Using BLACKSTONE Virtual Quantum Machines on Azure (v0.1)

> **WARNING**
> DRAFT — subject to approval for Preview release on Azure Marketplace, and other revisions.

The BLACKSTONE Virtual Quantum Machine (VQM) service by Black Brane Systems, Inc. (https://www.blackbrane.com/) is a cloud-based, virtualized, *128-qubit universal quantum computer*. It acts as a plug-and-play alternative target machine for the Microsoft Quantum Developer Kit and Q# programs.

BLACKSTONE is available through the Azure marketplace and includes integration and extension libraries for: Microsoft's Quantum Development Kit (https://www.microsoft.com/quantum/) (QDK); Azure services; and the .NET Core runtime. The QDK integration library includes an API-compatible drop-in replacement for *Quantum Simulator* target machines, enabling Q# developers (https://docs.microsoft.com/quantum/language) to build and run quantum programs on BLACKSTONE without modification. The system executes up to data-flow complexity of 128 qubits in *constant-factor time* over the expected performance of a physical 128-qubit quantum chip, versus the *exponential-factor time* of full-state quantum simulators.

BLACKSTONE VQM and Controller

BLACKSTONE virtual quantum machines (VQMs) are configured and provisioned on demand. Developers provision and execute the VQM via API and library support. The method explained in this document is for end users to create a virtual machine with Q# libraries for the BLACKSTONE API. This instance will also include public domain (MIT licensed) elements of Microsoft's QDK. As this will be a development environment, we offer this in both Windows and Linux configurations.

BLACKSTONE Controllers must be a GPU instance as they are used for pre and post processing the data that goes to the VQM. The size of this instance will be highly dependent on the desired quantum algorithm and surrounding logic.

> **NOTE**
> Black Brane does not charge for use of the BLACKSTONE Controller, although normal Azure charges will apply.

When the BLACKSTONE VQM is invoked, there will be a delay while the target Virtual Quantum Machine is created and provisioned. This can take up to 20 minutes. Charges for the VQM are only for actual execution time and our experiments to date indicate that the execution time for most Quantum circuits is under one second.

Create your BLACKSTONE Controller

To create an instance of the BLACKSTONE Controller:

1. Navigate to the BLACKSTONE marketplace listing on Azure portal (https://portal.azure.com/#create/blackbranesystemsinc-blackstone-vqm).
2. Select the Create button at the bottom of the page.
3. Basics:
    - Name: Unique name for the server you are creating.
    - Subscription: If you have more than one subscription, select the one on which the machine is to be created and billed.
    - Resource Group: Create a new one or use an empty existing Azure resource group in your subscription.
    - Location: Select a data center that has NC or ND series GPU VM instances. Within this criteria, you might prefer datacenters that already hold your data or are close to your physical location.
4. Settings:
    - Select one of the NC series (NC, NCv2, NCv3) or ND series GPU virtual machine sizes that meets your functional requirements for post-processing quantum measurements.
5. Summary:
    - Verify that all information you entered is correct.
6. Buy:
    - Click Buy to start the provisioning. A link is provided on this screen showing details of pricing for use of the VQM.

How to access the Virtual Quantum Machine controller

Once the Resource Group for the new instance of BLACKSTONE Controller has been provisioned, ensure that you have generated your authorization keys using the service configuration section on Azure portal, and have allowed connections from the Internet to be made to the SSL-Encrypted WebSockets and HTTPS ports. You can verify that your instance of the BLACKSTONE Controller is functioning correctly by requesting the following resource from your BLACKSTONE server address:

```
GET https://<unique-name>.vqm.blackstone.blackbrane.com/v1-0/status/
```

Once you have verified that your BLACKSTONE Controller has been successfully created and provisioned, you are ready to start using the extension libraries that have been configured to access your instances.

Create Configuration File

The first time you launch one of the BLACKSTONE integration libraries, it will prompt you to create a user configuration file with the settings needed to access your VQM.

Getting Started with the BLACKSTONE Virtual Quantum Machine

The BLACKSTONE Virtual Quantum Machine is deployed as a plug-and-play extension to Microsoft's Quantum Development Kit (https://www.microsoft.com/quantum/), and can be called from any .NET Core assembly.

If you're new to Q# and the QDK, first head over to the Microsoft Quantum Development documentation site, where you'll learn how to install the QDK, write your first Quantum program in Q#, and manage quantum machines and drivers.

Once you've gotten comfortable with the QDK, or if you're already a seasoned Quantum Programmer, you can get started with the BLACKSTONE Virtual Quantum Machine by refactoring your existing Q# projects.

Refactoring the Driver of the Bell State Q# Quickstart to Support BLACKSTONE

Quantum Machines are managed by classical *Drivers*, which are .NET Core assemblies that can be written in C#, F#, VB.NET, or any other language that targets the .NET Core runtime.

After completing the Q# Quickstart (https://docs.microsoft.com/quantum/quickstart) tutorial for writing your first quantum program, you should have a file `Driver.cs` in a Project and Solution called `Bell`, with the following contents:

```
using System;

using Microsoft.Quantum.Simulation.Core;
using Microsoft.Quantum.Simulation.Simulators;

namespace Quantum.Bell
{
    class Driver
    {
        static void Main(string[] args)
        {
            using (var qsim = new QuantumSimulator())
            {
                // Try initial values
                Result[] initials = new Result[] { Result.Zero, Result.One };
                foreach (Result initial in initials)
                {
                    var res = BellTest.Run(qsim, 1000, initial).Result;
                    var (numZeros, numOnes, agree) = res;
                    System.Console.WriteLine(
                        $"Init:{initial,-4} 0s={numZeros,-4} 1s={numOnes,-4} agree={agree,-4}");
                }
            }

            System.Console.WriteLine("Press any key to continue...");
            Console.ReadKey();
        }
    }
}
```

To replace the Quantum Simulator with the BLACKSTONE Virtual Quantum Machine, you need to add the BLACKSTONE package to your project dependencies, and then replace the used packages from the QDK to BLACKSTONE, by replacing these two lines at the top of `Driver.cs`:

```
using Microsoft.Quantum.Simulation.Core;
using Microsoft.Quantum.Simulation.Simulators;
```

with these two lines:

```
using BlackBraneSystems.Blackstone.Quantum.Core;
using BlackBraneSystems.Blackstone.Quantum.SimulatorApi;
```

The *Quantum Simulator API* included with the BLACKSTONE package on Nuget allows you to use the existing Quantum Simulator API from the QDK without modifying your code. All you have to change to get started is replacing the `using` statements at the beginning of your Driver files.

In keeping with best practices, or to use both the QDK Simulators and the BLACKSTONE VQM in the same assembly, you will want to respect the scoping boundaries of the respective namespaces, and refactor your source-code accordingly. Rewrite your copy of `Driver.cs` as follows:

```
using System;

using BlackBraneSystems.Blackstone.Quantum.Core;
using BlackBraneSystems.Blackstone.Quantum.Machine;

namespace Quantum.Bell
{
    class Driver
    {
        static void Main(string[] args)
        {
            using (var vqm = new VirtualQuantumMachine())
            {
                // Try initial values
                Result[] initials = new Result[] { Result.Zero, Result.One };
                foreach (Result initial in initials)
                {
                    var res = BellTest.Run(vqm, 1000, initial).Result;
                    var (numZeros, numOnes, agree) = res;
                    System.Console.WriteLine(
                        $"Init:{initial,-4} 0s={numZeros,-4} 1s={numOnes,-4} agree={agree,-4}");
                }
            }

            System.Console.WriteLine("Press any key to continue...");
            Console.ReadKey();
        }
    }
}
```

Instead of using the Quantum Simulator API, the Driver is now using the BLACKSTONE Quantum Machine namespace directly. The instance of the `QuantumSimulator` class has been replaced with an instance of `VirtualQuantumMachine`. And lastly, the local variable `qsim` for the instance of the `QuantumSimulator` class has been refactored to be the more appropriate `vqm`.

Command Line / Visual Studio Code | Visual Studio 2019

Run the program with these modifications in place:

```
dotnet run
```

This will automatically download the BLACKSTONE dependency libraries, rebuild the application, and run it at the command line.

Your output should be similar to the following:

```
Init:Zero 0s=499  1s=501  agree=1000
Init:One  0s=490  1s=510  agree=1000

Press any key to continue...
```