Azure Blockchain Hands-on Lab

DURATION

1 Day ; Instructor-Lead

AUDIENCE

Application developer

REQUIREMENTS

Azure Pass

METHODOLOGY

This hands on-lab will be conducted with interactive lectures, PowerPoint presentations and practical exercises.

WORKSHOP

In this workshop, you will lean how to design a solution with Ethereum Blockchain and several Azure services to collect device telemetry information and enforce contract specifics related to conditions during the transport of goods.

ABSTRACT AND LEARNING OBJECTIVES

Students will learn how to build and configure an Internet of Things (IoT) Audit Solution using Azure Blockchain. Students will do this using Ethereum Blockchain with the use of Smart Contracts to collect device telemetry information in addition to enforce contract specifics related to conditions during transport of gods. Specifically, the IoT devices will report temperature and humidity data that will be validated through the Smart Contracts against agreed upon acceptable ranges.

Students will learn how to:
- Deploy and Configure Azure Blockchain Workbench
- Write and Deploy Ethereum Smart Contracts with Solidity
- Integrate bot IoT and Blockchain together into a single solution

AGENDA

Time	Task
9.00AM - 9.30AM	Presentation + Exercise 1
9.30AM-12.00AM	Exercise 2
12.00AM-1.00PM	Lunch
1.00PM - 1.15PM	Exercise 3
1.15PM - 3.30PM	Exercise 4
3.30PM - 4.00PM	Exercise 5
4.00PM - 5.00PM	Exercise 6

COURSE OUTLINE

Exercise 1: Setup Azure Active Directory Tenant

Duration: 15 minutes

In this exercise, the student will configure a new Azure AD Tenant that the Azure Blockchain Workbench will use to authenticate users.

The authentication and authorization of users in the Blockchain App Builder is performed using an Azure Active Directory (AAD) Tenant. To reduce the potential for conflict, a new AAD Tenant will be created for use with this lab.

Exercise 2: Deploy Azure Blockchain Workbench

Duration: 60 – 90 minutes

In this exercise, the student will deploy and setup Azure Blockchain App Builder.

Exercise 3: Check Blockchain Workbench Web Client Deployment

Duration: 15 minutes

In this exercise, the student will access the Azure Blockchain Workbench Web Client to make sure the deployment is working as expected. The student will also check the Ethereum network status.

Exercise 4: Create Smart Contract

Duration: 30 minutes

In this exercise, you will create a new Smart Contract that targets the Ethereum blockchain that is written in the Solidity programming language.

Exercise 5: Assign Users to Contract Personas

Duration: 15 minutes

In this exercise, the student will create some additional Users within Azure AD that correspond to the different user roles within the Blockchain solution. These Users will also be assigned to the different personas within the Azure Blockchain Workbench their roles within the workflow coincide with.

Exercise 6: Create and Process an Instance of the Smart Contract

Duration: 45 minutes

In this exercise, the student will create a new instance of the TelemetryCompliance Smart Contract, and work through the workflow of the Supply Chain as programmed in the code of the Smart Contract. The student will also test the Compliance piece by simulating IoT Device Telemetry by logging in as the “Simulated Device” user.

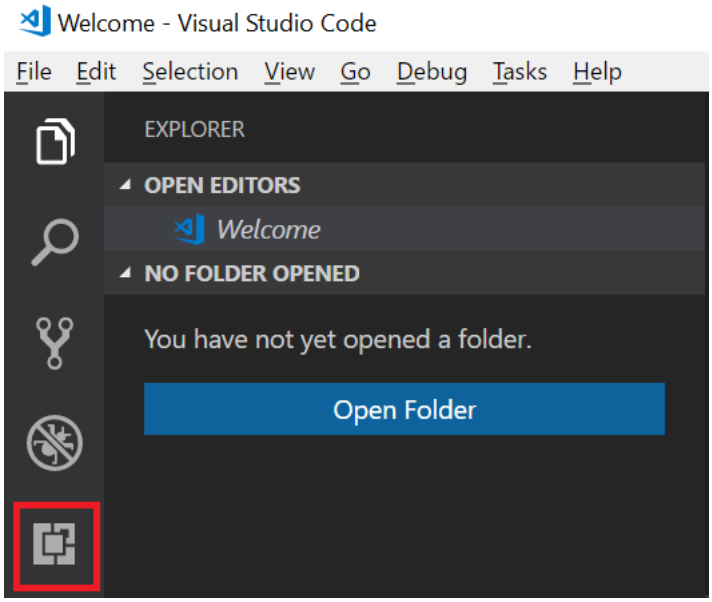
Setup Guide

Install **Visual Studio Code** with **Windows 10 Pro**

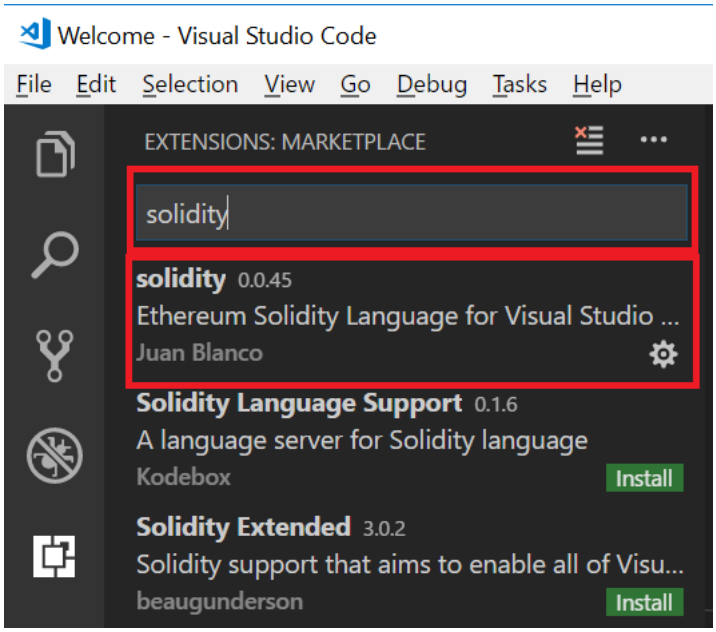
1. Download and install Visual Studio Code onto the development machine. You can download it from: https://code.visualstudio.com/

2. Run Visual Studio Code

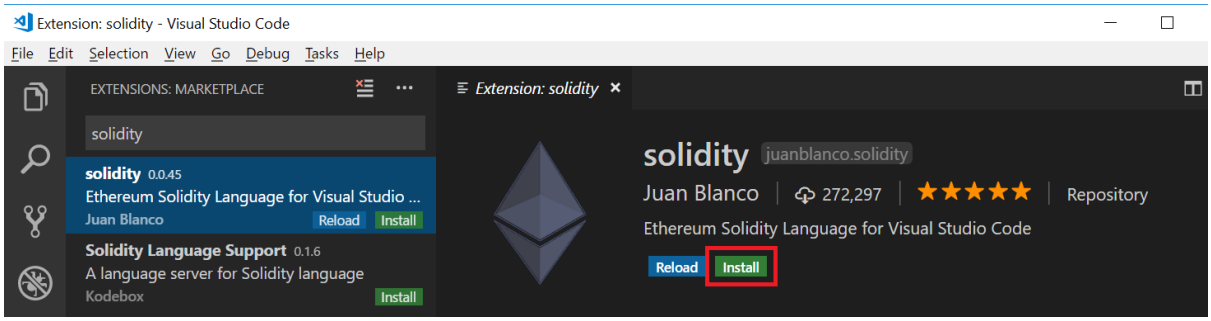
3. Click on the Extensions tab on the left-side of the window.



4. Type **Solidity** into the *Search Extensions* in Marketplace box, and click on the **solidity** extension authored by **Juan Blance**.



5. On the **solidity** extension, click **Install** to install this extension into Visual Studio Code.



6. Install puttygen.exe 64-bit (a RSA and DSA key generation utility)
https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html

puttygen.exe (a RSA and DSA key generation utility)

32-bit:	puttygen.exe	(or by FTP)	(signature)
64-bit:	puttygen.exe	(or by FTP)	(signature)