverage frameworks on the server. This is used the Node Package Manager(npm) to install packages, dependencies & setup projects.

We'll install nodejs & npm with the following commands.

sudo apt update

sudo apt install nodejs

sudo apt install npm

#### Creating an Express App

First, we create our base project directory for our express installation.

mkdir express

cd express

Now we'll initialize our project & install express

npm init

npm install --save express

An app.js file will be created with the following **code below** which will print "Hello World" at the root while listening on port 3000.

sudo nano app.js

Code:

const express = require('express')  
const app = express()

app.get('/', (req, res) => res.send('Hello World!'))  
app.listen(3000, () => console.log('Node.js app listening on port 3000.'))

#### Running Express App

We can now run our app & visit port 3000 to confirm that it is running.

node app.js

A screenshot of a cell phone

Description automatically generated

#### Configuring Nginx

We'll now create a new configuration from within "/etc/nginx/conf.d/" & configure it with the following content below. The config file sets the server to listen for requests on port 80 at our apps domain & proxy the requests to localhost:3000.

The proxy\_pass directive within the Location block is what defines this as a reverse proxy. This directive specifies request matching the "/" root path to be forward to our Node.js app running at localhost port 3000.

Commands:

cd /etc/nginx/conf.d

sudo nano node.conf

Config:

server {

listen 80;

listen [::]:80;

server\_name testnginx.centralus.cloudapp.azure.com;

location / {

proxy\_pass http://localhost:3000/;

}

}

#### Testing the Reverse Proxy

We'll test our configurations & reload Nginx to update our changes!

sudo nginx -t

sudo nginx -s reload

|  |  |
| --- | --- |
|  |  |
| A picture containing drawing  Description automatically generated | Note: A warning may be present if the server name is conflicting with a previous config.  Example: nginx: [warn] conflicting server name "hostname.com " on [::]:80, ignored |
|  |  |

Below we see that the response of "Hello World" from localhost:3000 & the server\_name we specified in the "node.conf" while our default config set within /site-enabled/ is still handling request at localhost on port 80.

A screenshot of a cell phone

Description automatically generated

Hands-on: Installing & Setting up Nginx with new Domain and Reverse Proxy for Node.js

|  |  |
| --- | --- |
|  |  |
| A picture containing drawing, light  Description automatically generated | Individual activity. Duration 30 minutes.  Learn how to install and configure Nginx into Linux VM, adding new domain and using nginx as a reverse proxy for an express node.js application.  Follow the instructions in the article below:  <https://supportability.visualstudio.com/AzureAppService/_wiki/wikis/AzureAppService/334046/Nginx?anchor=hands-on-labs---setting-up-nginx-with-new-domain-and-reverse-proxy-for-node.js> |
|  |  |

Configuring Web.config for NodeJS Apps on Windows

|  |  |
| --- | --- |
|  |  |
| clock icon | The time to complete this lesson, including exercises, is 30 minutes. |
|  |  |
| target icon | After this lesson you will be able to:   * Configure Web.config for NodeJS Apps on Windows. * Understand how iisnode works to host node.js applications within IIS. |
|  |  |

|  |  |
| --- | --- |
|  |  |
| eye icon | Resource Lookup Note: For more information about Configuring Web.config for NodeJS Apps on Windows, review: <https://supportability.visualstudio.com/AzureAppService/_wiki/wikis/AzureAppService/336738/Azure-App-Service-OSS-Nodejs-Configuration?anchor=web.config> |
|  |  |

### Overview

When deploying a Node app via git, Kudu automatically generates a web.config file which sets up everything for iisnode.

If you ever need to use a custom web.config, you can start with what would normally be generated. The content below assumes that your start file is called server.js, so you may need to adjust that if you have a different file (you'll need to change it in 3 places).

### Sample Web.config

<?xml version="1.0" encoding="utf-8"?>

<!--

This configuration file is required if iisnode is used to run node processes behind

IIS or IIS Express. For more information, visit:

https://github.com/tjanczuk/iisnode/blob/master/src/samples/configuration/web.config

-->

<configuration>

<system.webServer>

<!-- Visit http://blogs.msdn.com/b/windowsazure/archive/2013/11/14/introduction-to-websockets-on-windows-azure-web-sites.aspx for more information on WebSocket support -->

<webSocket enabled="false" />

<handlers>

<!-- Indicates that the server.js file is a node.js site to be handled by the iisnode module -->

<add name="iisnode" path="server.js" verb="\*" modules="iisnode"/>

</handlers>

<rewrite>

<rules>

<!-- Do not interfere with requests for node-inspector debugging -->

<rule name="NodeInspector" patternSyntax="ECMAScript" stopProcessing="true">

<match url="^server.js\/debug[\/]?" />

</rule>

<!-- First we consider whether the incoming URL matches a physical file in the /public folder -->

<rule name="StaticContent">

<action type="Rewrite" url="public{REQUEST\_URI}"/>

</rule>

<!-- All other URLs are mapped to the node.js site entry point -->

<rule name="DynamicContent">

<conditions>

<add input="{REQUEST\_FILENAME}" matchType="IsFile" negate="True"/>

</conditions>

<action type="Rewrite" url="server.js"/>

</rule>

</rules>

</rewrite>

<!-- 'bin' directory has no special meaning in node.js and apps can be placed in it -->

<security>

<requestFiltering>

<hiddenSegments>

<remove segment="bin"/>

</hiddenSegments>

</requestFiltering>

</security>

<!-- Make sure error responses are left untouched -->

<httpErrors existingResponse="PassThrough" />

<!--

You can control how Node is hosted within IIS using the following options:

\* watchedFiles: semi-colon separated list of files that will be watched for changes to restart the server

\* node\_env: will be propagated to node as NODE\_ENV environment variable

\* debuggingEnabled - controls whether the built-in debugger is enabled

See https://github.com/tjanczuk/iisnode/blob/master/src/samples/configuration/web.config for a full list of options

-->

<!--<iisnode watchedFiles="web.config;\*.js"/>-->

</system.webServer>

</configuration>

### ExpressJS

When building an Express app, the startup will be bin/www. Replace server.js in the sample provided above.

### NestJS

When building a Nestjs app, the startup will be dist/main.js. Replace server.js` in the sample provided above.

### Production Build

For Angular and other frameworks such as Reactjs, it is suggested that the customer build their application for production workloads. The the application is built, it will generate static files under the "dist" directory for Angular. Reactjs will place the built files under the "build" directory.

#### Configure the Virtual Directory

In order to serve the content under the "dist" or "build" directory, you'll need to modify the Virtual Directory configured for the app.

|  |  |
| --- | --- |
|  |  |
| A picture containing drawing  Description automatically generated | Note: By default, Angular apps will be have content under "dist/<app-name>". Please update the path accordingly. |
|  |  |

#### Azure Portal

1. Go to https://portal.azure.com
2. Select your Windows Web App and go to "Configuration".
3. Select the "Path mappings" tab.
4. Edit the "Virtual Path" for "/" and change the "Physical path" to the necessary directory. In the example below, I'm changing it to "site/wwwwroot/build".

A screenshot of a cell phone

Description automatically generated

Azure CLI

Update the <app\_name> and <rg\_name before using.

az resource update --name <app\_name>/config/web --resource-group <rg\_name> --resource-type Microsoft.Web/sites/config --set 'properties.virtualApplications=[{\"physicalPath\":\"site\\wwwroot\",\"preloadEnabled\":false,\"virtualDirectories\":null,\"virtualPath\":\"/\"}

Express Routes - Web.config

In order to properly route requests, we'll need to create a web.config file under D:\home\site\wwwroot with the following content.

<?xml version="1.0"?>

<configuration>

<system.webServer>

<rewrite>

<rules>

<rule name="React Routes" stopProcessing="true">

<match url=".\*" />

<conditions logicalGrouping="MatchAll">

<add input="{REQUEST\_FILENAME}" matchType="IsFile" negate="true" />

<add input="{REQUEST\_FILENAME}" matchType="IsDirectory" negate="true" />

<add input="{REQUEST\_URI}" pattern="^/(api)" negate="true" />

</conditions>

<action type="Rewrite" url="/" />

</rule>

</rules>

</rewrite>

</system.webServer>

</configuration>

Angular Routes - Web.config

For older versions of Angular, it may be necessary to include a web.config file for routing to work properly. Sample web.config provided below.

<?xml version="1.0"?>

<configuration>

<system.webServer>

<rewrite>

<rules>

<rule name="Angular Routes" stopProcessing="true">

<match url=".\*" />

<conditions logicalGrouping="MatchAll">

<add input="{REQUEST\_FILENAME}" matchType="IsFile" negate="true" />

<add input="{REQUEST\_FILENAME}" matchType="IsDirectory" negate="true" />

</conditions>

<action type="Rewrite" url="/index.html" />

</rule>

</rules>

</rewrite>

</system.webServer>

</configuration>

IISNode.yml

The iisnode module allows many of the configuration options to be adjusted using the iisnode.yml file or the system.webServer/iisnode section of web.config. Settings in the iisnode.yml file, if present, take precedence over settings in the web.config. Below is the list of options (most of which were described above) with their default values. For detailed and most current description of the options check out the configuration sample.

# The optional iisnode.yml file provides overrides of

# the iisnode configuration settings specified in web.config.

node\_env: production

nodeProcessCommandLine: "c:\program files\nodejs\node.exe"

nodeProcessCountPerApplication: 1

maxConcurrentRequestsPerProcess: 1024

maxNamedPipeConnectionRetry: 100

namedPipeConnectionRetryDelay: 250

maxNamedPipeConnectionPoolSize: 512

maxNamedPipePooledConnectionAge: 30000

asyncCompletionThreadCount: 0

initialRequestBufferSize: 4096

maxRequestBufferSize: 65536

watchedFiles: \*.js;iisnode.yml

uncFileChangesPollingInterval: 5000

gracefulShutdownTimeout: 60000

loggingEnabled: true

logDirectoryName: iisnode

debuggingEnabled: true

debuggerPortRange: 5058-6058

debuggerPathSegment: debug

maxLogFileSizeInKB: 128

maxTotalLogFileSizeInKB: 1024

maxLogFiles: 20

devErrorsEnabled: true

flushResponse: false

enableXFF: false

promoteServerVars:

Configuration Description and Examples

|  |  |
| --- | --- |
|  |  |
| A picture containing drawing  Description automatically generated | Resource Lookup Note: For information about Configuration Description and Examples please review: <https://supportability.visualstudio.com/AzureAppService/_wiki/wikis/AzureAppService/336738/Azure-App-Service-OSS-Nodejs-Configuration?anchor=configuration-description-and-examples> |

Hands-on: Deploying Node.js Frameworks on App Service Windows

|  |  |
| --- | --- |
|  |  |
| A picture containing drawing, light  Description automatically generated | Individual activity. Duration 30 minutes.  Learn how to deploy several Nodejs/JavaScript frameworks on App Service Windows.  Follow the instructions in the article below:  <https://supportability.visualstudio.com/AzureAppService/_wiki/wikis/AzureAppService/336740/Azure-App-Service-OSS-Nodejs-Deployment?anchor=hands-on-labs---deploy-nodejs/javascript-frameworks-and-review-configurations> |
|  |  |

Troubleshooting Performance Issues for NodeJS on Windows

|  |  |
| --- | --- |
|  |  |
| clock icon | The time to complete this lesson, including exercises, is 30 minutes. |
|  |  |
| target icon | After this lesson you will be able to:   * Use node built-in profilers for CPU and Memory and review output to determine bottlenecks for Node.js on Windows |
|  |  |

|  |  |
| --- | --- |
|  |  |
| eye icon | Resource Lookup Note: For more information about Performance Issues for NodeJS built-in profilers and best practices on Windows, review:   * High CPU: <https://supportability.visualstudio.com/AzureAppService/_wiki/wikis/AzureAppService/336735/Azure-App-Service-OSS-Nodejs-Performance?anchor=high-cpu> * High Memory: <https://supportability.visualstudio.com/AzureAppService/_wiki/wikis/AzureAppService/336735/Azure-App-Service-OSS-Nodejs-Performance?anchor=high-memory> * Node.js Best practices for App Service Windows: <https://docs.microsoft.com/en-us/azure/app-service/app-service-web-nodejs-best-practices-and-troubleshoot-guide> |
|  |  |

### Overview

This lesson introduces you to troubleshooting performance issues for Node.js on Windows.

### Finding Memory Leaks and CPU Usage in Azure Node.js Web App

|  |  |
| --- | --- |
|  |  |
| A picture containing drawing  Description automatically generated | Resource Lookup Note: For Finding Memory Leaks and CPU Usage in Azure Node.js Web App follow the next link,   * High CPU: <https://supportability.visualstudio.com/AzureAppService/_wiki/wikis/AzureAppService/336735/Azure-App-Service-OSS-Nodejs-Performance?anchor=high-cpu> * High Memory: <https://supportability.visualstudio.com/AzureAppService/_wiki/wikis/AzureAppService/336735/Azure-App-Service-OSS-Nodejs-Performance?anchor=high-memory> |
|  |  |

### IISNODE Configuration

This schema file shows all the settings that you can configure for iisnode. Some of the settings that are useful for your application:

<https://github.com/Azure/iisnode/blob/master/src/config/iisnode_schema_x64.xml>

nodeProcessCountPerApplication

This setting controls the number of node processes that are launched per IIS application. The default value is 1. You can launch as many node.exes as your VM vCPU count by changing the value to 0. The recommended value is 0 for most applications so you can use all of the vCPUs on your machine. Node.exe is single-threaded so one node.exe consumes a maximum of 1 vCPU. To get maximum performance out of your node application, you want to use all vCPUs.

nodeProcessCommandLine

This setting controls the path to the node.exe. You can set this value to point to your node.exe version.

maxConcurrentRequestsPerProcess

This setting controls the maximum number of concurrent requests sent by iisnode to each node.exe. On Azure App Service, the default value is Infinite. You can configure the value depending on how many requests your application receives and how fast your application processes each request.

maxNamedPipeConnectionRetry

This setting controls the maximum number of times iisnode retries making the connection on the named pipe to send the requests to node.exe. This setting in combination with namedPipeConnectionRetryDelay determines the total timeout of each request within iisnode. The default value is 200 on Azure App Service. Total Timeout in seconds = (maxNamedPipeConnectionRetry \* namedPipeConnectionRetryDelay) / 1000

namedPipeConnectionRetryDelay

This setting controls the amount of time (in ms) iisnode waits between each retry to send the request to node.exe over the named pipe. The default value is 250 ms. Total Timeout in seconds = (maxNamedPipeConnectionRetry \* namedPipeConnectionRetryDelay) / 1000

By default, the total timeout in iisnode on Azure App Service is 200 \* 250 ms = 50 seconds.

logDirectory

This setting controls the directory where iisnode logs stdout/stderr. The default value is iisnode, which is relative to the main script directory (directory where main server.js is present)

debuggerExtensionDll

This setting controls what version of node-inspector iisnode uses when debugging your node application. Currently, iisnode-inspector-0.7.3.dll and iisnode-inspector.dll are the only two valid values for this setting. The default value is iisnode-inspector-0.7.3.dll. The iisnode-inspector-0.7.3.dll version uses node-inspector-0.7.3 and uses web sockets. Enable web sockets on your Azure webapp to use this version. See https://ranjithblogs.azurewebsites.net/?p=98 for more details on how to configure iisnode to use the new node-inspector.

flushResponse

The default behavior of IIS is that it buffers response data up to 4 MB before flushing, or until the end of the response, whichever comes first. iisnode offers a configuration setting to override this behavior: to flush a fragment of the response entity body as soon as iisnode receives it from node.exe, you need to set the iisnode/@flushResponse attribute in web.config to 'true':

XML

<configuration>

<system.webServer>

<!-- ... -->

<iisnode flushResponse="true" />

</system.webServer>

</configuration>

Enable the flushing of every fragment of the response entity body adds performance overhead that reduces the throughput of the system by ~5% (as of v0.1.13). The best to scope this setting only to endpoints that require response streaming (for example, using the <location> element in the web.config)

In addition to this, for streaming applications, you must also set responseBufferLimit of your iisnode handler to 0.

XML

<handlers>

<add name="iisnode" path="app.js" verb="\\*" modules="iisnode" responseBufferLimit="0"/>

</handlers>

watchedFiles

A semi-colon separated list of files that are watched for changes. Any change to a file causes the application to recycle. Each entry consists of an optional directory name as well as a required file name, which are relative to the directory where the main application entry point is located. Wild cards are allowed in the file name portion only. The default value is \*.js;iisnode.yml

recycleSignalEnabled

The default value is false. If enabled, your node application can connect to a named pipe (environment variable IISNODE\_CONTROL\_PIPE) and send a “recycle” message. This causes the w3wp to recycle gracefully.

idlePageOutTimePeriod

The default value is 0, which means this feature is disabled. When set to some value greater than 0, iisnode will page out all its child processes every ‘idlePageOutTimePeriod’ in milliseconds. See documentation to understand what page out means. This setting is useful for applications that consume a high amount of memory and want to page out memory to disk occasionally to free up RAM.

|  |  |
| --- | --- |
|  |  |
| A picture containing drawing, light  Description automatically generated | Note: Use caution when enabling the following configuration settings on production applications. The recommendation is to not enable them on live production applications. |
|  |  |

debugHeaderEnabled

The default value is false. If set to true, iisnode adds an HTTP response header iisnode-debug to every HTTP response it sends the iisnode-debug header value is a URL. Individual pieces of diagnostic information can be obtained by looking at the URL fragment, however, a visualization is available by opening the URL in a browser.

loggingEnabled

This setting controls the logging of stdout and stderr by iisnode. Iisnode captures stdout/stderr from node processes it launches and writes to the directory specified in the ‘logDirectory’ setting. Once this is enabled, your application writes logs to the file system and depending on the amount of logging done by the application, there could be performance implications.

devErrorsEnabled

The default value is false. When set to true, iisnode displays the HTTP status code and Win32 error code on your browser. The win32 code is helpful in debugging certain types of issues.

debuggingEnabled (do not enable on live production site)

This setting controls debugging feature. Iisnode is integrated with node-inspector. By enabling this setting, you enable debugging of your node application. Upon enabling this setting, iisnode creates node-inspector files in ‘debuggerVirtualDir’ directory on the first debug request to your node application. You can load the node-inspector by sending a request to http://yoursite/server.js/debug. You can control the debug URL segment with ‘debuggerPathSegment’ setting. By default, debuggerPathSegment=’debug’. You can set debuggerPathSegment to a GUID, for example, so that it is more difficult to be discovered by others.

|  |  |
| --- | --- |
|  |  |
| A picture containing drawing, light  Description automatically generated | Note: Read Debug node.js applications on Windows for more details on debugging through the following link:  <https://tomasz.janczuk.org/2011/11/debug-nodejs-applications-on-windows.html> |

### Scenarios and Recommendations/Troubleshooting

|  |  |
| --- | --- |
|  |  |
| A picture containing drawing  Description automatically generated | Resource Lookup Note: For information about Scenarios and Recommendations/Troubleshooting, review the following link:  <https://docs.microsoft.com/en-us/azure/app-service/app-service-web-nodejs-best-practices-and-troubleshoot-guide#scenarios-and-recommendationstroubleshooting>. |
|  |  |

Hands-on: Profiling samples - App Service Windows

|  |  |
| --- | --- |
|  |  |
| A picture containing drawing, light  Description automatically generated | Individual activity. Duration 30 minutes.  Learn how to configure, collect, analyze profiler traces with built-in profilers.  Follow the instructions in the article below:  <https://supportability.visualstudio.com/AzureAppService/_wiki/wikis/AzureAppService/336735/Azure-App-Service-OSS-Nodejs-Performance?anchor=hands-on-labs---profiling-samples---app-service-windows> |
|  |  |