(https://vscode.dev/github/microsoft/vscode-docs/blob/main/docs/remote/ssh-tutorial.md)

Remote development over SSH

This tutorial walks you through creating and connecting to a virtual machine (VM) on Azure using the Visual Studio Code Remote - SSH (https://marketplace.visualstudio.com/items?itemName=ms-vscode-remote.remote-ssh) extension. You'll create a Node.js Express web app to show how you can edit and debug on a remote machine with VS Code just like you could if the source code was local.

> **Note**: Your Linux VM can be hosted anywhere - on your local host, on premise, in Azure, or in any other cloud, as long as the chosen Linux distribution meets these prerequisites (/docs/remote/linux#_local-linux-prerequisites).

Prerequisites

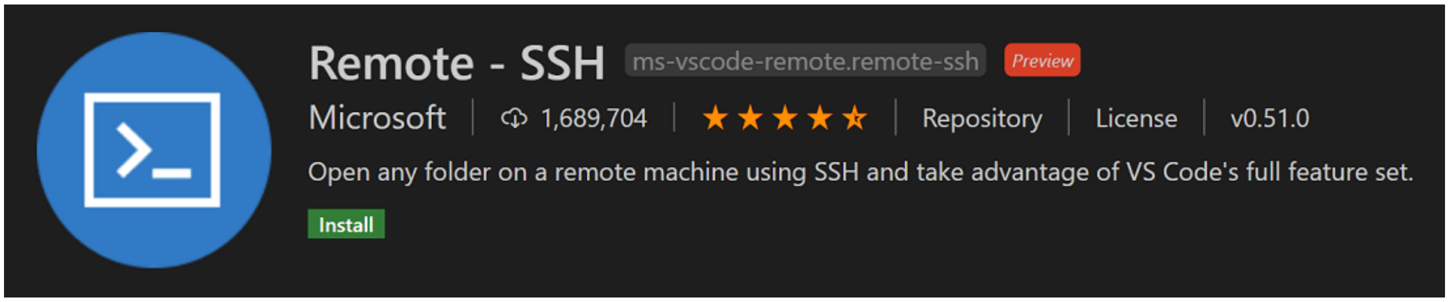
To get started, you need to have done the following steps:

1. Install an OpenSSH compatible SSH client (/docs/remote/troubleshooting#_installing-a-supported-ssh-client) (PuTTY is not supported).
2. Install Visual Studio Code (https://code.visualstudio.com).
3. Have an Azure subscription (If you don't have an Azure subscription, create a free account (https://azure.microsoft.com/free/?WT.mc_id=A261C142F) before you begin).

Install the extension

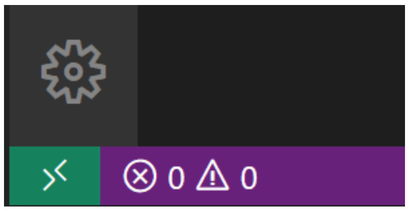
The Remote - SSH extension is used to connect to SSH hosts.

> Install the Remote - SSH extension (vscode:extension/ms-vscode-remote.remote-ssh)

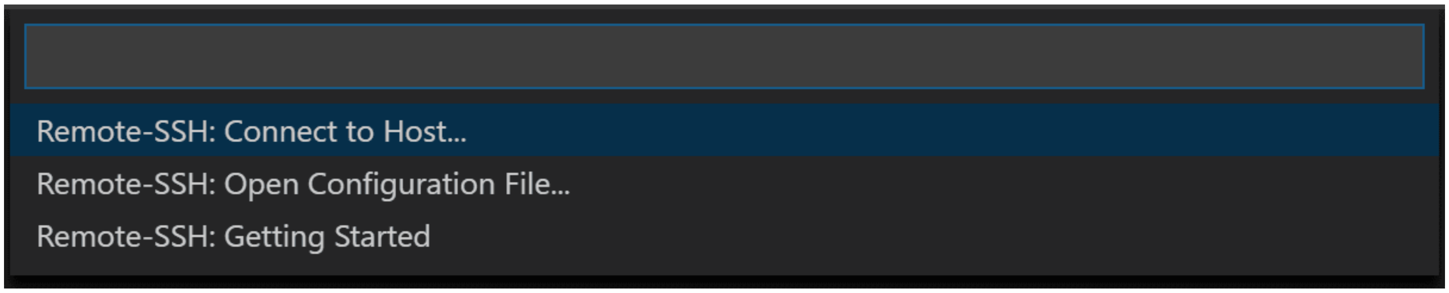

Remote - SSH
ms-vscode-remote.remote-ssh
Preview
Microsoft
1,689,704
Repository
License
v0.51.0
Open any folder on a remote machine using SSH and take advantage of VS Code's full feature set.
Install

Remote - SSH

With the Remote - SSH extension installed, you will see a new Status bar item at the far left.



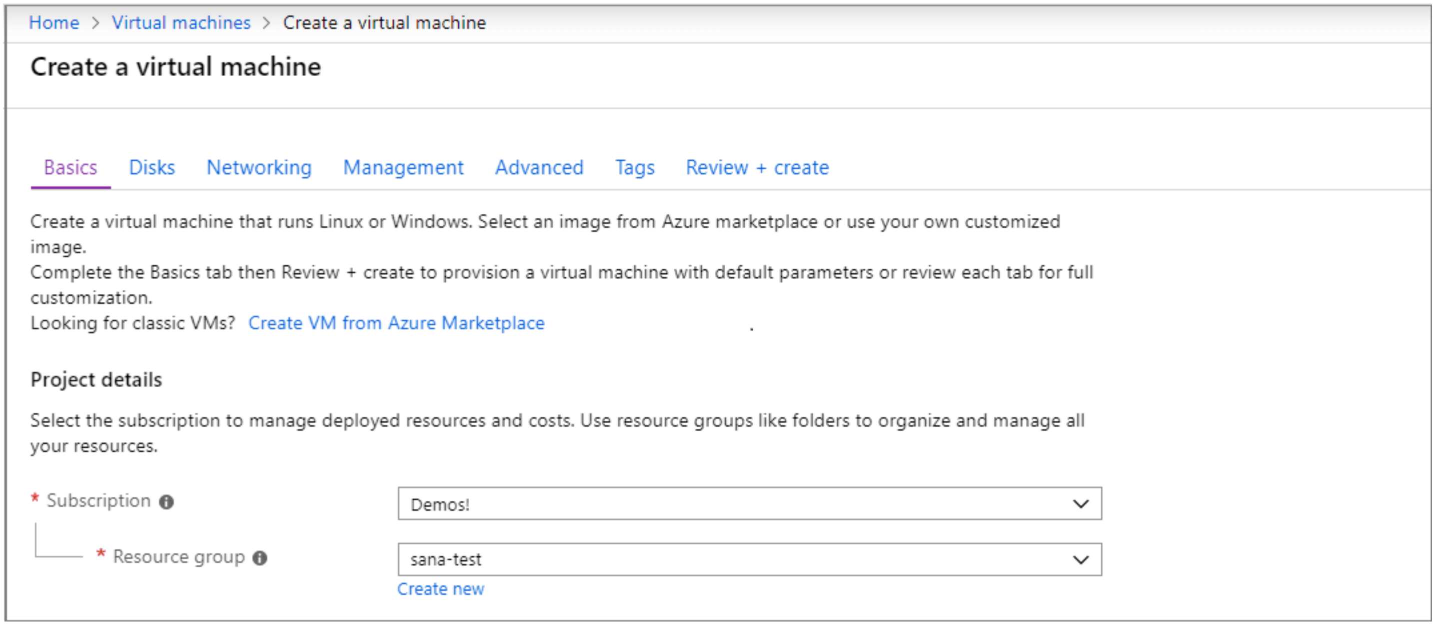
The Remote Status bar item can quickly show you in which context VS Code is running (local or remote) and clicking on the item will bring up the Remote - SSH commands.


Remote-SSH: Connect to Host...
Remote-SSH: Open Configuration File...
Remote-SSH: Getting Started

Create a virtual machine

If you don't have an existing Linux virtual machine, you can create a new VM through the Azure portal (https://portal.azure.com). In the Azure portal, search for "Virtual Machines", and choose **Add**. From there, you can select your Azure subscription and create a new resource group, if you don't already have one.

> **Note**: In this tutorial, we are using Azure, but your Linux VM can be hosted anywhere, as long as the Linux distribution meets these prerequisites (/docs/remote/linux#_local-linux-prerequisites).



Create a virtual machine

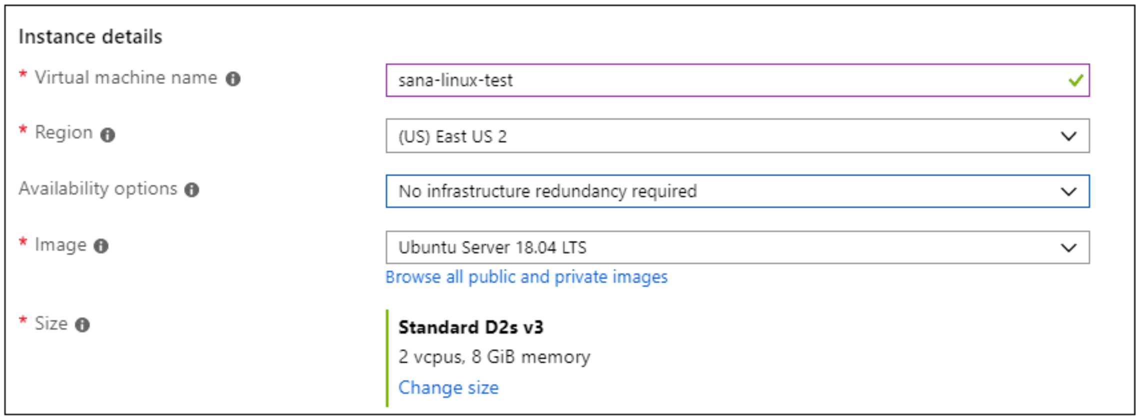
Create a virtual machine that runs Linux or Windows. Select an image from Azure marketplace or use your own customized image.
Complete the Basics tab then Review + create to provision a virtual machine with default parameters or review each tab for full customization.
Looking for classic VMs? Create VM from Azure Marketplace

Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

* Subscription: Demos!
* Resource group: sana-test

Now you can specify details of your VM, such as the name, the size, and the base image. Choose Ubuntu Server 18.04 LTS for this example, but you can choose recent versions of other Linux distros and look at VS Code's supported SSH servers (/docs/remote/troubleshooting#_installing-a-supported-ssh-server).



Instance details

* Virtual machine name: sana-linux-test
* Region: (US) East US 2
Availability options: No infrastructure redundancy required
* Image: Ubuntu Server 18.04 LTS
* Size: Standard D2s v3
2 vcpus, 8 GiB memory

Set up SSH

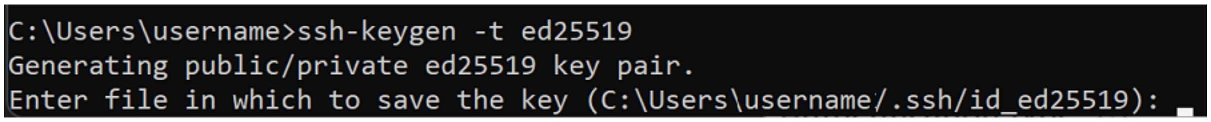
There are several authentication methods into a VM, including an SSH public/private key pair or a username and password. We recommend using key-based authentication (if you use a username/password, you'll be prompted to enter your credentials more than once by the extension). If you're on Windows and have already created keys using PuttyGen, you can reuse them (/docs/remote/troubleshooting#_reusing-a-key-generated-in-puttygen).

Create an SSH key

If you don't have an SSH key pair, open a bash shell or the command line and type in:

```
ssh-keygen -t ed25519
```

This will generate the SSH key. Press `Enter` at the following prompt to save the key in the default location (under your user directory as a folder named `.ssh`).



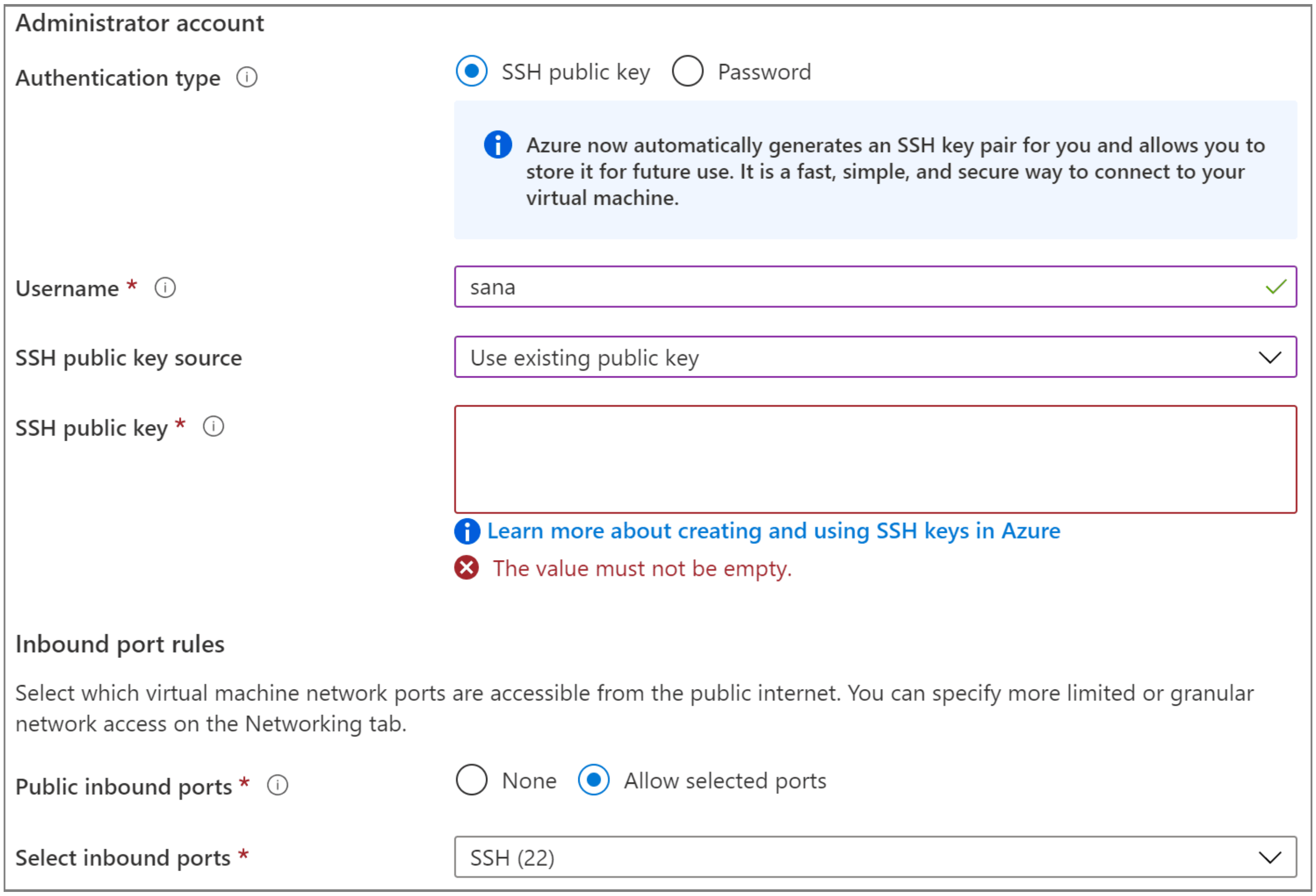
```
C:\Users\username>ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (C:\Users\username/.ssh/id_ed25519):
```

You will then be prompted to enter a secure passphrase, but you can leave that blank. You should now have a `id_ed25519.pub` file which contains your new public SSH key.

> **Note**: If you are using a legacy system that doesn't support the Ed25519 algorithm, you can use rsa instead: `ssh-keygen -t rsa -b 4096`.

Add SSH key to your VM

In the previous step, you generated an SSH key pair. Select **Use existing public key** in the dropdown for **SSH public key source** so that you can use the public key you just generated. Take the public key and paste it into your VM setup, by copying the entire contents of the `id_ed25519.pub` in the **SSH public key**. You also want to allow your VM to accept inbound SSH traffic by selecting **Allow selected ports** and choosing **SSH (22)** from the **Select inbound ports** dropdown list.



Administrator account

Authentication type	[✓ SSH public key] Password
	Azure now automatically generates an SSH key pair for you and allows you to store it for future use. It is a fast, simple, and secure way to connect to your virtual machine.
Username *	sana
SSH public key source	Use existing public key
SSH public key *	
	Learn more about creating and using SSH keys in Azure
	The value must not be empty.

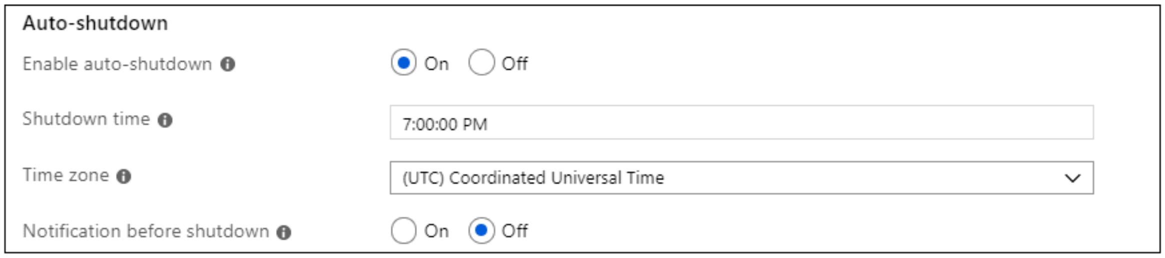
Inbound port rules

Select which virtual machine network ports are accessible from the public internet. You can specify more limited or granular network access on the Networking tab.

Public inbound ports *	None [✓ Allow selected ports]
Select inbound ports *	SSH (22)

Auto shutdown

A cool feature of using Azure VMs is the ability to enable auto shutdown (because let's face it, we all forget to turn off our VMs...). If you go to the **Management** tab, you can set the time you want to shut down the VM daily.



Auto-shutdown

Enable auto-shutdown	[✓ On] Off
Shutdown time	7:00:00 PM
Time zone	(UTC) Coordinated Universal Time
Notification before shutdown	On [✓ Off]

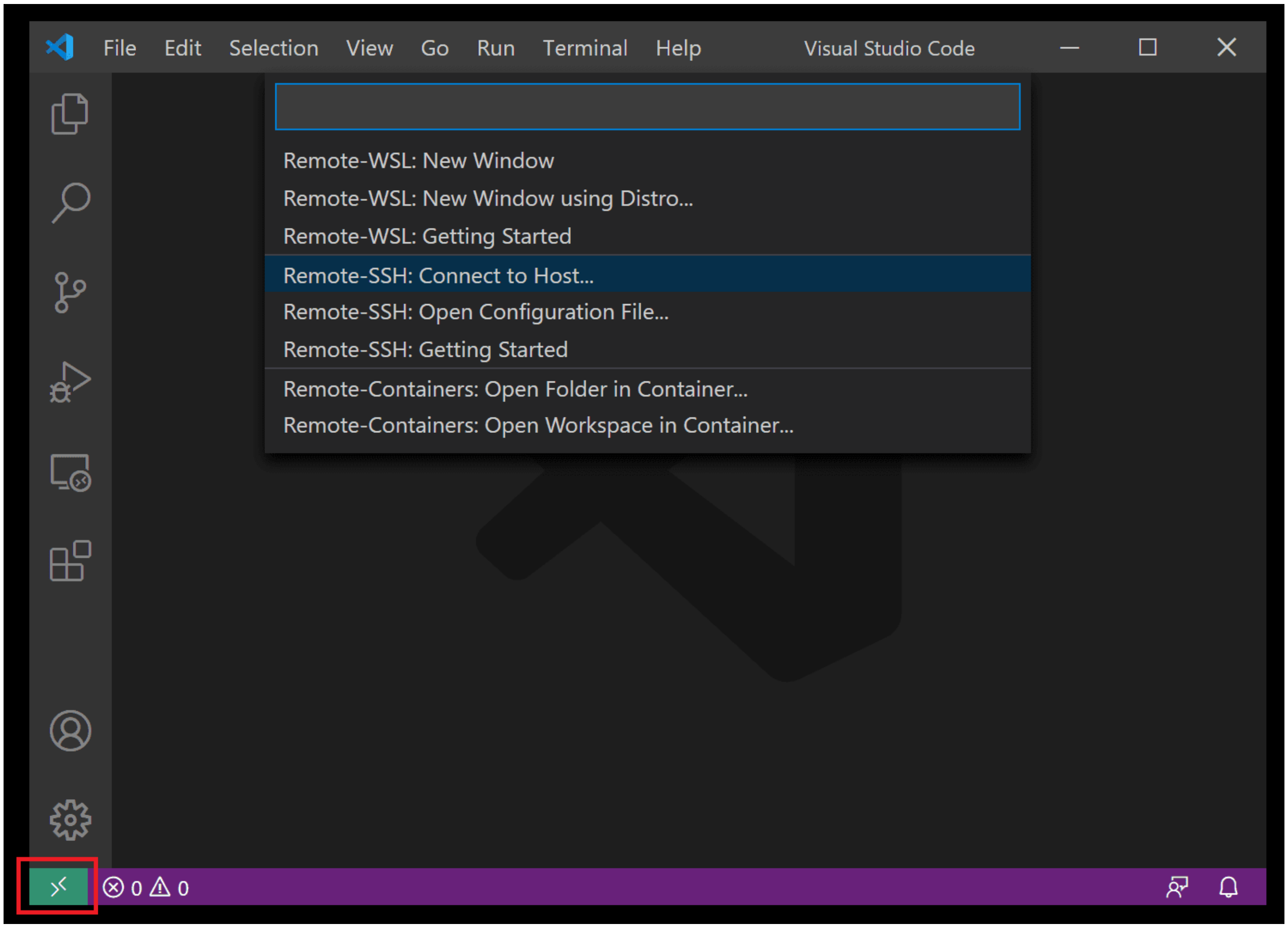
Select **Review and Create**, then **Create**, and Azure will deploy your VM for you!

Once the deployment is finished (it may take several minutes), go to the new resource view for your virtual machine.

Connect using SSH

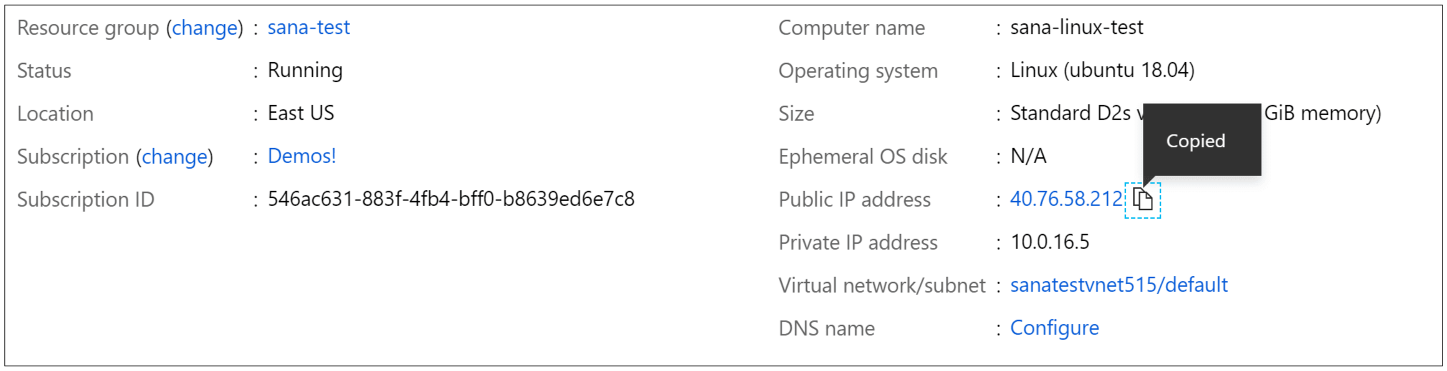
Now that you've created an SSH host, let's connect to it!

You'll have noticed an indicator on the bottom-left corner of the Status bar. This indicator tells you in which context VS Code is running (local or remote). Click on the indicator to bring up a list of Remote extension commands.


File
Edit
Selection
View
Go
Run
Terminal
Help
Visual Studio Code
Remote-WSL: New Window
Remote-WSL: New Window using Distro...
Remote-WSL: Getting Started
Remote-SSH: Connect to Host...
Remote-SSH: Open Configuration File...
Remote-SSH: Getting Started
Remote-Containers: Open Folder in Container...
Remote-Containers: Open Workspace in Container...
0
0

Choose the **Remote-SSH: Connect to Host** command and connect to the host by entering connection information for your VM in the following format: `user@hostname`.

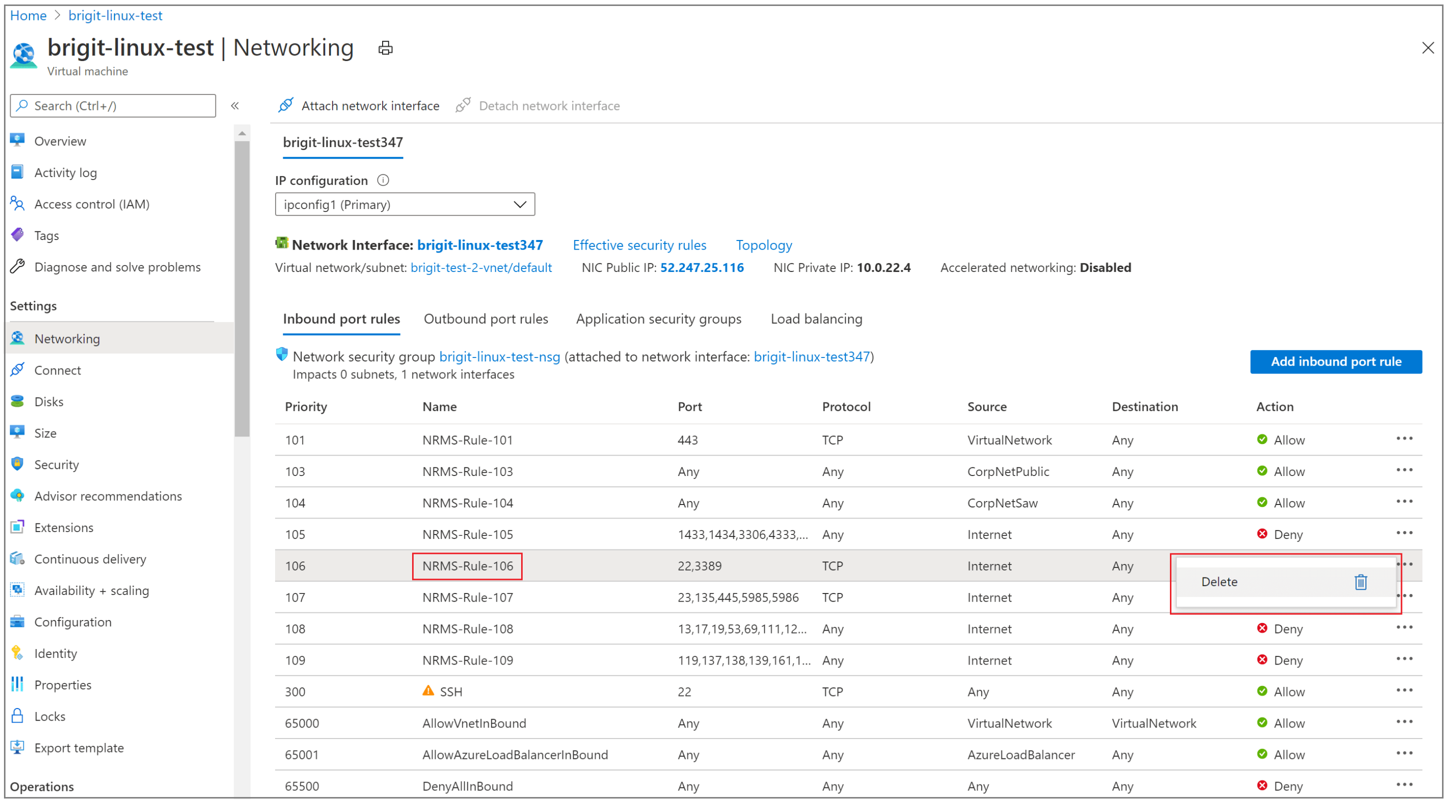
The `user` is the username you set when adding the SSH public key to your VM. For the `hostname`, go back to the Azure portal (https://portal.azure.com) and in the **Overview** pane of the VM you created, copy the **Public IP address**.



Resource group (change)	: sana-test	Computer name	: sana-linux-test
Status	: Running	Operating system	: Linux (ubuntu 18.04)
Location	: East US	Size	: Standard D2s v GiB memory)
Subscription (change)	: Demos!	Ephemeral OS disk	: N/A
Subscription ID	: 546ac631-883f-4fb4-bff0-b8639ed6e7c8	Public IP address	: 40.76.58.212
		Private IP address	: 10.0.16.5
		Virtual network/subnet	: sanatestvnet515/default
		DNS name	: Configure

Before connecting in Remote - SSH, you can verify you're able to connect to your VM via a command prompt using `ssh user@hostname`.

> Note: If you run into an error `ssh: connect to host <host ip> port 22: Connection timed out`, you may need to delete NRMS-Rule-106 from the Networking tab of your VM:



Home > brigit-linux-test

brigit-linux-test | Networking

Virtual machine

IP configuration: ipconfig1 (Primary)

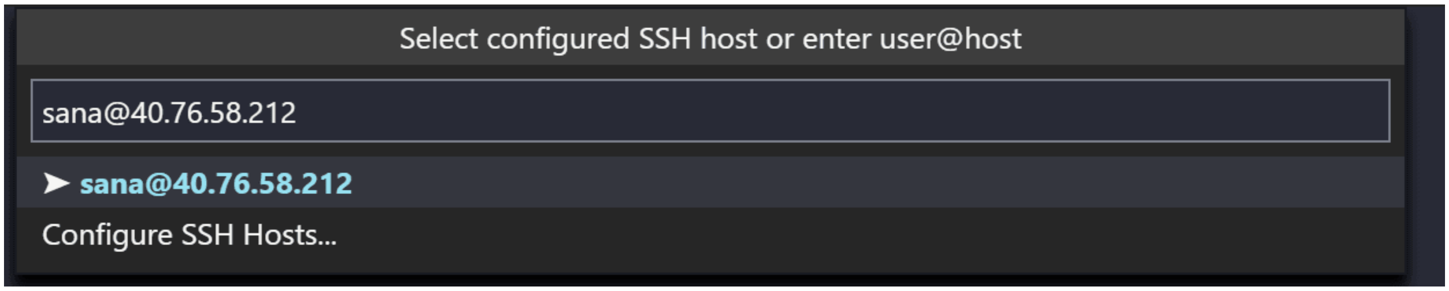
Network Interface: brigit-linux-test347 Effective security rules Topology

Virtual network/subnet: brigit-test-2-vnet/default NIC Public IP: 52.247.25.116 NIC Private IP: 10.0.22.4 Accelerated networking: Disabled

Network security group brigit-linux-test-nsg (attached to network interface: brigit-linux-test347)
Impacts 0 subnets, 1 network interfaces

Priority	Name	Port	Protocol	Source	Destination	Action
101	NRMS-Rule-101	443	TCP	VirtualNetwork	Any	Allow
103	NRMS-Rule-103	Any	Any	CorpNetPublic	Any	Allow
104	NRMS-Rule-104	Any	Any	CorpNetSaw	Any	Allow
105	NRMS-Rule-105	1433,1434,3306,4333,...	Any	Internet	Any	Deny
106	NRMS-Rule-106	22,3389	TCP	Internet	Any	
107	NRMS-Rule-107	23,135,445,5985,5986	TCP	Internet	Any	
108	NRMS-Rule-108	13,17,19,53,69,111,12...	Any	Internet	Any	Deny
109	NRMS-Rule-109	119,137,138,139,161,1...	Any	Internet	Any	Deny
300	⚠ SSH	22	TCP	Any	Any	Allow
65000	AllowVnetInBound	Any	Any	VirtualNetwork	VirtualNetwork	Allow
65001	AllowAzureLoadBalancerInBound	Any	Any	AzureLoadBalancer	Any	Allow
65500	DenyAllInBound	Any	Any	Any	Any	Deny

Set the user and hostname in the connection information text box.



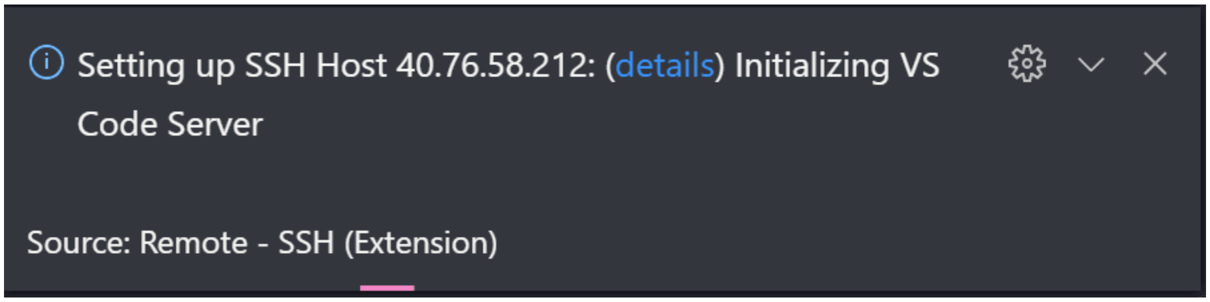
Select configured SSH host or enter user@host

sana@40.76.58.212

➤ sana@40.76.58.212

Configure SSH Hosts...

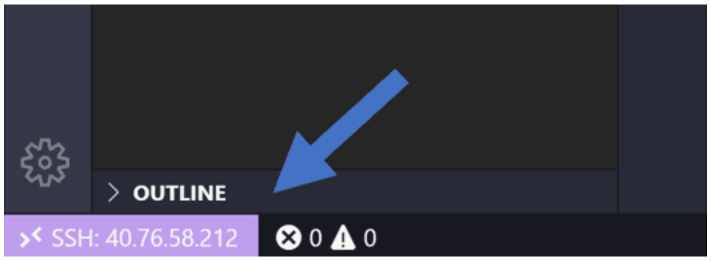
VS Code will now open a new window (instance). You'll then see a notification that the "VS Code Server" is initializing on the SSH Host. Once the VS Code Server is installed on the remote host, it can run extensions and talk to your local instance of VS Code.



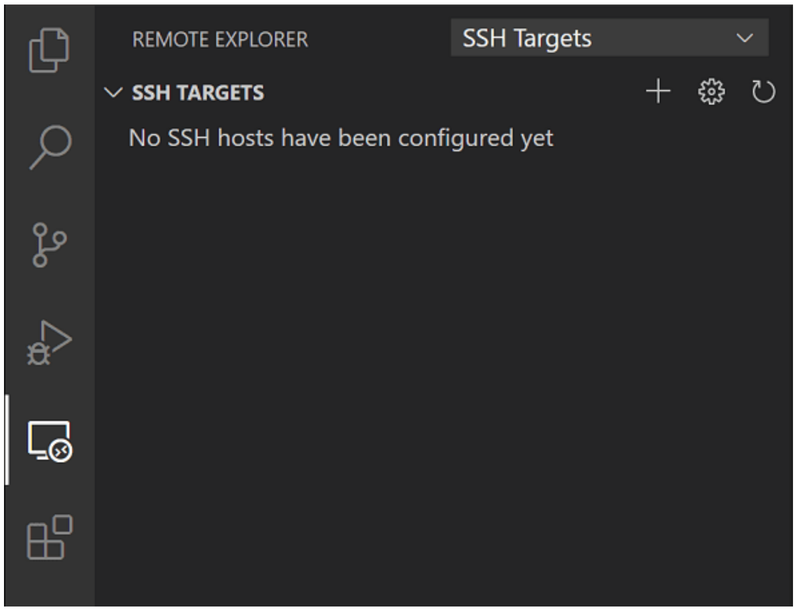
Setting up SSH Host 40.76.58.212: (details) Initializing VS Code Server

Source: Remote - SSH (Extension)

You'll know you're connected to your VM by looking at the indicator in the Status bar. It shows the hostname of your VM.

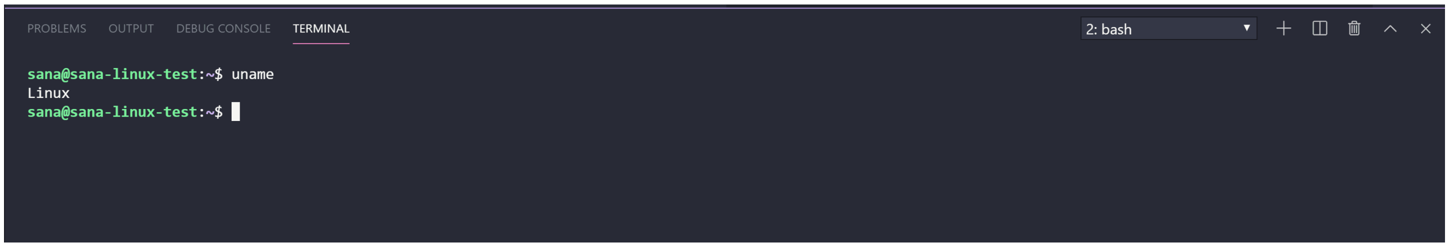


The Remote - SSH extension also contributes a new icon on your Activity bar, and clicking on it will open the Remote explorer. From the dropdown, select **SSH Targets**, where you can configure your SSH connections. For instance, you can save the hosts you connect to the most and access them from here instead of entering the user and hostname.



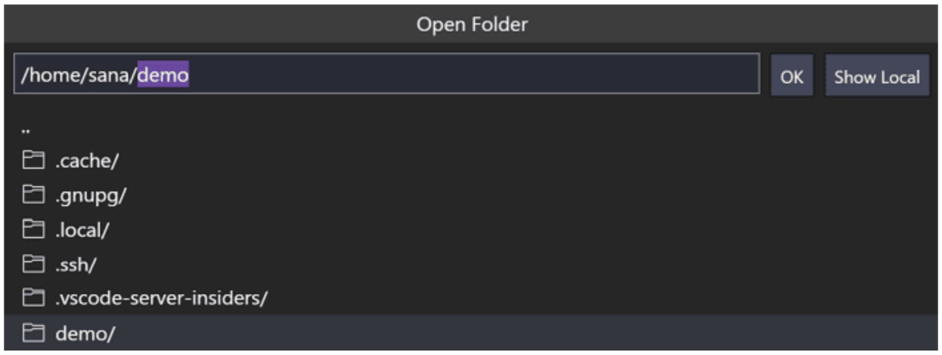
No SSH hosts have been configured yet

Once you're connected to your SSH host, you can interact with files and open folders on the remote machine. If you open the integrated terminal (`Ctrl+`` `), you'll see you're working inside a bash shell **while you're on Windows**.



```
sana@sana-linux-test:~$ uname
Linux
sana@sana-linux-test:~$
```

You can use the bash shell to browse the file system on the VM. You can also browse and open folders on the remote home directory with **File** > **Open Folder**.



```
/home/sana/demo
..
.cache/
.gnupg/
.local/
.ssh/
.vscode-server-insiders/
demo/
```

Create your Node.js application

In this step, you will create a simple Node.js application. You will use an application generator to quickly scaffold out the application from a terminal.

Install Node.js and npm

From the integrated terminal (`Ctrl+`` `), update the packages in your Linux VM, then install Node.js, which includes npm, the Node.js package manager.

```
sudo apt-get update
curl -sL https://deb.nodesource.com/setup_lts.x | sudo -E bash -
sudo apt-get install -y nodejs
```

You can verify the installations by running:

```
node --version
npm --version
```

Install the Express generator

Express (https://www.expressjs.com) is a popular framework for building and running Node.js applications. You can scaffold (create) a new Express application using the Express Generator (https://expressjs.com/en/starter/generator.html) tool. The Express Generator is shipped as an npm module and installed by using the npm command-line tool `npm`.

```
sudo npm install -g express-generator
```

The `-g` switch installs the Express Generator globally on your machine so that you can run it from anywhere.

Create a new application

You can now create a new Express application called `myExpressApp` by running:

```
express myExpressApp --view pug
```

The `--view pug` parameters tell the generator to use the pug (https://pugjs.org/api/getting-started.html) template engine.

To install all of the application's dependencies, go to the new folder and run `npm install`.

```
cd myExpressApp
npm install
```

Run the application

Last, let's ensure that the application runs. From the terminal, start the application using the `npm start` command to start the server.

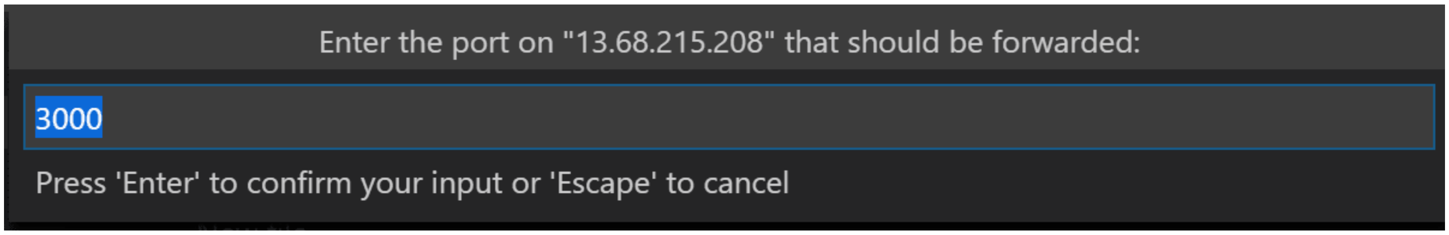
```
npm start
```

The Express app by default runs on http://localhost:3000 (http://localhost:3000). You won't see anything in your local browser on localhost:3000 because the web app is running on your virtual machine.

Port forwarding

To be able to browse to the web app on your local machine, you can leverage another feature called Port forwarding (/docs/remote/ssh#_temporarily-forwarding-a-port).

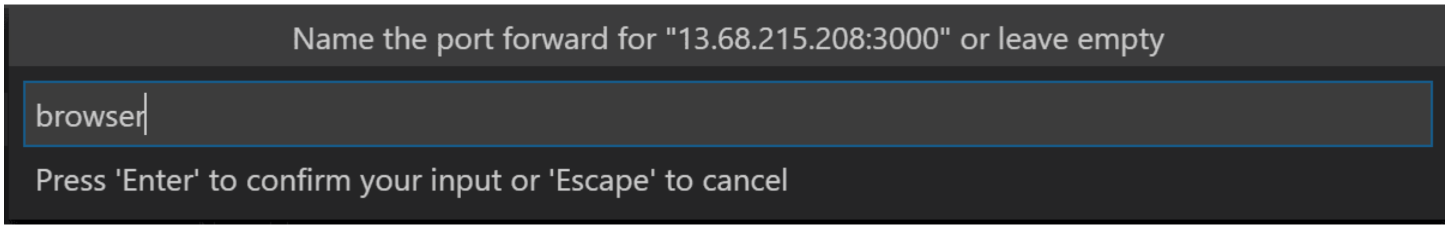
To be able to access a port on the remote machine that may not be publicly exposed, you need to establish a connection or a tunnel between a port on your local machine and the server. With the app still running, open the SSH Explorer and find the **Forwarded Ports** view. Click on the **Forward a port** link and indicate that you want to forward port 3000:


Enter the port on "13.68.215.208" that should be forwarded:

3000

Press 'Enter' to confirm your input or 'Escape' to cancel

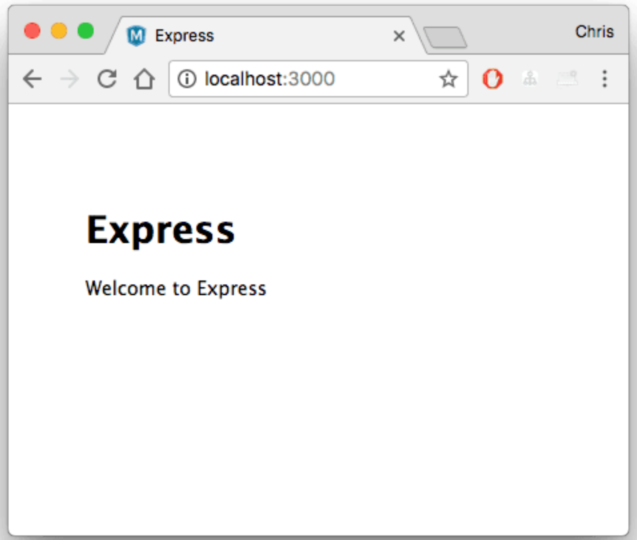
Name the connection "browser":


Name the port forward for "13.68.215.208:3000" or leave empty

browser

Press 'Enter' to confirm your input or 'Escape' to cancel

The server will now forward traffic on port 3000 to your local machine. When you browse to http://localhost:3000 (http://localhost:3000), you see the running web app.



Express

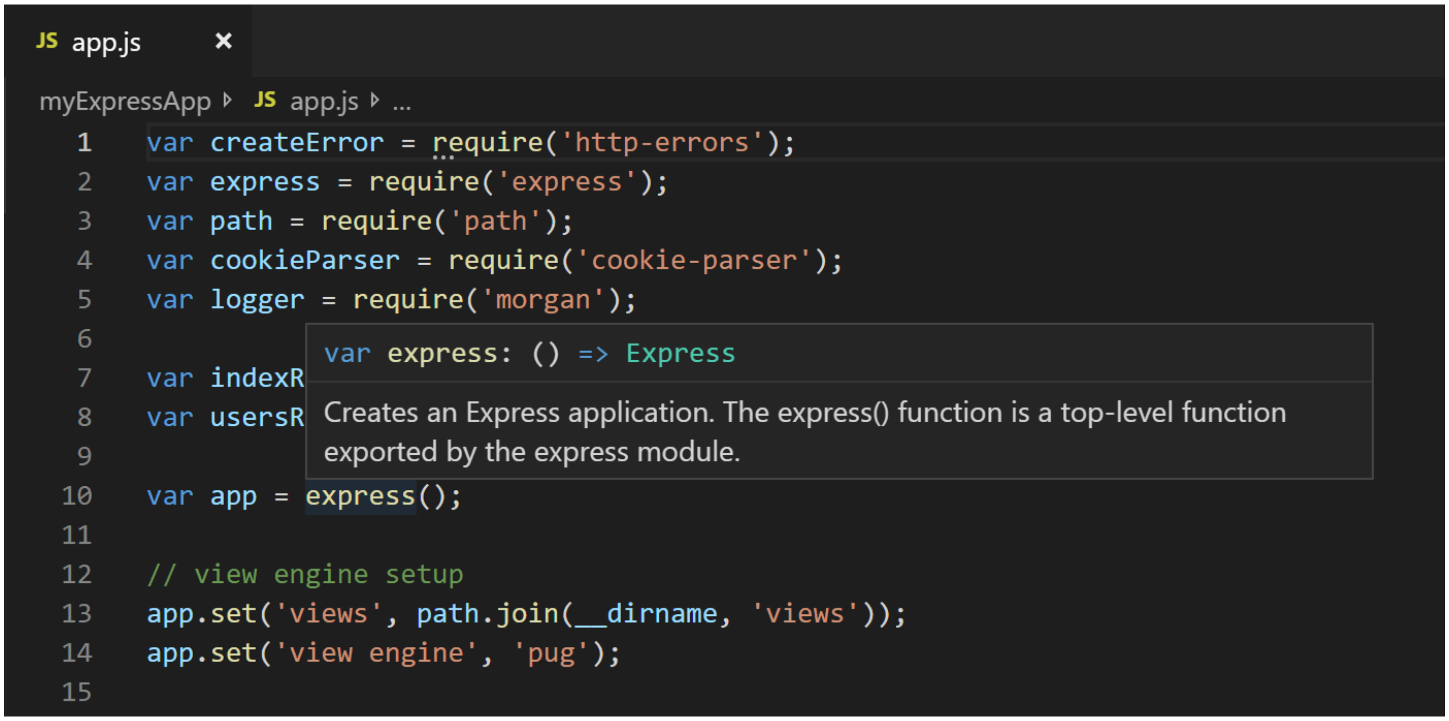
Welcome to Express

Edit and debug

From the Visual Studio Code File Explorer (`Ctrl+Shift+E`), navigate to your new `myExpressApp` folder and double-click the `app.js` file to open it in the editor.

IntelliSense

You have syntax highlighting for the JavaScript file as well as IntelliSense with hovers, just like you would see if the source code was on your local machine.

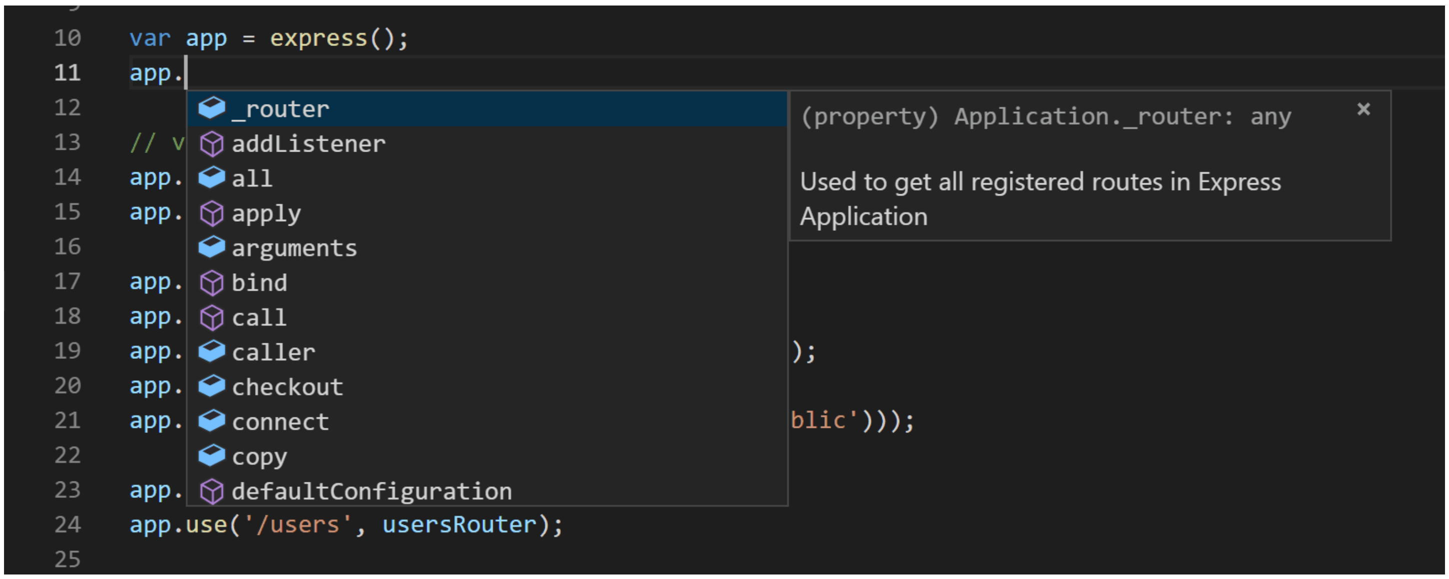


```
var createError = require('http-errors');
var express = require('express');
var path = require('path');
var cookieParser = require('cookie-parser');
var logger = require('morgan');

var indexR
var usersR

var app = express();

// view engine setup
app.set('views', path.join(__dirname, 'views'));
app.set('view engine', 'pug');

```

```
var express: () => Express
```

Creates an Express application. The express() function is a top-level function exported by the express module.

When you start typing, you'll get smart completions for the object methods and properties.



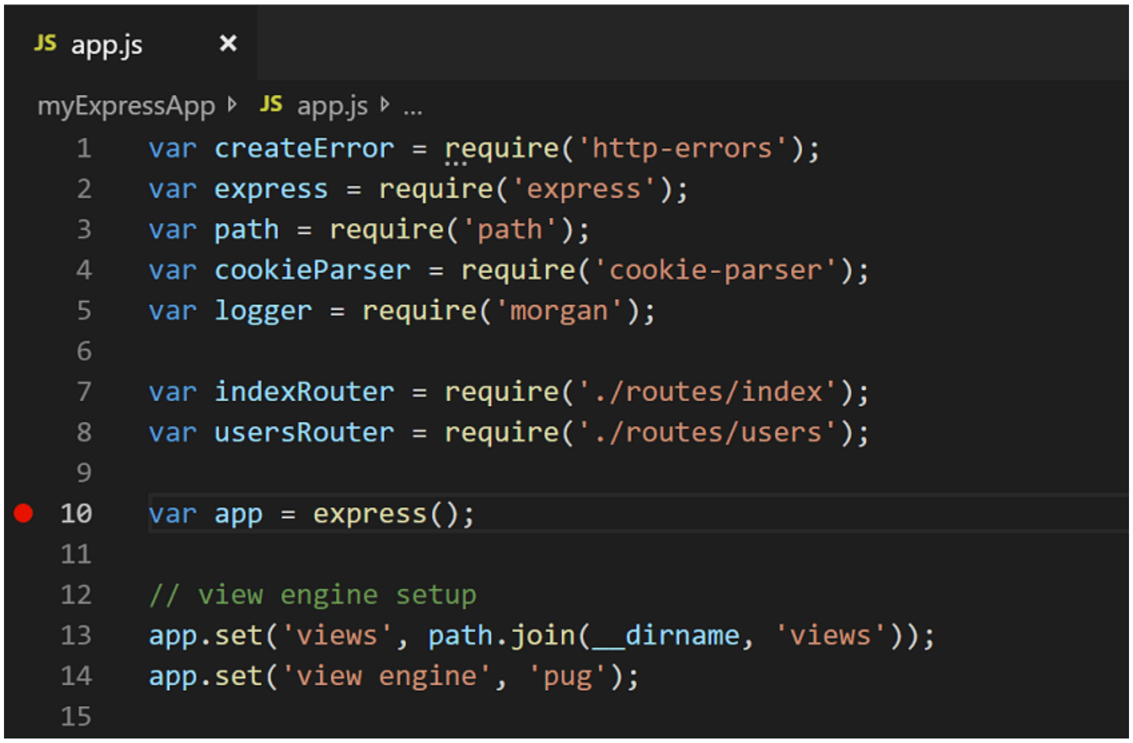
```
var app = express();
app.
    _router
// v  addListener
app.  all
app.  apply
      arguments
app.  bind
app.  call
app.  caller                                    );
app.  checkout
app.  connect                                   blic')));
      copy
app.  defaultConfiguration
app.use('/users', usersRouter);
```

(property) Application._router: any

Used to get all registered routes in Express Application

Debugging #

Set a breakpoint on line 10 of `app.js` by clicking in the gutter to the left of the line number or by putting the cursor on the line and pressing `F9`. The breakpoint will be displayed as a red circle.

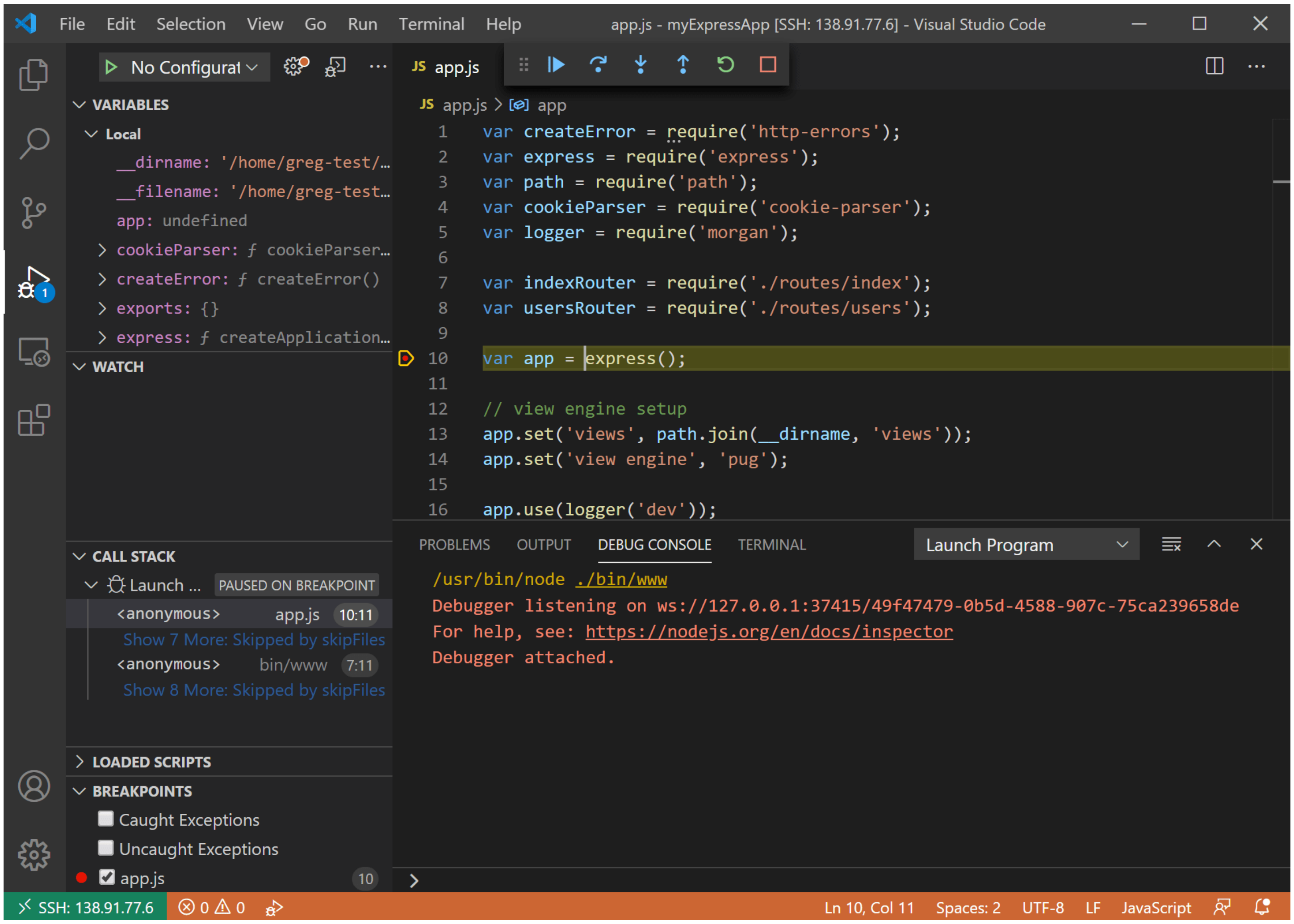


```
1  var createError = require('http-errors');
2  var express = require('express');
3  var path = require('path');
4  var cookieParser = require('cookie-parser');
5  var logger = require('morgan');
6
7  var indexRouter = require('./routes/index');
8  var usersRouter = require('./routes/users');
9
10 var app = express();
11
12 // view engine setup
13 app.set('views', path.join(__dirname, 'views'));
14 app.set('view engine', 'pug');
15
```

Now, press `F5` to run your application. If you are asked how to run the application, choose **Node.js**.

The app will start, and you'll hit the breakpoint. You can inspect variables, create watches, and navigate the call stack.

Press `F10` to step or `F5` again to finish your debugging session.


File Edit Selection View Go Run Terminal Help
app.js - myExpressApp [SSH: 138.91.77.6] - Visual Studio Code
No Configurat
app.js
VARIABLES
Local
WATCH
CALL STACK
Launch ...
PAUSED ON BREAKPOINT
LOADED SCRIPTS
BREAKPOINTS
Caught Exceptions
Uncaught Exceptions
app.js 10
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Launch Program
SSH: 138.91.77.6
Ln 10, Col 11 Spaces: 2 UTF-8 LF JavaScript

```
__dirname: '/home/greg-test/…
__filename: '/home/greg-test…
app: undefined
> cookieParser: ƒ cookieParser…
> createError: ƒ createError()
> exports: {}
> express: ƒ createApplication…
```

```
<anonymous>    app.js 10:11
Show 7 More: Skipped by skipFiles
<anonymous>    bin/www 7:11
Show 8 More: Skipped by skipFiles
```

```javascript
var createError = require('http-errors');
var express = require('express');
var path = require('path');
var cookieParser = require('cookie-parser');
var logger = require('morgan');

var indexRouter = require('./routes/index');
var usersRouter = require('./routes/users');

var app = express();

// view engine setup
app.set('views', path.join(__dirname, 'views'));
app.set('view engine', 'pug');

app.use(logger('dev'));
```

```
/usr/bin/node ./bin/www
Debugger listening on ws://127.0.0.1:37415/49f47479-0b5d-4588-907c-75ca239658de
For help, see: https://nodejs.org/en/docs/inspector
Debugger attached.
```

You get the full development experience of Visual Studio Code connected over SSH.

Ending your SSH connection

You can end your session over SSH and go back to running VS Code locally with **File** > **Close Remote Connection**.

Congratulations

Congratulations, you've successfully completed this tutorial!

Next, check out the other Remote Development extensions.

- Remote - WSL (https://marketplace.visualstudio.com/items?itemName=ms-vscode-remote.remote-wsl)
- Remote - Containers (https://marketplace.visualstudio.com/items?itemName=ms-vscode-remote.remote-containers)

Or get them all by installing the Remote Development (https://marketplace.visualstudio.com/items?itemName=ms-vscode-remote.vscode-remote-extensionpack) Extension Pack.

Was this documentation helpful?

Yes No

9/1/2022

IN THIS ARTICLE