

Table of Contents

[Getting Started](#)

[Overview](#)

[Platform](#)

[Architecture](#)

[Devices](#)

[Get Involved](#)

[Overview](#)

[Community Involvement](#)

[Request a Feature](#)

[Report a Bug](#)

[Influence Priorities](#)

[Track Work](#)

[Track Releases](#)

[Improve Documentation](#)

[Contribute Code](#)

[Overview](#)

[Development Tools](#)

[Overview](#)

[Guidance by Role](#)

[Install Tools](#)

[One Time Setup](#)

[Source Code](#)

[Overview](#)

[Commit Changes](#)



Getting Started

Welcome to the 3M IoT Platform on Azure documentation. This section provides an overview of the platform and how to influence platform improvements through community participation or code contributions.

Platform documentation has been organized into the following sections:

SECTION	DESCRIPTION
Getting Started	Introductory resources for those new to the platform
Use Cases	Examples of how the platform is being used at 3M.
Guides	Topics related to common questions and activities like how to manage devices.
APIs	Information about integrating the common services into custom solutions.

Throughout the documentation you'll find several sections can be downloaded as PDF for offline use.

Although we have taken care to ensure the accuracy of this content, mistakes do happen. See [How to Improve This Documentation](#) to help us keep it accurate over time.



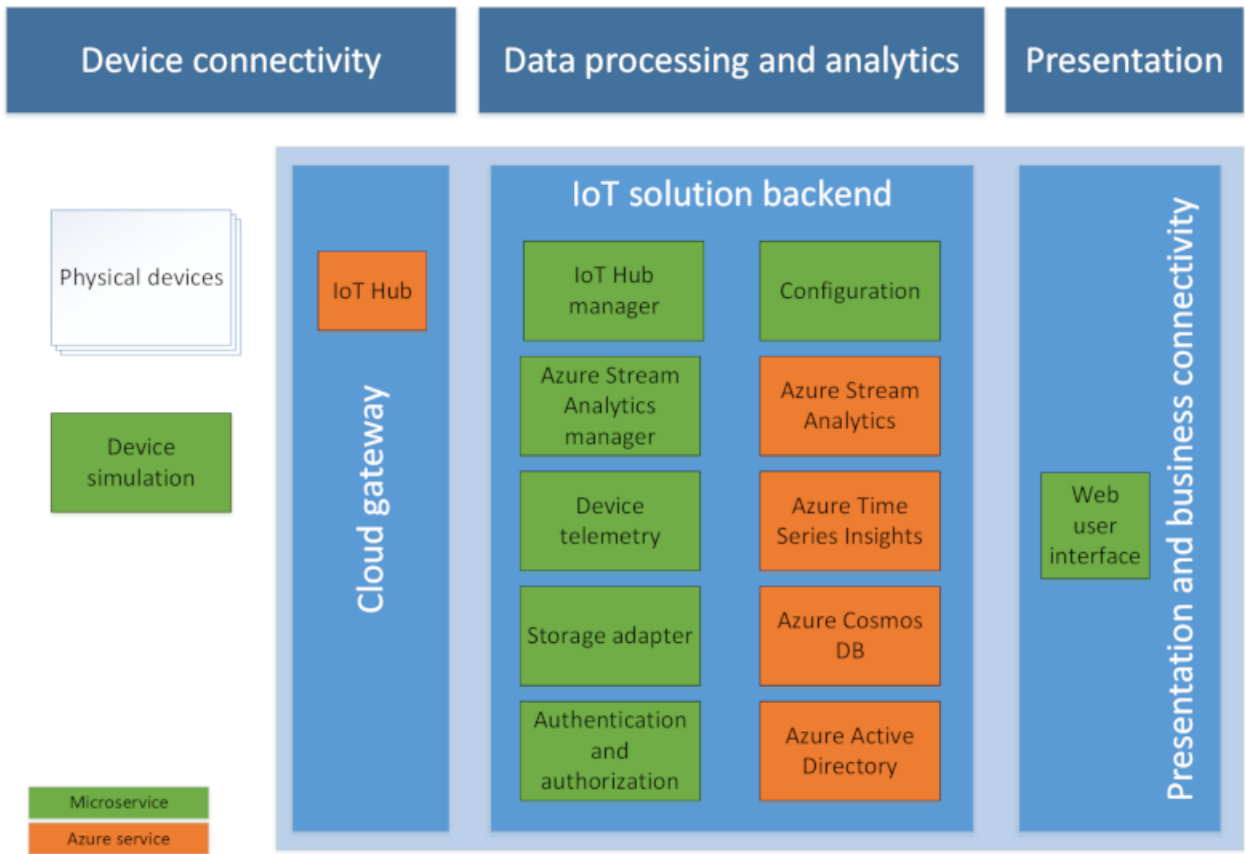
3M IoT Platform Overview

3M's IoT Platform on Azure offers an end-to-end solution for acceleration innovation and time-to-market by offering core sources that can be integrated into custom IoT Solutions. Device Management, Security, and cloud resource provisioning and other core requirements of any IoT solution can use the platform's user interface to manage devices. In addition, platform APIs offer direct integration to allow development to focus on their unique business objectives instead of common requirements. The platform combines Azure and custom services to pride a generic implementation ready for any IoT business scenario.

The platform code is available on [GitHub](#) where new features can be tracked and requested using the issues list. Currently requests are restricted to 3M. To learn how to request features or report issues, see the documentation [TODO](#).

Logical Components

The following diagram outlines the logical components of the platform overlaid on the IoT architecture:



IoT Hub

The [IoT Hub](#) is a Microsoft Azure Resource used to ingest telemetry sent from both real and simulated devices into the cloud. The hub makes the telemetry available to the services in the IoT solution for processing.

The IoT Hub also provides the following:

- Maintains an identity registry that stores the IDs and authentication keys of all the devices permitted to connect to the portal.
- Invokes methods on your devices on behalf of the solution accelerator.
- Maintains device twins for all registered devices. A device twin stores the property values reported by a device. A device twin also stores desired properties, set in the solution portal, for the device to retrieve when it next connects.
- Schedules jobs to set properties for multiple devices or invoke methods on multiple devices.

Web UI

A web user interface is available for device management. This presentation offers a user friendly way to register and configure devices, deploy firmware, and manage alerts, and provides a dashboard for generally observing device telemetry.

The web user interface is a [React Javascript](#) application that:

- Is served to the browser via backend services
- Is styled with Css and SaSS
- Interacts with other backend services like the authentication and authorization service to protect user data and communicate with Azure resources

Backend Services

The platform is made up several services running in [Azure Kubernetes Service \(AKS\)](#). The platform containerizes services to offer enterprise-grade security and scalability. The services - often called *microservices*- are written in .NET (C#) and provide RESTful endpoints that can be used for direct integration in business specific solutions.

Data processing and analytics

The solution includes the following components in the data processing and analytics part of the logical architecture:

IoT Hub manager

The solution includes the IoT Hub manager microservice to handle interactions with your IoT hub such as:

Creating and managing IoT devices. Managing device twins. Invoking methods on devices. Managing IoT credentials. This service also runs IoT Hub queries to retrieve devices belonging to user-defined groups.

The microservice provides a RESTful endpoint to manage devices and device twins, invoke methods, and run IoT Hub queries.

Device telemetry

The device telemetry microservice provides a RESTful endpoint for read access to device telemetry stored in Time Series Insights. The RESTful endpoint also enables CRUD operations on rules and read/write access for alarm definitions from storage.

Storage adapter

The storage adapter microservice manages key-value pairs, abstracting the storage service semantics, and presenting a simple interface to store data of any format using Azure Cosmos DB.

Values are organized in collections. You can work on individual values or fetch entire collections. Complex data structures are serialized by the clients and managed as simple text payload.

The service provides a RESTful endpoint for CRUD operations on key-value pairs. values

Azure Cosmos DB

Deployments use Azure Cosmos DB to store rules, alerts, configuration settings, and all other cold storage.

Azure Stream Analytics

The Azure Stream Analytics manager microservice manages Azure Stream Analytics (ASA) jobs, including setting their configuration, starting and stopping them, and monitoring their status.

The ASA job is supported by two reference data sets. One data set defines rules and one defines device groups. The rules reference data is generated from the information managed by the device telemetry microservice. The Azure Stream Analytics manager microservice transforms telemetry rules into stream processing logic.

The device groups reference data is used to identify which group of rules to apply to an incoming telemetry message. The device groups are managed by the configuration microservice and use Azure IoT Hub device twin queries.

The ASA jobs deliver the telemetry from the connected devices to Time Series Insights for storage and analysis.

Azure Stream Analytics is an event-processing engine that allows you to examine high volumes of data streaming from devices.

Azure Time Series Insights

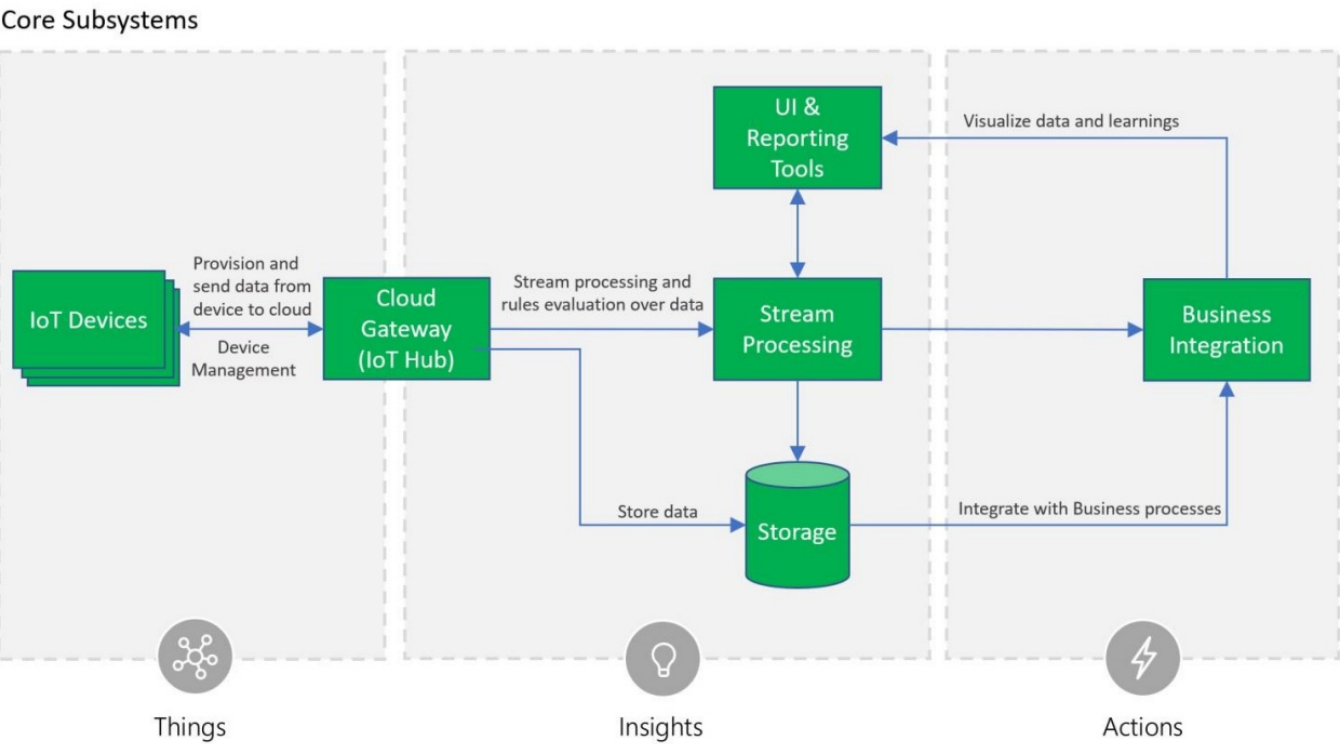
Azure Time Series Insights stores the telemetry from the devices connected to the solution accelerator. It also enables visualizing and querying device telemetry in the solution web UI.

Authentication and authorization microservice

The authentication and authorization microservice manages the users authorized to access the solution accelerator. User management supports a variety of identity service providers that support OpenId Connect, including Azure B2C.

Architecture

The 3M IoT Platform on Azure consists of the following subsystems: 1) devices (and/or on premise edge gateways) that have the ability to securely register with the cloud, and connectivity options for sending and receiving data with the cloud, 2) a cloud gateway service, or hub, to securely accept that data and provide device management capabilities, 3) stream processors that consume that data, integrate with business processes, and place the data into storage, and 4) a user interface to visualize telemetry data and facilitate device management.



Security is a critical consideration in each of the subsystems. The platform protects IoT devices, data, and communication by securely provisioning devices, secure connectivity between devices, edge devices, and the cloud, secure access to the backend solutions, and secure data protection in the cloud during processing and storage (encryption at rest).

The platform's use of Azure IoT Hub offers a fully-managed service that enables reliable and secure bi-directional communication between IoT devices and Azure services such as Azure Machine Learning and Azure Stream Analytics by using per-device security credentials and access control. Both Azure Cosmos DB for warm path storage and Azure Blob Storage for cold storage both are available and support encryption at rest.

All data access and permissions are governed by Azure Active Directory. External access is available via OpenID Connectivity as needed.



IoT Devices

The 3M IoT Platform provides a Web UI and set of APIs that can be tailored to unique industry and product requirements related to device enrollment and discovery, connectivity, remote configuration, and software updates.

The platform's common services include the following device management capabilities:

1. Device provisioning and discovery
2. Device access management
3. Remote control
4. Remote administration and monitoring
5. Remote configuration
6. Remote firmware and software update

To start working with devices, see the [Device Guide](#) for more information



How to Make Platform Improvements

The 3M IoT Platform on Azure has a robust CI/CD process to accommodate feedback and contributes from the community.

The IoT Platform source code is stored in a public GitHub repo and developed under an open-source model. The following table describes the common ways to initialize improvements to the platform:

RESOURCE	DESCRIPTION
Issues List	Offers the following: <ul style="list-style-type: none">• Report Bugs• Request Features• Influence Priorities• Track Progress
Documentation	All documentation offers community contributions.
Development	You can make direct contributions to the code.
Community Involvement	Join the 3M IoT Platform Community.

Use the links below to get more information on each topic.

Issues List

- [How to Report a Bug](#)
- [How to Request a Feature](#)
- [How to Influence Priorities](#)
- [How to Track Development](#)
- [How to Track Releases](#)

Documentation

- [How to Improve Documentation](#)

Development

- [How to Make Code Contributions](#)

Community Involvement

- [How to Make Code Contributions](#)



IoT Platform Community

Get involved in the 3M IoT Platform Community.

Details coming soon.

Emails will go out announcing the first event that is scheduled to start between October 15th and the 23rd.



How to Request a Feature

To request a feature to improve the 3M IoT Platform on Azure, follow the steps below:

1. Open the [Issues List](#)
2. Navigate to the Issues List
3. Click **New issue**
4. Click **Get started** on the Feature request row
5. Populate the request (See section: *How to Complete Feature Request*)
6. Submit Issue
7. Track Progress (See [How to Track Progress](#))
8. Add Tag Name for the group requesting: e.g. CHIM

How to Complete Feature Request

- Title
- Describe Need
- Describe Improvement
- Describe Alternatives
- Attachments



How to Report a Bug

To report a bug related to the 3M IoT Platform on Azure, follow the steps below:

1. Open the [Issues List](#)
2. Navigate to the Issues List
3. Click **New issue**
4. Click **Get started** on the Bug report row
5. Populate the report (See section: How to Complete Bug Report)
6. Submit Issue
7. Track Progress (See [How to Track Progress](#))
8. Add Tag Name for the group requesting: e.g. CHIM

How to Complete Bug Report

A Bug Report consists of the following sections:

- Title
- Description
- Steps to Reproduce
- Expected Behavior
- Screenshots
- Desktop
- Additional context
- Attachments

The remaining sections will describe how to complete each section.

Title

TODO

Description

Steps to Reproduce

Expected Behavior

Screenshots

Desktop

Additional context

Attachments



How to Influence Priorities

This article describes how you can influence platform improvements by voting on issue prioritization.



How to Track Progress



How to Track Releases

Versions record what improvements are associated with a given release. When a new release is deployed to your environment, the version number is reflected in the UI. Versions organize platform improvements into identifiable releases. What's included in each version is recorded in the ChangeLog which is publicly available on [GitHub](#). A new version must be deployed to each environment. As such, environments typically run on different releases.

How to check the current version

To see the list of improvements by version, open the [Change Log](#).

How to check the improvements per version

From the releases page, click the link to the changelog for the version you're interested in or simply click the first link. Improvements for all other versions will also be listed. Here's what it looks like:

How to check which version you're running

The easiest way to identify which version an environment has deployed is through the user interface. By clicking on the gear icon in the upper right, the current version is easily identified as depicted below:

The link ***View release notes*** offers a quick way to open the ChangeLog on the public repo.



How to Improve Documentation



Getting Started with Contributing Code

Enhancing the 3M IoT Platform on Azure requires development tools. At a minimum, contributing to the platform requires one or both of the following Integrated Development Environments (IDE):

- Visual Studios 2019 (or above)
- VS Code

The tools and skills necessary to contribute are generally influenced by the following contribution areas:

1. User Interface (UI)
2. APIs
3. Backend Services
4. CI/CD: Build, Test and Deploy
5. Documentation

Contribution Areas

Although many of the tools overlap by area, the following table offers a simplified version of the differences required by each area.

AREA	TOOLS	SKILLS
UI	VS Code	ReactJS, CSS, JavaScript
APIs	Visual Studio, DevSpaces	C#, REST, Docker
Backend	Visual Studio	C# and Azure IoT Hub, Storage, Cosmos DB
CI/CD	VS or VS Code	DevOps, YAML, Azure Pipelines, HELM
Documentation	VS Code	Markdown

Common Roles

Specific details for each contribution area are organized according to the following roles:

- [Web Developer](#)
- [API Developer](#)
- [Azure Developer](#)
- [DevOps Engineer](#)
- [Documentation Contributor](#)

Alternatively, [click here](#) to install all tools for platform contribution.

Next Step

Once you're done setting up the appropriate tools, perform the One-Time-Setup Instructions to begin working with the solution locally.



Getting Started for Web Developers

This document provides setup instructions for contributing to the 3M IoT Platform as a *Web Developer*. In addition, it outlines relevant skills and helpful links to learn about or brush up on related topics.

Useful Skills

Be familiar with the following:

- JavaScript
- CSS/Sass
- ReactJS (16.6.3)
- Redux (5.1.1)
- IoT
- [Azure IoT UX Fluent Controls](#)
- Git

Common Contributions

Web Developers typically contribute to the platform in the following ways:

- Enhancing the web User Interface
- Communicating with the Backend Services

Tools to Install

- [VS Code](#) or other IDEs
- Node 10.14.1 (or above)
- NPM 6.4.1 (or above)
- Git

Helpful Resources

- [IoT School](#)
- [Microsoft IoT](#)
- [ReactJS](#)
- [Redux \(a ReactJS event management tool\)](#)
- Fluent UI: [IoT Controls](#)
- [Git](#)



Getting Started for API Developers

This document provides setup instructions for contributing to the 3M IoT Platform as an *API Developer*. In addition, it outlines relevant skills and helpful links to learn about or brush up on related topics.

Useful Skills

- C#
- .Net core
- IoT
- Azure services- IoT Hub, Azure Storage, Cosmos DB
- Containerization
- Dev spaces (optional)
- Azure functions
- Kubernetes (optional)
- Docker (optional)
- Git

Common Contributions

API Developers typically contribute to the platform in the following ways:

- Updating/enhancing back-end services
- Building service to interact with IoT Hub to configure devices and provide configuration updates to devices via IoT Hub
- Building Analytics to check for anomalies
- Services to store data for Audit

Tools to Install

- Visual Studio 2019 (or above) / VS Code
- Docker desktop (optional)
- Azure CLI (optional)
- Git

Helpful Resources

- [IoT School](#)
- [Microsoft IoT](#)
- [Azure CLI](#)
- [Azure Storage](#)
- [Cosmos DB](#)
- [Azure IoT Hub](#)
- [Git](#)



Getting Started for Azure Developers

This document provides setup instructions for contributing to the 3M IoT Platform as a *Azure Developer*. In addition, it outlines relevant skills and helpful links to learn about or brush up on related topics.

Useful Skills

Be familiar with the following list of Azure services

- AKS
- Storage Account
- DPS (Device Provisioning Service)
- Function App
- App Configuration
- Cosmos DB
- Event Hub
- Key Vault
- Send Grid
- IOT Hub
- Stream Analytics Job
- Azure Maps

Common Contributions

Azure Developer typically contribute to the platform in the following ways:

- Should have knowledge on Azure Services and how to deploy code to these services.
- Enhancing the Azure Services and integration between the services.

Tools

- Az cli
- Azure Storage Explorer
- Azure Devspaces cli and extension
- Azure IOT Explorer
- VS Code
- Docker

Helpful Resources

- [IoT School](#)
- [Microsoft IoT](#)
- [Azure Storage Explorer](#)
- [Az Cli](#)
- [Devspaces](#)



Getting Started for DevOps Engineers

This document provides setup instructions for contributing to the 3M IoT Platform as a *DevOps Engineer*. In addition, it outlines relevant skills and helpful links to learn about or brush up on related topics.

Useful Skills

- Azure Services
 - AKS
 - Storage Account
 - DPS (Device Provisioning Service)
 - Function App
 - App Configuration
 - Cosmos DB
 - Event Hub
 - Key Vault
 - Send Grid
 - IOT Hub
 - Stream Analytics Job
 - Azure Maps
 - Application Insights
 - Application Gateway
 - Log Analytics
 - Azure Automation Account
 - Virtual Network
 - Network Security Group
 - DNS Zone
 - Load Balancer
 - VM Scale set
- Azure Devops
- Powershell
- Bash
- Kubernetes
- Helm
- Docker
- Docker Compose

Helpful Resources

- [Microsoft IoT](#)
- [Azure Cli](#)
- [Azure IOT Explorer](#)
- [Git](#)
- [Git Desktop](#)
- [Helm](#)
- [Kubernetes](#)
- [Storage Explorer](#)
- [Devspaces](#)
- [Azure Devops](#)

Common Contributions

DevOps Engineer typically contribute to the platform in the following ways:

- Enhancing existing CI/CD pipelines.
- Creating new pipelines if there are any
- Creating the infrastructure in Azure through Infrastructure as a Code.

Tools

- Az cli
- Azure Storage Explorer
- Azure Devspaces cli and extension
- Azure IOT Explorer
- VS Code
- Docker
- Bash
- Kubernetes
- Helm
- Git
- GitHub Desktop
- Docker Desktop
- Docker Compose
- Powershell
- VS Code



Getting Started for Document Contributors

This document provides setup instructions for contributing to the 3M IoT Platform as a *Document Contributor*. In addition, it outlines relevant skills and helpful links to learn about or brush up on related topics.

Useful Skills

- Markdown
- IoT (optional)
- Git

Common Contributions

Documentation Contributors typically contribute in the following ways:

- Add new content to the documentation
- Improve or organize the existing documentation

Tools to Install

- VS Code
- [Docfx](#)
- Git

Helpful Resources

- [Docfx User manual](#)
- [Docfx - API Documentation](#)
- [Docfx - Rest API](#)
- [IoT School](#)
- [Microsoft IoT](#)
- [Git](#)



3M IoT Development Environment Setup Instructions

This document provides detailed documentation on setting up a development environment to contribute to 3M's IoT Azure Platform, called Bluebird. Note: If you're using a 3M Virtual Machine, it's possible some of the products to install may exist already.

Products to Install

Please install the following products in the order listed below. Click the project name to open the specific steps for each product below. Wherever relevant, OS specific steps will be provided for both Mac and PC. I would like to also point out that we are not allowed to be admins on our 3M machines, we use elevated access. This is accomplished by right clicking and selecting the elevated access option.

Note: Install Issues not specifically addressed in the specific product install guides below should be recorded here. Since we have moved into an open source environment for Odin it is not unreasonable to use our own machines for dev. Sometimes this proves less restrictive and has better up-time.

Product Install Guides (by install order)

- [Virtualization](#) (only necessary if developing on Windows)
- [.NET Core 3.1](#)
- [PowerShell](#)
- [Azure CLI](#) (plus extensions)
- [Visual Studio Code](#)
- [Git](#) / [Bash](#)
- [NodeJS](#) / [npm](#)
- [Visual Studio 2019](#)
- [Azure Data Studio](#)
- [Storage Explorer](#)
- [Azure IoT Explorer](#)
- [Docker](#)
- [Terraform](#)
- [Kubernetes](#)
- [Helm](#)
- [Redux](#)
- [Postman](#)
- [Source Code](#)



Whats new in VS 2019

Check the details [here](#)

Use the community edition if you don't have a licensed version: <https://visualstudio.microsoft.com/downloads/>

Download

Download Visual Studio 2019 from [here](#)



Overview

If your development environment runs on Windows you must setup virtualization in order to run docker containers locally since some of them run on Linux which Windows does not natively support. This is not necessary on a MacOS since it's based on Unix. Which virtualization product you use is a matter of preference and is typically influence by the OS of your host machine. Historically, Hyper-V was popular for servers and VirtualBox on clients.

Virtualization Product

PRODUCT	VENDOR	GUIDES
Hyper-V	Microsoft	MS Install Guide
Virtual Box	Oracle	- Docker for Mac - Using Docker with Virutal Box on Windows

Prepare Virtual Machine using Hyper-V

1. Enable Hyper-V. [Click for more details](#).
2. Create a Virtual Machine. [Click for more details](#). Note: Microsoft offers instances of Ubuntu and an evaluation copy of Windows. Currently the evaluation only last 5 days so it's not a great option unless you have an Win 10 Enterprise Key to use to upgrade. Otherwise you need an ISO and License Key for the OS you intend to install. Using the eval requires a 16GB download.
3. Prepare Installation Media
4. Create Virtual Machine

Prepare Installation Media

If you're going to use your own copy of an operating system, you need to first create an ISO so Hyper-V or VirtualBox can use it to build the Virtual Machine. Microsoft provides a useful tool called **Create Windows 10 installation media** that can be downloaded here <https://www.microsoft.com/en-us/software-download/windows10>. The link includes instructions on how to use.

The above approach can also be used for Windows 8.1 and Windows 7. Alternatively there are several open source and 3rd party products that can be used to create ISO.

Create Virtual Machine

Once you have your ISO, you can start the create virtual machine process. To continue, follow these steps: <https://docs.microsoft.com/en-us/virtualization/hyper-v-on-windows/quick-start/quick-create-virtual-machine>

Alternative

As untried alternative to virtualization, if you're using Windows you could try the Linux subsystem for Windows. Note: We have not tried this. Key consideration is whether or not docker images can be instantiated.

Use Windows Subsystem for Linux for production: <https://docs.microsoft.com/en-us/windows/nodejs/setup-on-windows#use-windows-subsystem-for-linux-for-production>

The Windows Subsystem for Linux, introduced in the [Anniversary Update](#), became a stable feature in the [Fall Creators Update](#). You can now run Ubuntu and openSUSE on Windows, with Fedora and more Linux distributions coming soon.

This documented might be a good guide: <https://docs.microsoft.com/en-us/virtualization/windowscontainers/quick-start/set-up-environment?tabs=Windows-10-Client>



Overview

.NET Core is a cross-platform successor to the .NET Framework that runs on Windows, Linux, and macOS operating systems. You can download from [here](#). It was open sourced by Microsoft.

TODO

Did you choose 3.1.0 or 3.1.3 (latest as of 4/8)?



Overview

PowerShell is a command-line shell and associated scripting language that can run on on Windows, Linux and macOS as of version 7.

Installation Guides

- [Windows](#)
- [macOS](#)
- [Linux](#)
- [Installing PowerShell in Azure Resources via ARM](#)

To access the download packages go here: <https://github.com/PowerShell/PowerShell>

Additional

Windows PowerShell 7 is automatically part of Windows 10 IoT Enterprise. Additional details look here:

<https://docs.microsoft.com/en-us/powershell/scripting/install/installing-powershell-core-on-windows?view=powershell-7#deploying-on-windows-10-iot-enterprise>



Overview

The Azure command-line interface (Azure CLI) is a set of commands used to create and manage Azure resources. The Azure CLI is available across Azure services and is designed to get you working quickly with Azure, with an emphasis on automation.

Click [here](#) to get more details Be sure to add [additional extensions, see below](#).

Note: Requires Elevated Permissions on 3M's Virtual Machine

Install the Azure CLI

The Azure CLI is available to install in Windows, macOS and Linux environments. It can also be run in a Docker container and Azure Cloud Shell.

The current version of the Azure CLI is **2.3.1**. For information about the latest release, see the [release notes](#). To find your installed version and see if you need to update, run `az --version`.

- [Install on Windows](#)
- [Install on macOS](#)
- Install on Linux or [Windows Subsystem for Linux \(WSL\)](#)
 - [Install with apt on Debian or Ubuntu](#)
 - [Install with yum on RHEL, Fedora, or CentOS](#)
 - [Install with zypper on openSUSE or SLE](#)
 - [Install from script](#)
- [Run in Docker container](#)
- [Run in Azure Cloud Shell](#)

Use extensions with Azure CLI

The Azure CLI offers the capability to load extensions. Extensions are Python wheels that aren't shipped as part of the CLI but run as CLI commands. With extensions, you gain access to experimental and pre-release commands along with the ability to write your own CLI interfaces.

Click [here](#) to get details on how to use extensions

Additional Extensions

Add the extensions listed below. To do so, use the following command:

```
az extension add --name {extension_name}
```

To see the list of available extensions and their details such as if they're enabled, run the following:

```
az extension list-available
```

Note: if any of your extensions need to be updated, you can swap the **add** commands with **update**.

Dev Spaces

- **Description:** Dev Spaces provides a rapid, iterative Kubernetes development experience for teams.
- **Command:** `az extension add --name dev-spaces`

- [Documentation](#)
 - **Note:** there is a preview version as well. dev-spaces-preview

IoT

- **Description:** Comprehensive data-plane functionality to manage Internet of Things (IoT) assets.
- **Command:** `az extension add --name azure-iot`
- [Documentation](#)

DevOps

- **Description:** Manage Azure DevOps organization level operations including pipelines, boards, repos, artifacts, etc.
- **Command:** `az extension add --name azure-devops`
- [Documentation](#)

Azure Kubernetes Service (AKS)

- **Description:** Manage Azure Kubernetes Services.
- **Command:** `az extension add --name aks-preview`
- [Documentation](#)

Introduction

Git is a distributed version-control system for tracking changes in source code during software development. It is designed for coordinating work among programmers, but it can be used to track changes in any set of files. Its goals include speed, data integrity, and support for distributed, non-linear workflows. It is designed to handle everything from small to very large projects with speed and efficiency.

Pre-requisites


Be sure to installed VS Code first. Then you'll have the option of hooking Git into VS Code as you're editor.

If you're going to use recommended [NVM](#) (Node Version Manager) on Windows, you'll need to **Enable symbolic links** during the installation. See settings section, install step **Configuring extra options** below for more information.

Git

Click [here](#) to download git for different OS.

Settings

Use the default values except on the screenshots below. For those steps, use the settings indicated in the screenshot (assuming Win Install). 



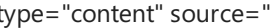
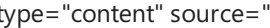








GitHub Desktop (optional)

Click [here](#) to download GitHub Desktop for macOS, Win, and Linux.

Connecting Your GitHub Repository to 3M

1. The repository you are look for is <https://github.com/3Mcloud/azure-iot-platform-dotnet/>
2. You will then select Fork in the top right hand corner as displayed below
 - 
3. This will then allow you to tie your personal account to the 3m cloud
4. You are able to verify that you are associated by clicking the number next to fork. It will display the different accounts attached as shown below.
 - 
5. You can see mine is highlighted, using this method you are able to raise PR's and also submit Reviews. Keep in mind we need two reviews for QA purposes.
6. You can also see what your teammates have been working on by clicking on their account.
7. You are able to sync data from your repo to the 3m master. There are 2 main ways to do this.
 - This way is done by the cli.
 - <https://help.github.com/en/github/collaborating-with-issues-and-pull-requests/syncing-a-fork>
 - This is the way to do it through the web gui. I think this is the ideal way to do this. 

shots.

- <https://youtu.be/YhwBgYPfoVE>
 - As you can see my repo is 15 commits behind.
 -  Pull request 1
 - After comparing it will tell me I am able to merge.
 -  Pull request 2
 - You would then submit a PR to sync your fork.
 -  Pull request 3

CLI Approach to Refresh Local Master with Upstream Changes

Note: This approach will completely replace your local master (forked 3M repo) with changes upstream (3M Cloud):

```
git remote add upstream /url/to/original/repo
```

```
git fetch upstream
```

```
git checkout master
```

```
git reset --hard upstream/master
```

```
git push origin master --force
```

CLI Approach to Creating New Branch

By on a new copy of "master" (using above approach). Then create a new branch

```
git checkout -b 1245-fix(webui)-deployment-flyout
```


Introduction

Bash is the shell, or command language interpreter, for the GNU operating system. The name is an acronym for the 'Bourne-Again SHell', a pun on Stephen Bourne, the author of the direct ancestor of the current Unix shell sh, which appeared in the Seventh Edition Bell Labs Research version of Unix. Bash is largely compatible with sh and incorporates useful features from the Korn shell ksh and the C shell csh. It is intended to be a conformant implementation of the IEEE POSIX Shell and Tools portion of the IEEE POSIX specification (IEEE Standard 1003.1). It offers functional improvements over sh for both interactive and programming use. While the GNU operating system provides other shells, including a version of csh, Bash is the default shell. Like other GNU software, Bash is quite portable. It currently runs on nearly every version of Unix and a few other operating systems - independently-supported ports exist for MS-DOS, OS/2, and Windows platforms.

The improvements offered by Bash include:

- Command line editing
- Unlimited size command history
- Job Control
- Shell Functions and Aliases
- Indexed arrays of unlimited size
- Integer arithmetic in any base from two to sixty-four

Download


There are many ways to install bash. A couple options are as following:

GNU

Bash can be found on the main GNU ftp server: <http://ftp.gnu.org/gnu/bash/> (via HTTP) and <ftp://ftp.gnu.org/gnu/bash/> (via FTP). It can also be found on the [GNU mirrors](#); please [use a mirror](#) if possible.

Click [here](#) for more details

Install with Git

You can get bash on windows by [installing GIT](#). This might be the easiest approach for developers using Windows. During the install, choose Windows Explorer integration: :::image type="content" source="../../images/git-install.PNG" alt-text="Git install":::

VS Code

Check out this: <https://stackoverflow.com/questions/42606837/how-do-i-use-bash-on-windows-from-the-visual-studio-code-integrated-terminal>

Overview

NodeJS is a extension from [Joyant](#) that builds on Google's V8 Engine that offers a very fast run-time enviornment for solutions built using JavaScript that can run both client-side (web and desktop - via Electron) and server side. The speed comes from Google's non-blocking IO model.

BE SURE TO TARGET VERSIONS: **NPM** 6.4.1 **NODE** 10.14.1 (local) 11.1 (prod)

We recommended using Node Version Manager (NVM) for Windows to be able to support switching between NodeJS versions (which also influences the active NPM version). We have found install issues on some Windows 10 Machines running McAfee. Follow these instructions to avoid install issues: <https://medium.com/@tysonpaul89/maintain-multiple-versions-of-node-js-in-windows-operating-system-using-nvm-3c6bf5b63f29>

Alternatively, to install NodeJS directly go to <https://nodejs.org/>, but you want be able to switch versions so be sure to install the correct version (see above).

Additional details about configuring and switching versions using NVM are below

Another helpful guide is: [Set up your Node.js development environment directly on Windows](#)

NVM Alternatives

If you have install issues with NVM or are running on a non-windows machine, try the following:

While windows-nvm is currently the most popular version manager for node, there are alternatives to consider:

[nvs](#) (Node Version Switcher) is a cross-platform nvm alternative with the ability to [integrate with VS Code](#).

[Volta](#) is a new version manager from the LinkedIn team that claims improved speed and cross-platform support.

To install Volta as your version manager (rather than windows-nvm), go to the **Windows Installation** section of their [Getting Started guide](#), then download and run their Windows installer, following the setup instructions.

Important: You must ensure that [Developer Mode](#) is enabled on your Windows machine before installing Volta.

To learn more about using Volta to install multiple versions of Node.js on Windows, see the [Volta Docs](#).

Node Version Manager (NVM)

Use Node Version Manager (NVM) to support multiple versions:

- Windows: <https://github.com/coreybutler/nvm-windows>
- Linux and MAC: <https://github.com/nvm-sh/nvm>

Note

This solution uses older versions of NodeJS. Uses older versions will cause warning to be triggered like:

```
npm WARN npm npm does not support Nodejs vXX.XX.XX These can be ignored.
```

It's best to do uninstall any direct node installs before installing NVM. Even though the install on windows seems to merge existing installs, you'll likely still run into issues with the Node Package Manager (NPM). See [Uninstall Existing Node Documentation](#) below:

Once installed, run the following to install the latest version of NodeJS:

Useful NVS Commands

List Versions currently installed

```
"engines" : { "node" : ">=0.12" }  
list
```

Install version

```
nvm install "latest"  
or  
nvm install 10.1
```

Set current version

```
nvm use 10.1
```

Set NPM Versions

You can change your NPM Versions up or down using the following

Getting Lastest NPM

Node comes with npm installed so you should have a version of npm. However, npm gets updated more frequently than Node does, so you'll want to make sure it's the latest version.

```
npm install npm@latest -g
```

Targeting Specific NPM Version

Just replace @latest with the version number you want to downgrade to. I wanted to downgrade to version 3.10.10, so I used this command:

```
npm install -g npm@3.10.10
```

Note: If you do a lot of version switching, you might run into the following issue:

Error: Node Sass does not yet support your current orted please see: environment: Windows 64-bit with Unsupported runtime1.0e

There's an easy fix:

```
npm rebuild node-sass
```

Additional information can be found [here](#). This approach requires python to be installed.

Uninstall Existing Node and NPM Versions

Uninstall existing Node Version

Please note, you need to uninstall any existing versions of node.js before installing NVM for Windows. Also delete any existing nodejs installation directories (e.g., "C:\Program Files\nodejs") that might remain. NVM's generated symlink will not overwrite an existing (even empty) installation directory.

Uninstall existing NPM Version

You should also delete the existing npm install location (e.g. "C:\Users<user>\AppData\Roaming\npm"), so that the nvm install location will be correctly used instead. Backup the global `npmrc` config (e.g.

`C:\Users\<user>\AppData\Roaming\npm\etc\npmrc`), if you have some important settings there, or copy the settings to the user config `C:\Users\<user>\.npmrc`.

Helpful Tools

TOOL	DESCRIPTION	NOTES
NPM-Check	Display package versions and if updates are available.	Warning: Careful on making updates. Those are sweeping changes that shouldn't be taken lightly (unless of course you're not installing into the solution or changing dependencies) .

ESLint Issues

Sometimes ESLint causes a lot of issues. This often happens on a fresh clone that came from a repo built by a different OS. This issue can be easily resolved by running:

```
npm run lint -- --fix
```



Overview

npm (Node Package Manager) is a package manager for the JavaScript programming language. It is the default package manager for the JavaScript runtime environment Node.js. It consists of a command line client, also called npm, and an online database of public and paid-for private packages, called the npm registry. The registry is accessed via the client, and the available packages can be browsed and searched via the npm website. The package manager and the registry are managed by npm, Inc.

Command Line Client

npm includes a **CLI** (Command Line Client) that can be used to download and install software:

Windows Example

```
C:\>npm install <package>
```

Mac OS Example

```
>npm install <package>
```

Installing npm

npm is installed with Node.js. This means that you have to install Node.js to get npm installed on your computer. Download Node.js from the official Node.js web site: <https://nodejs.org>



Overview

A light-weight editor that can run on-demand SQL queries, view and save results as text, JSON, or Excel. Edit data, organize your favorite database connections, and browse database objects in a familiar object browsing experience.

Download

Azure Data Studio can be downloaded from [here](#).



Overview

Upload, download, and manage Azure blobs, files, queues, and tables, as well as Azure Cosmos DB and Azure Data Lake Storage entities. Easily access virtual machine disks, and work with either Azure Resource Manager or classic storage accounts. Manage and configure cross-origin resource sharing rules.

Download

Storage Explorer can be downloaded from [here](#)



Introduction

The Azure IoT explorer is a graphical tool for interacting with and testing your IoT Plug and Play Preview devices. After installing the tool on your local machine, you can use it to connect to a device. You can use the tool to view the telemetry the device is sending, work with device properties, and call commands.

Download

Azure IoT Explorer can be downloaded from [here](#)

Install and use Azure IoT explorer

Click [here](#) to know how to install and use Azure IoT explorer

Quick Demo

Check out this video for a quick e2e [demo](#).



Installation guide for Docker

Docker is a set of platform as a service products that uses OS-level virtualization to deliver software in packages called containers. Containers are isolated from one another and bundle their own software, libraries and configuration files; they can communicate with each other through well-defined channels. To install it, following the guides below specific to your operating system.

Docker on Mac

Installation instructions for installing Docker on a Mac can be found [here](#).

Docker on Windows

What to know before installation

- Windows 10 64-bit: Pro, Enterprise, or Education (Build 15063 or later).
- Hyper-V and Containers Windows features must be enabled.
- The following hardware prerequisites are required to successfully run Client Hyper-V on Windows 10:
 - 64-bit processor with [Second Level Address Translation \(SLAT\)](#)
 - 4GB system RAM
 - BIOS-level hardware virtualization support must be enabled in the BIOS settings. For more information, see [Virtualization](#).

For additional information about requirements see: [What to know before you install](#).

Steps to setup Docker on Windows

- Download Docker Desktop <https://hub.docker.com/editions/community/docker-ce-desktop-windows/>
- Double-click Docker Desktop Installer.exe to run the installer.
- If you haven't already downloaded the installer (Docker Desktop Installer.exe), you can get it from [Docker Hub](#). It typically downloads to your Downloads folder, or you can run it from the recent downloads bar at the bottom of your web browser.
- Follow instructions on the installation wizard to accept the license, authorize the installer, and proceed with the install.
- When prompted, authorize the Docker Desktop Installer with your system password during the install process. Privileged access is needed to install networking components, links to the Docker apps, and manage the Hyper-V VMs.
- Click Finish on the setup complete dialog and launch the Docker Desktop application.

For additional information, see [Install Docker Desktop on Windows](#). To test your installation, follow [this guide](#).

Alternative Setup

It might now be possible to use Docker on Windows without having to virtualize thanks to the release of the Linux subsystem for Windows. This approach has not been tested to determine if it will work for this solution. It allows you to run Ubuntu and openSUSE on Windows, with Fedora and more Linux distributions coming soon.

As untried alternative to virtualization, if you're using Windows you could try the Linux subsystem for Windows.

Note: We have not tried this. Key consideration is whether or not docker images can be instantiated.

This documented might be a good guide: <https://docs.microsoft.com/en-us/virtualization/windowscontainers/quick-start/set-up-environment?tabs=Windows-10-Client>

For additional information on the Linux Subsystem: The Windows Subsystem for Linux, introduced in the [Anniversary Update](#), became a stable feature in the [Fall Creators Update](#).



Installation guide for Terraform

Steps to set up terraform on Windows:

- To install Terraform, find the [appropriate package](#) for your system and download it. Terraform is packaged as a zip archive.
- After downloading Terraform, unzip the package. Terraform runs as a single binary named terraform. Any other files in the package can be safely removed and Terraform will still function.
- The final step is to make sure that the terraform binary is available on the PATH.
- Verify the installation by executing terraform on new terminal session

Installation guide for Kubernetes

Steps to set up a kubectl on Windows

- Download the kubectl.exe using a [link](#) and save the file in any folder on windows file system.
- Add the kubectl.exe folder location in path variable - "Advanced System Settings -> Advanced -> Environment Variables -> Path". For example, if you have saved file to C:/kube then add this folder path to the path variable.
- Open a command prompt and type kubectl and you should see all commands supported by kubectl.

Install minikube *(Not Mandatory)

- VT-x or AMD-v virtualization must be enabled in your computer's BIOS.
- Install the virtualization platform such as Virtualbox or KVM. You are not really required to configure the image.
- Download the minikube-windows-amd64 file from [here](#).
- Add this folder path location in path variable: "Advanced System Settings -> Advanced -> Environment Variables -> Path." For example, if you have saved the file to C:/kube then add this folder path to the path variable.
- Open the command prompt and fire a command minikube and you should see all the commands supported by minikube.

On windows, you can get similar kind of linux kind of user experience with Cygwin. Install Cygwin by following the steps listed on [its website](#).

Start minikube :

- Open the Cygwin terminal and run command \$ minikube start.
- Run command kubectl version to confirm the working of minikube.



Installation guide for Helm

Prerequisites

- You will need the command line program kubectl installed on your Windows 10 computer and configured to work with a Kubernetes Cluster.
- 7-Zip compression / decompression program is needed to extract the Helm program for Windows from the compressed file-folder from the Helm site. You can download it here: <https://www.7-zip.org/download.html>

Steps to Install

- Download the latest version of the compressed executable from the Helm GitHub site, <https://github.com/kubernetes/helm/releases>.
- Navigate to the folder you downloaded the helm-vX.X.X-windows-amd64.tar.gz compressed file from and move the file to its own directory.
- Navigate to the new directory and right click on the tar.gz file and with 7Zip, open the tar.gz archive.
- Double click the single tar file in that directory, helm-v2.7.2-windows-amd64.tar.
- You should now see a windows file folder in the 7Zip window, windows-amd64. Right click on the folder, select Copy To, and select the directory you want to copy the folder to.
- add the helm program to the System File path to make it easily accessible from the command line.
- Open the Control Panel's System panel.
- Select the Advanced system settings link on the left.
- Select Environment Variables.
- Under System variables, select Path, and then select Edit.
- Select the New button and then add the folder path where you copied the helm folder to and then press OK.
- Open a new command line window and type helm on the command line to make sure you have access to helm from the command line.
- Assuming you have the kubectl program configured for your Kubernetes cluster you can now initialize helm.
- Now you are ready to deploy Kubernetes applications to your kube cluster.

Note: Ubuntu and an evaluation copy of Windows. Otherwise you need an ISO and License Key for the OS you intend to install. Using the eval requires a 16GB download.



Overview

Redux is an open-source JavaScript library for managing application state. It is most commonly used with libraries such as React or Angular for building user interfaces. It was created by Dan Abramov and Andrew Clark.

Redux is a predictable state container for JavaScript apps.

It helps you write applications that behave consistently, run in different environments (client, server, and native), and are easy to test. On top of that, it provides a great developer experience, such as [live code editing combined with a time traveling debugger](#).

You can use Redux together with [React](#), or with any other view library. It is tiny (2kB, including dependencies), but has a large ecosystem of addons available.

Installation

To install the stable version:

`npm install redux` This assumes you are using [npm](#) as your package manager.

If you're not, you can [access these files on unpkg](#), download them, or point your package manager to them.

Click [here](#) to get more details



Introduction

Postman is a collaboration platform for API development. Postman's features simplify each step of building an API and streamline collaboration so you can create better APIs—faster.

Get more detail [here](#)

Installation and updates

Postman is available as a native app for Mac, Windows (32-bit / 64-bit), and Linux (32-bit / 64-bit) operating systems.

To get the latest version of the Postman app, visit the [download page](#) and click Download for your platform.

Installing Postman

- [Mac](#)
- [Windows](#)
- [Linux](#)



Installing and Configuring 3M's Azure IoT Source Code

Assuming you've completed the major installs required to run the source code, you're now ready to clone the 3M repo and configure the solution on your local development environment.

To do so, requires the following steps:

1. Clone Repo
2. Configure Environment Settings
3. Update Source Code Dependencies
4. Build and Run

You might also want to watch the setup video: https://www.youtube.com/watch?v=asICCW_1uIl

Grant KeyVault Access to Project

```
dotnet user-secrets set --project C:{your path}\src\services\common\Services\Services.csproj AppConfiguratonConnectionString
"Endpoint=https://crsliot-appconfig-dev.azconfig.io;Id=AMGf-l4-
s0:rAtTcp3u1hOQszVB49Tg;Secret=R4VYRXHcDx8FqQxc9xNC4F16Mxt0yG4FReBFVxQF7Is="
```

TODO

Organize content into the sections below and provide coverage of how/when to use: Icons in React JS Fluent Framework:

<https://developer.microsoft.com/en-us/fluentui#/styles/web/icons>

```
<crsliot-aks-dev.centralus.cloudapp.azure.com>
```

Web UI:

```
dotnet user-secrets set --project C:{your path}\src\services\common\Services\Services.csproj AppConfiguratonConnectionString
"Endpoint=https://crsliot-appconfig-dev.azconfig.io;Id=AMGf-l4-
s0:rAtTcp3u1hOQszVB49Tg;Secret=R4VYRXHcDx8FqQxc9xNC4F16Mxt0yG4FReBFVxQF7Is="
```

```
.env NODE_PATH src/
```

App config:

```
const baseUrl = "https://crsliot-aks-dev.centralus.cloudapp.azure.com";//process.env.REACT_APP_BASE_SERVICE_URL || "";
```

```
dotnet user-secrets set --project C:\DevOps\OpenSource\azure-iot-platform-dotnet\src\services\common\Services\Services.csproj
AppConfiguratonConnectionString "Endpoint=https://crsliot-appconfig-dev.azconfig.io;Id=AMGf-l4-
s0:rAtTcp3u1hOQszVB49Tg;Secret=R4VYRXHcDx8FqQxc9xNC4F16Mxt0yG4FReBFVxQF7Is="
```

```
dotnet build mmm.iot.sln C:\DevOps\OpenSource\azure-iot-platform-dotnet\webui
```

<https://github.mmm.com/mmm/azure-iot-services-dotnet>

```
npm install
```

```
npm start
```

Clone Repo

There are many ways to clone the 3M Repo. Doing so depends on having Git installed. 3M IoT Platform on Azure - © 3M 2020

- Internal Repo: <https://github.mmm.com/mmm/azure-iot-services-dotnet>
- Public Repo: <https://github.com/3M-Company/azure-iot-platform-dotnet/>

Configure Environment Settings

Use the relevant sections below to configure environment Settings

Follow recommendations here: <https://github.com/3M-Company/azure-iot-platform-dotnet/blob/master/docs/DEVELOPMENT.md> More information on configuring environment variables here.

WebUI Environment Variables Use the section below that matches your environment.

- Windows
- MAC
- LINUX

Docker and Kubernetes Settings

WebUI Development Settings

It's necessary to set a few environment settings to get things up and running. Use the section below that matches your environment:

Update Source Code Dependencies

Build and Run

If you have SASS issues run:

```
npm run lint -- --fix
```

Check-in Changes

TODO: <https://designmodo.com/react-ci-cd/>

Services Development Settings

TODO: 3MC02YM21KJG5J:webui a9q25zz\$ export REACT_APP_BASE_SERVICE_URL="<https://crsliot-aks-dev.centralus.cloudapp.azure.com/>"

References

Azure IoT UX Fluent Controls	https://www.microsoft.com/design/fluent/	https://github.com/Azure/iot-ux-fluent-controls

Library version updates in 3M source code

Web UI section

update "node-sass" version to "4.13.1"

To install, use:

```
npm install node-sass@4.13.1
```




Guide

A PowerShell script for configuring DevSpaces is available [here](#). Works on Windows and macOS. Not sure about Linux. But we also have a bash version (under Files\Tools\DevSpaces). Additional documentation is [here](#).

Here are instructions on how to use:

To use this file, download DevSpaces.ps1 to your machine

Then edit DevSpaces.ps1 line 10 and change the \$script:MmmSourceDirectory variable to point to the containing folder of the Serenity repository

Then edit your PowerShell profile and add a line to "dot-source" DevSpaces.ps1 like so:

```
. 'C:\Users\aa30hzz\DevSpaces.ps1'
```

Then, restart your PowerShell Core terminal

Then, create a new DevSpace like so (creates the default/kyle DevSpace):

```
New-DevSpace -Name kyle -Parent default
```

Then, deploy Serenity to the DevSpace like so:

```
Start-SerenityDevSpace -Name kyle
```

Use Stop-SerenityDevSpace and Remove-DevSpace to stop the DevSpace and remove the DevSpace.

Scripts

NAME	WINDOWS	MACOS	LINUX
Configure Dev Spaces	DevSpaces.ps1	DevSpaces.ps1	new_dev_space.sh



Install Guide

You can download the tool from here: <https://xunit.net/>

XUnit is a unit testing tool for the .NET Framework



Prerequisites

Once you've installed the appropriate tools according to your role, you need to configure your environment to begin working with the IoT Platform source code. To do so, complete the following setup instructions.

One-Time Setup

Ensure the `AppConfigurationConnectionString` is set before building so that the `Mmm.Iot.Config.ClassGeneration` NuGet package can execute during build to automatically generate configuration classes in `./common/Services/Config` based on key-value pairs in Azure App Configuration and Azure Key Vault. This can be done in one of two ways:

1. [Set an environment variable](#)
2. Use `dotnet user-secrets` (recommended)

Either way, you will need to choose an Azure App Configuration instance and make note of its `<name>` and `<resource-group>` for use in the steps below.

Set an environment variable

Windows

In a PowerShell shell:

```
[System.Environment]::SetEnvironmentVariable('AppConfigurationConnectionString', (az appconfig credential list --name <name> --resource-group <resource-group> --query "[?name=='Primary'].connectionString | [0]" --output tsv), 'User')
```

Non-Windows

Set the `AppConfigurationConnectionString` environment variable in the Bash configuration file of your choice.

Use dotnet user-secrets

Windows

In a PowerShell shell:

```
dotnet user-secrets set --project ./src/services/common/Services/Services.csproj AppConfigurationConnectionString (az appconfig credential list --name <name> --resource-group <resource-group> --query "[?name=='Primary'].connectionString | [0]" --output tsv)
```

Then check the value of the secret:

```
dotnet user-secrets list --project ./src/services/common/Services/Services.csproj --json | Select-Object -Skip 1 | Select-Object -SkipLast 1 | ConvertFrom-Json | Select-Object -ExpandProperty AppConfigurationConnectionString
```

Non-Windows

In a Bash shell:

```
dotnet user-secrets set --project ./src/services/common/Services/Services.csproj AppConfigurationConnectionString `az appconfig credential list --name <name> --resource-group <resource-group> --query "[?name=='Primary'].connectionString | [0]" --output tsv`
```

Then check the value of the secret:

```
dotnet user-secrets list --project ./src/services/common/Services/Services.csproj --json | sed '1d;$d' | jq --raw-output '.AppConfigurationConnectionString'
```

Building

Build all services

```
dotnet build Mmm.Iot.sln
```

Build an individual service

```
dotnet build ./<service-name>/<service-name>.sln
```

E.g., to build the Storage Adapter service:

```
dotnet build ./storage-adapter/storage-adapter.sln
```

Build a Docker image for an individual service

You must provide a value for the `AppConfigurationConnectionString` environment variable to the Docker build. This value is a secret and must not be set directly in the Dockerfile via the `ENV` instruction. Therefore, you must provide the value in the `--build-args` option of the `docker build` command.

```
docker build --file ./<service-name>/WebService/Dockerfile --build-arg AppConfigurationConnectionString=$AppConfigurationConnectionString .
```

where `$AppConfigurationConnectionString` is either an environment variable or dotnet user secret.

E.g., to build the Storage Adapter container image:

```
docker build --file ./storage-adapter/WebService/Dockerfile --build-arg AppConfigurationConnectionString=$AppConfigurationConnectionString .
```

Running

Run all services

Azure DevSpaces

Non-Windows

First, setup your Azure DevSpaces for use

```
azds use \  
  --name <Name of your managed kubernetes cluster> \  
  --resource-group <Name of the resource group your cluster is within>
```

After setting up Azure DevSpaces, you can build and run each microservice in Azure DevSpaces. From the root of the project, run the script `up.sh`. To build and run each service in Azure DevSpaces.

```
./up.sh
```

Each service that is built and ran by `up.sh` has a unique url in the format of

`http://my-dev-space.s.default.service-name.ABC1234DE.cus.azure.io`. You can use this url to begin making requests to each service's API endpoints.

Run an individual service

The simplest is to use `dotnet run` to spin up a service on a random port on localhost:

```
dotnet run --project ./<service-name>/WebService/WebService.csproj
```

Debugging

Use either Visual Studio or Visual Studio Code



Working with 3M Cloud Repository

This documentation is about understanding the process of cloning the 3M cloud repository outside the 3M network to local machines and working with it.

1. Connect to Github with SSH
2. Clone solution
3. Creating new working branch
4. Commit and push the branch
5. Raise a pull request
6. PR Approval

Connect to Github with SSH

Using the SSH protocol, you can connect and authenticate to remote servers and services. Use the steps below for connecting to github with SSH

Check for existing keys/ Generate new key

Check for existing keys: Before generating a new SSH key, you can check if you have any existing keys from [here](#). Generate a new key if you don't have an existing public and private key pair, or don't wish to use any that are available to connect to GitHub.

Generate a new SSH key

Generate a new SSH key: If there is any existing SSH key, this step can be skipped. Follow the [steps](#) to generate a new key.

Add SSH key to ssh-agent

Add SSH key to the ssh-agent: Once you have the key ready, follow the steps [here](#) to add your SSH key to the ssh-agent.

Add SSH key to Github account

Add SSH key to Github account: Once the key is added to the ssh-agent, login to Github and follow the steps [here](#) in order to add your SSH key to Github account.

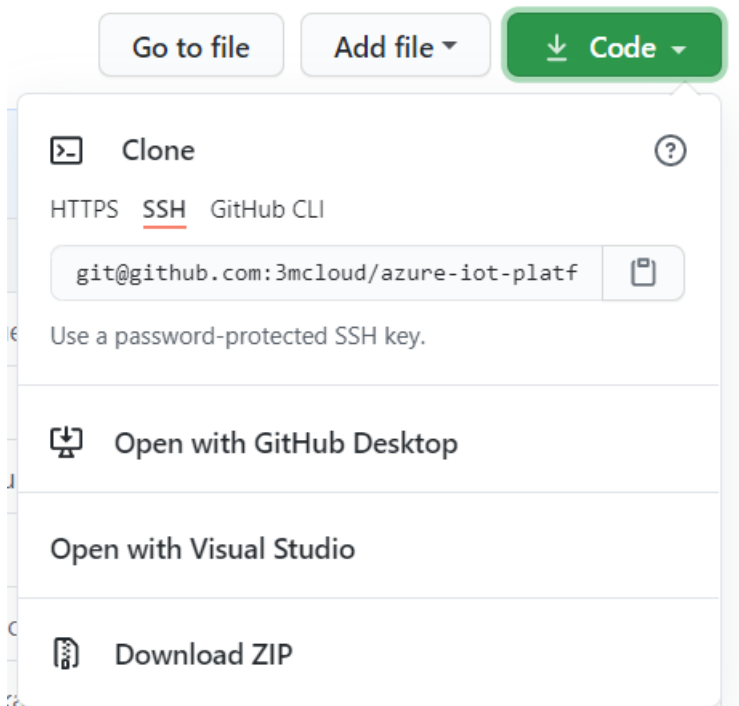
Testing the SSH connection

Testing the SSH connection: After you've set up your SSH key and added it to your GitHub account, you can test your connection by following the steps [here](#)

Clone Solution

To clone the solution, do the following:

1. Login to Github
2. Navigate to 3M cloud [azure-iot-platform-dotnet](#) repository
3. Click on Code button as shown below, switch to SSH and copy the path to clone



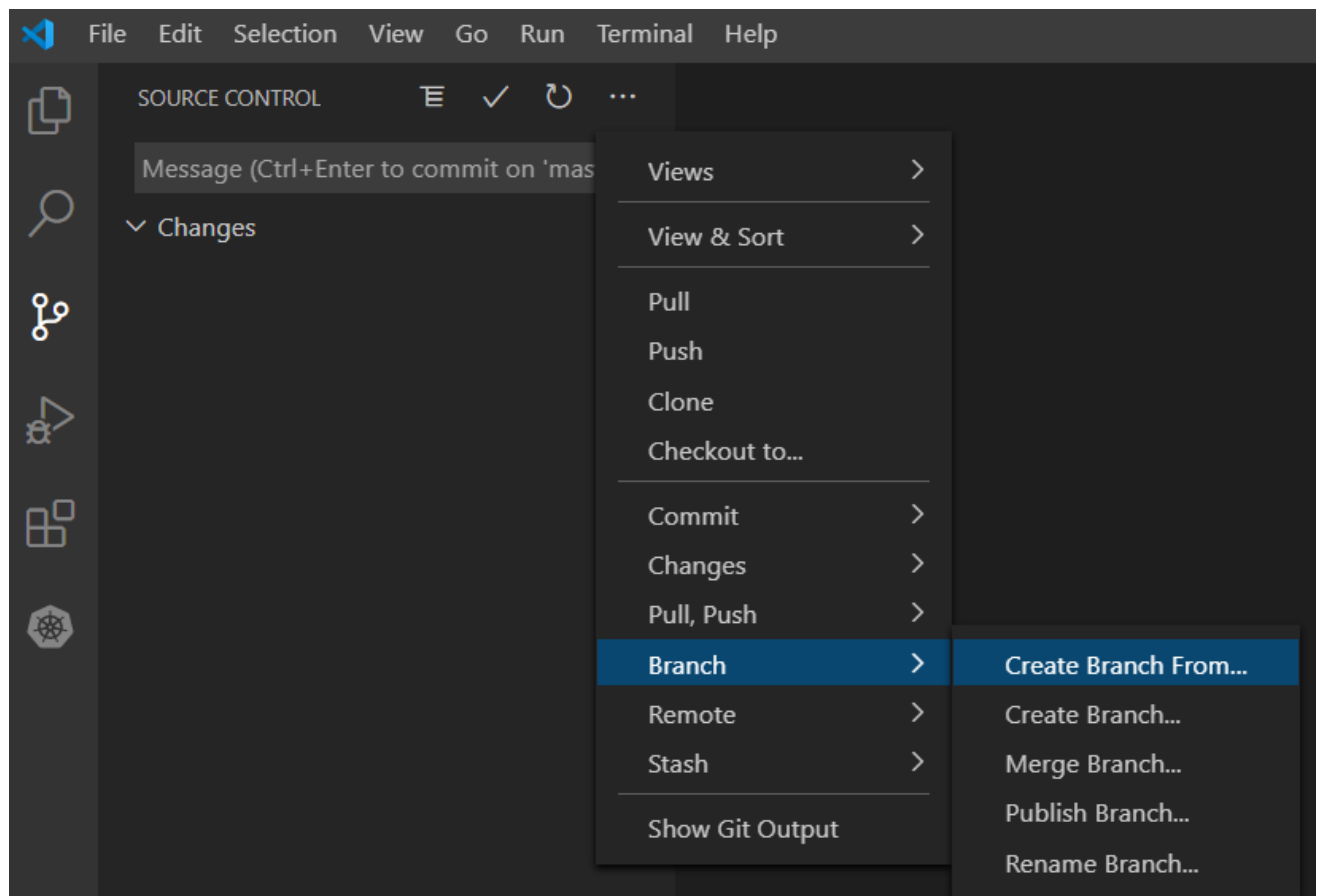
4. Clone the repository with this path from VS code/Visual Studio/Git bash or any other command liner

Creating new working branch

1. Create a new branch from master branch either from terminal or from Source Control window/palette

Ex: Creating a new branch in VS Code:

1. Go to Source Control(Ctrl+Shift+G)
2. Click on Views and More Actions
3. Go to Branch => Create Branch From



2. The naming convention should be followed based on team guidelines for different work item types(user story/bug)

Ex: Branch name for a user story: userstory-<userstorynumber>-<title/shortdescription>

3. The new branch is setup and always make sure to keep your branch upto date before raising a pull request

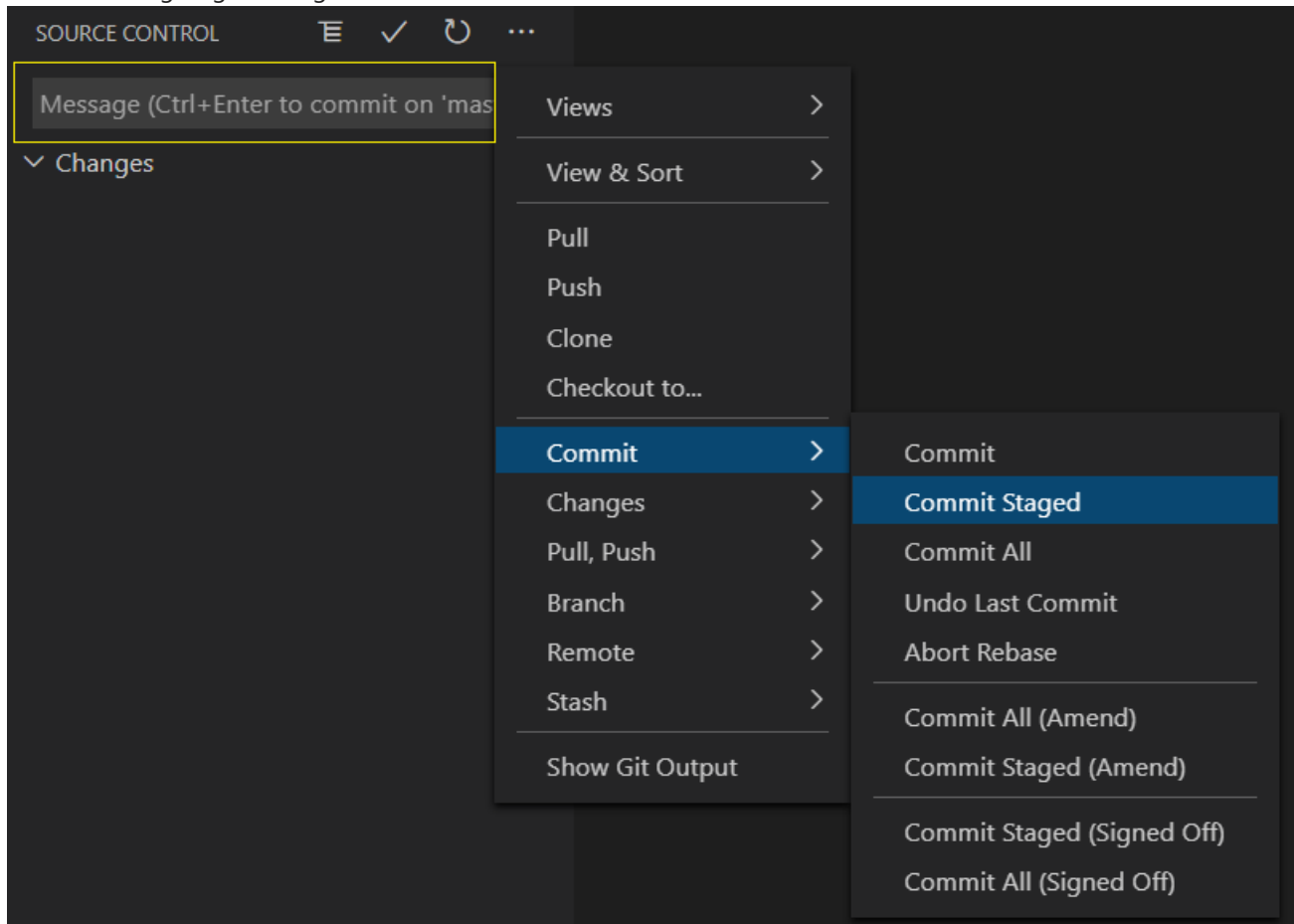
Commit and push the branch

1. Once the changes are done, go to pending changes and stage the changes to check-in

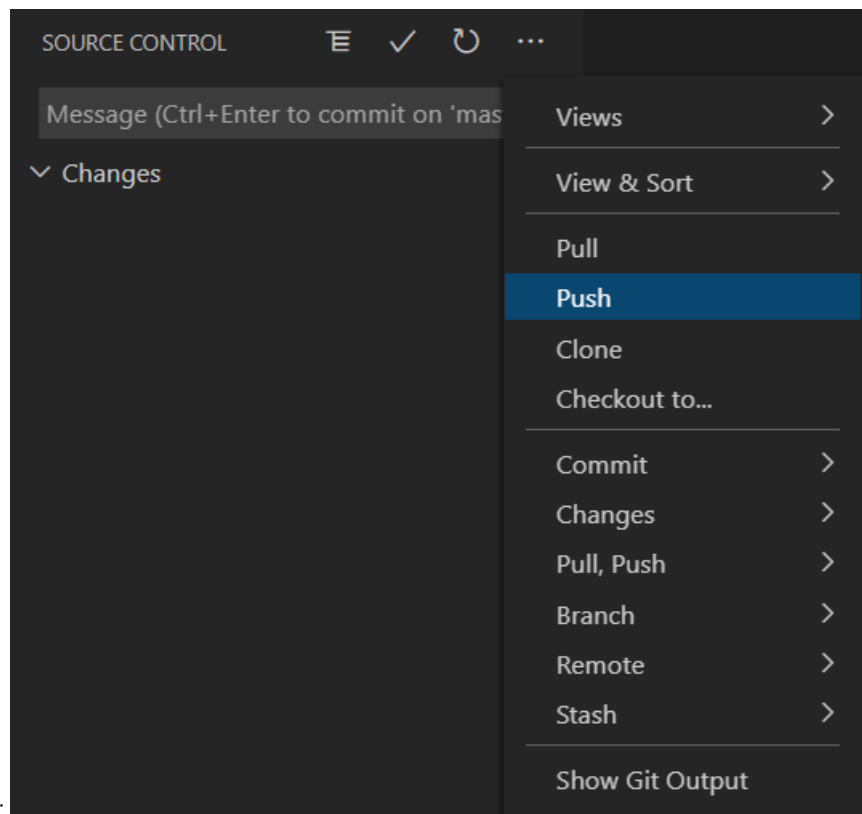
Note: Make sure to get latest/keep the branch in sync with master before committing in order to prevent any conflicts due to changes from the upstream

2. Enter the commit message with information about your changes and commit your changes

Ex: Committing staged changes in VS Code:



3. Once the changes are committed push the branch to origin



Ex: Push branch in VS Code:

Raise a pull request

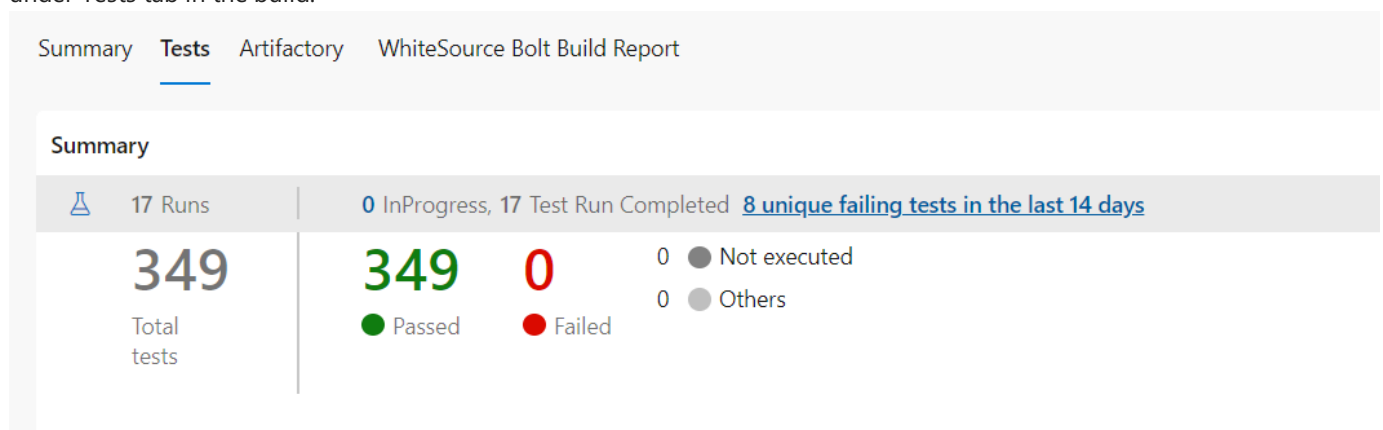
1. Login to VM
2. Open the browser and navigate to [pull requests tab](#)
3. Click "New pull request"
4. Select **base:master** and **compare:<your_branch_name>**
5. If required, review the commits and files changed
6. Give a brief description about the changes under Conversation tab
7. Add the revieweres if required and create the pull request

Note:

For more information on Naming branches and Pull requests, [check this](#)

PR Approval





1. Notify others for review: Either post on Wiki or in a team's Communication channel regarding the Pull request
2. Address the changes requested or any merge conflicts
3. If a build runs with your PR, make sure that the checks are 100% passed. This can be viewed under "Checks" section.
4. Make sure that the tests are 100% passed(if there are any). Troubleshoot the errors if there are any. You can verify these under Tests tab in the build.



5. If the build fails, go through the log and address the issue/inform the team
6. If all the checks are passed, the PR is ready for approval.
7. Update the work item
 - Set the status as Review
 - Add discussion notes and attach any files if required for reference
 - Link the Pull request. For this,
 1. Go to the respective work item
 2. Under Development section, select Add link
 3. Add link pop-up opens up
 4. Select the link type as "GitHub Pull request"
 5. Enter the Pull request url under "Github pull request url" field
 6. Add a comment if required and click "Ok"


Add link

You are adding a link from:

   21923 Document Working with 3M Cloud Repo
Updated 2 hours ago,  Active

Link type

GitHub Pull Request

 [Learn more](#) about GitHub integration with Azure Boards.

GitHub pull request url *

Enter GitHub pull request url

Comment

OK

Cancel

- Save the work item
8. Once the deployment to Development environment is completed:
 - Test your changes
 - Add appropriate notes
 - Mark the work item as resolved

Useful References

1. [Connecting to Github with SSH](#)

2. [Working with repositories in VS Code](#)
3. [Working with repositories in Git Extensions](#)



Naming Branches and Pull requests

Branch and Pull Request naming conventions are used to keep track of different builds and now, as part of our move to open source, for change log documentation. This guide documents how to format the names (or “Titles”) so we all standardize on the same convention.

Title Format

The title of a Branch and PR format should conform to the conventions defined by [Conventional Commits v.1.0.0](#). The convention describes key flags that are picked up by the build process and recommends how to frame the body of the title, a human friendly description of the changes.

In summary, conventional syntax looks like this:

```
<type>[optional scope] <description>
```

In addition to following the conventions, all branches should be prefixed with the Work Item ID. Do not include the story type (Bug, User Story, etc.) or any separators between the id and the body, such as hyphens or colons. The next section summarizes these differences.

Stage: On Create Branch

When creating a branch, prefix the title with the work item ID and then follow the conventional format.
For example:

```
15983 fix(webui) correct minor typos in code
```

Stage: On Merge

When merging a branch, remove the prefixed work item id. Then Squash and Merge to commit the changes into master.
For example:

```
fix(webui) correct minor typos in code  
`
```