Getting Ready to Go

Before you get started with these challenges, you'll want to make sure your environment has the tools and setup to work with the environment.

Tools

All the tools used for these challenges are cross platform available and are usable on Mac OS X, Linux, and Windows environments.

> **NOTE**: If you are using a Linux platform, it is recommended to use one of these Ubuntu versions (https://github.com/Azure/azure-functions-core-tools#linux) to ensure the full functionality with the environment.

Tool	Examples / Download Links	Purpose
Your choice of editor or integrated development environment (IDE)	Visual Studio Code (https://code.visualstudio.com/download)	Editing and updating of code and configuration files
Terminal Environment	Bash (or similar such as Zsh) in Windows Subsystem for Linux (https://docs.microsoft.com/en-us/windows/wsl/install-win10) (either version, WSL2 recommended), (included by default in Mac and Linux); PowerShell on Windows (https://docs.microsoft.com/en-us/powershell/scripting/install/installing-powershell-core-on-windows), Mac (https://docs.microsoft.com/en-us/powershell/scripting/install/installing-powershell-core-on-macos), or Linux (https://docs.microsoft.com/en-us/powershell/scripting/install/installing-powershell-core-on-linux)	Used for running commands and scripts
Azure Command Line Interface (CLI)	Azure CLI (https://docs.microsoft.com/en-us/cli/azure/install-azure-cli)	Used for interacting with Azure resourcesq
Docker	Docker for Windows (https://docs.docker.com/docker-for-windows/install/), Mac (https://docs.docker.com/docker-for-mac/install/), or Ubuntu (https://docs.docker.com/engine/install/ubuntu/)	Used to build and run Docker containers locally
Kubernetes command-line tool	kubectl (https://kubernetes.io/docs/tasks/tools/) or via az aks install-cli (https://docs.microsoft.com/en-us/cli/azure/aks?view=azure-cli-latest#az_aks_install_cli)	Used for control and management of Kubernetes clusters, such as Azure Kubernetes Services (AKS) (https://azure.microsoft.com/en-us/services/kubernetes-service/)
Helm	helm (https://helm.sh/docs/intro/install/)	Used for package management within Kubernetes
Git	Git (https://git-scm.com/downloads), or one of the many GUI Clients (https://git-scm.com/downloads/guis/)	Local and remote code repository interaction and version control

Installation

1. Install on your local machine.
2. Utilize a remote container in VS Code or GitHub Codespaces (the configuration needed for this is provided within the source code repository).
3. Use Azure Cloud Shell (https://docs.microsoft.com/en-us/azure/cloud-shell/overview) which provides a terminal environment with `az`, `git`, `kubectl`, and `helm` (and other tools) pre-installed
4. Set up a VM in Azure and install tools in the VM

Local machine setup

Install tooling listed above.

Using Windows

While it's possible to complete this using Powershell, you may have an easier time using Bash on WSL (Windows Subsystem for Linux, linked above). Much of the Kubernetes ecosystem is very Linux-oriented; additionally, some of the sample CURL commands or other code snippets may need slight changes (`/` vs `\` and similar differences) to work in a Windows environment.

WSL2 is highly recommended as it offers significant performance improvements and some ease of use improvements over WSL1. To set up Docker with WSL2, follow the Docker WSL2 documention (https://docs.docker.com/desktop/windows/wsl/).

If you're using WSL1, check out this guide on setting up Docker with WSL1 (https://nickjanetakis.com/blog/setting-up-docker-for-windows-and-wsl-to-work-flawlessly).

Using Remote Containers

Remote container development allows you to use a pre-configured container with all necessary tooling for your project and limits what you need to install locally. The hack's source code repository comes with the configuration files necessary to build and run the project's "dev container."

VS Code

In order to get started developing in a container with VS Code, refer to the VS Code documentation on developing in a container - Getting Started (https://code.visualstudio.com/docs/remote/containers#_installation). On opening a project that supports remote containers, VS Code should prompt you to reopen in the container.

GitHub Codespaces

Check out GitHub's Quickstart for Codespaces (https://docs.github.com/en/codespaces/getting-started/quickstart).

Azure Cloud Shell

Check out the Quickstart for Bash in Azure Cloud Shell (https://docs.microsoft.com/en-us/azure/cloud-shell/quickstart).

As mentioned above, you can also use Powershell in Azure Cloud Shell (https://docs.microsoft.com/en-us/azure/cloud-shell/quickstart-powershell) but may find that the hack and Kubernetes ecosystem is slightly friendlier to Bash.

Create a VM in Azure

You can create an Azure VM (https://docs.microsoft.com/en-us/azure/virtual-machines/linux/quick-create-cli) to use as your development environment. However, this will offer similar functionality to Cloud Shell while requiring more time and effort to set up; it is likely preferable to use Cloud Shell over this option.

Further setup

Pull the SQL Server Docker Image

You'll run this in Challenge 1.

```
docker pull mcr.microsoft.com/mssql/server:2017-latest
```

Clone the Source Code Repository

The source code for the applications you'll be working with in this hack can be found at Microsoft-OpenHack/containers_artifacts on GitHub

Clone the repository from the command line (https://docs.github.com/en/github/creating-cloning-and-archiving-repositories/cloning-a-repository-from-github/cloning-a-repository) or use VS Code to clone a GitHub repository (https://code.visualstudio.com/docs/editor/github#_setting-up-a-repository).

If you're working with a team, it's highly recommended to create a new Git repository or create a fork (https://docs.github.com/en/get-started/quickstart/fork-a-repo) of the hack source code repository in order to have a way to collaborate and share files as you progress.

Useful Add-Ons

- Kubectx / Kubens (https://github.com/ahmetb/kubectx) allow for quick switching between Kubernetes contexts and namespaces
- Visual Studio Kubernetes Tools (https://marketplace.visualstudio.com/items?itemName=ms-kubernetes-tools.vscode-kubernetes-tools) can be helpful, especially to quickly create templated YAML files > *Note*: This is pre-installed in the remote container development option

References

Editor and Terminal Environment

- VS Code (https://code.visualstudio.com/)
- Windows Subsystem for Linux (https://docs.microsoft.com/en-us/windows/wsl/install-win10)
- Best Practices for Setting up a WSL Development Environment (https://docs.microsoft.com/en-us/windows/wsl/setup/environment)
- Install PowerShell (https://docs.microsoft.com/en-us/powershell/scripting/install/installing-powershell)
- Developing inside a VS Code remote container (https://code.visualstudio.com/docs/remote/containers)
- Creating a Github codespace using remote container (https://docs.github.com/en/codespaces/developing-in-codespaces/creating-a-codespace)
- Azure Cloud Shell (https://docs.microsoft.com/en-us/azure/cloud-shell/overview)
- Create a Linux VM in Azure (https://docs.microsoft.com/en-us/azure/virtual-machines/linux/quick-create-cli)

CLI Tooling

- Install Docker (https://docs.docker.com/engine/install/)
- Install Azure CLI (https://docs.microsoft.com/en-us/cli/azure/install-azure-cli)
- Git (https://git-scm.com/)
- Install Kubectl (https://kubernetes.io/docs/tasks/tools/)
- Install Helm (https://helm.sh/docs/intro/install/)

Working with Git

- Clone a GitHub repository (https://docs.github.com/en/github/creating-cloning-and-archiving-repositories/cloning-a-repository-from-github/cloning-a-repository)
- Use VS Code to clone a GitHub repository (https://code.visualstudio.com/docs/editor/github#_setting-up-a-repository)

Project Source Code

- Microsoft-OpenHack/containers_artifacts on GitHub (https://github.com/Microsoft-OpenHack/containers_artifacts)