**Azure VM Automation Documentation – Aug 2022**

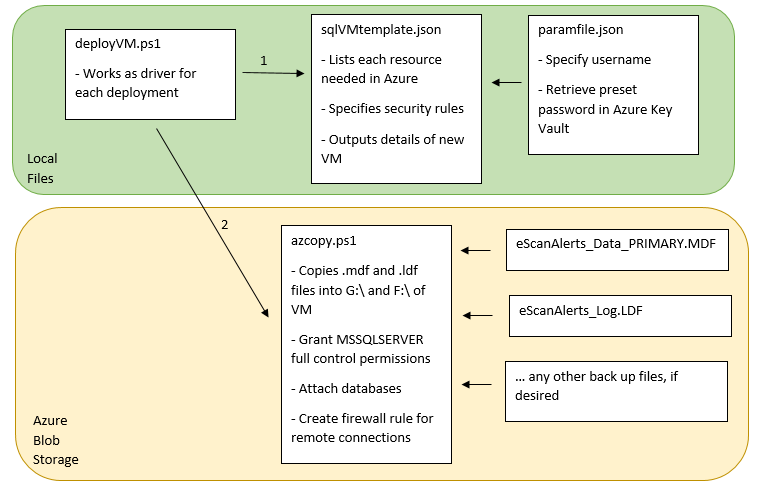
**Problem Statement**

A lot of the databases originating from TransUnion prior to the merge with FinThrive is available only on local servers. This means that only those with TU laptops can access them. FinThrive developers may wish to access them and have a “sandbox” environment to test out new features.

**Overview of Solution**

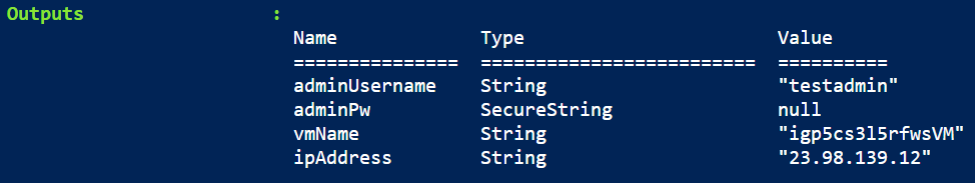
The system sets up an Azure environment for testing and developing. The cloud is accessible to all within FinThrive and is available on an as-needed basis. Each deployed virtual machine creates an isolated instance of a SQL Server preloaded with databases using backup files (.mdf and .ldf format) loaded into Azure blob storage. Therefore, as long as the database backup files are available in the blob storage, anyone in the company can set up a server to develop applications with.

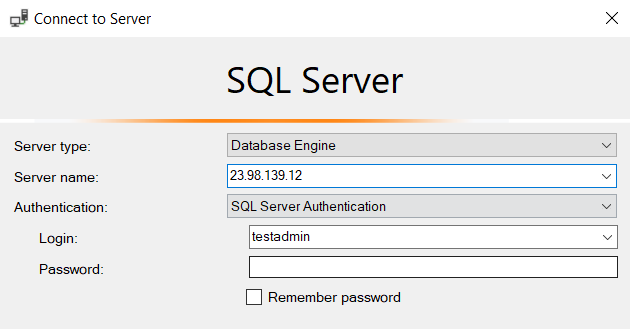
**System Components**



**User Guide**

1. Download cleanup.ps1, deployVM.ps1, paramfile.json, and sqlVMtemplate.json from [https://dev.azure.com/finthrive/insurance-discovery/\_git/gen-dev-sandbox](https://dev.azure.com/finthrive/insurance-discovery/_git/gen-dev-sandbox%20)
2. Upload these files to Cloud Shell (<portal.azure.com>) and run: .\deployVM.ps1
3. Once deployment finishes, use the ipAddress from output to enter as Server name in local SSMS





1. Log in via SQL Server Authentication
2. When ready to deallocate machine, use cleanup.ps1 and the resource prefix, which is output vmName minus “VM” at the end



Note: If wishing to log into the VM via Remote Desktop Protocol, uncomment last line of deployVM.ps1 before running. Then use command “download launch.rdp.” Open that file and log in.

**Clarifications**

All resources are currently deployed to the same resource group. The entity starting the deployment will need Contributor permission to create resources. Though the deployments are independent of each other, they all rely on a couple of shared preexisting resources:

1. Storage account (sqlvacmlqjaheddn2y): stores azcopy.ps1 script and all database files in blobs
2. Key vault (sqlVMdeployment): stores login password so that it isn’t viewable in code
3. Managed identity (VmtoDB): grants new VM permission to view storage account and key vault
4. Virtual network: groups VMs in allocated space so local devices can connect via cloud concepts

This process was developed using Azure Cloud Shell. Therefore, authentication was done beforehand. Using Windows Powershell would require a few extra steps (Az-Module, Az-Connect).

Currently, SQL Server Authentication and Windows Authentication (VM login) is the same but they can be made different.

**Next Steps**

Explore setting up pipeline for deployment. Would need to grant Azure roles to executing entity.

Explore auto-shutdown of VMs.

Explore security configurations to protect system from public access.

Consider scale at which this will be used—can we deploy multiple VMs in parallel from a single script? Hypderscale architecture and loadbalancing?

Consider password privacy. Do we want all users to share the same password? How will users know their password?

**Points of Contact**

Regarding existing code: Adarsh Sriram Sathyanarayana [adarsh.sriramsathyanarayana@finthrive.com](mailto:adarsh.sriramsathyanarayana@finthrive.com)

Regarding pipelines and DevOps: Tarak Ghosh [tarak.ghosh@finthrive.com](mailto:tarak.ghosh@finthrive.com)

Regarding project direction: Joe Jacob [joe.jacob@finthrive.com](mailto:joe.jacob@finthrive.com)