# Project Objectives

Implement a Feature-level component , inside of an NG App that exhibits lazy-loading, and has automated testing.

## UI objectives

1. Implement with the most modern version of NG (Angular 10.0.07)
2. Implement an App level solution with a TopMenu and LeftNav using Angular CLI
3. Feature component (*called* ‘Member’) leverages the NG [*element*](https://angular.io/guide/elements) to create a true Web Component. *We should be able to use / re-use with any HTML based CMS or framework*
4. Use[*webpack*](https://webpack.js.org/concepts/)for compaction (downline-load speed) < 250 kb is great!
5. Implement Routine / Lazy Loading for the Add Member component and dynamic import, an optimization and App architecture pattern
6. Implement Tests: Test coverage of the NG App and feature component with Jasmine
7. Implement only the CreateMember() API through the NG Service and API stack
8. Host the App on Azure with a T.B.D. hosting option

*Optional*

* Evaluate if Ivy is meaningful (esp. if small apps), [article here](https://indepth.dev/angular-with-ivy-build-performance-review/)

# SVC Objectives

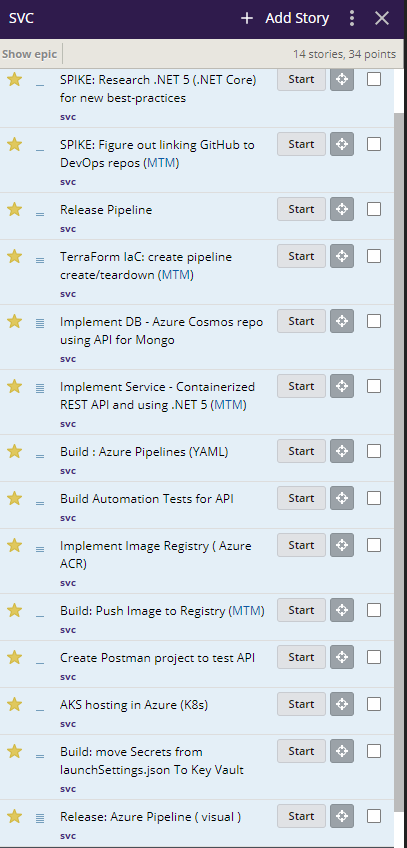
* Use .NET 5 ( successor to .NET Core and the merger with .NET Frk’s) for Service
* Containerize and have a Linux code base, tested locally in Docker For Windows
* Learn HashiCorp TerraForm for create IaC solution in Azure:
  + Azure Container Registry (ACR) image registry
  + AKS (Kubernetes Service)
  + SPIKE: Is there a Load Balancer in front of AKS? How to route to service hosted in K8s?
  + Create NS Record (DNS) name to refer to the service (LoadBalancer)
  + Hosted-Mongo (NoSQL) in Azure Cosmos, see [API for MongoDB](https://docs.microsoft.com/en-us/azure/cosmos-db/mongodb-introduction)
  + Both Provision and Tear-down with TerraForm (no charges after sandbox session)
    - Auto-tear down in Azure? Is that what [DevTestLabs is for – it can do this](https://azure.microsoft.com/en-us/services/devtest-lab/#overview)!

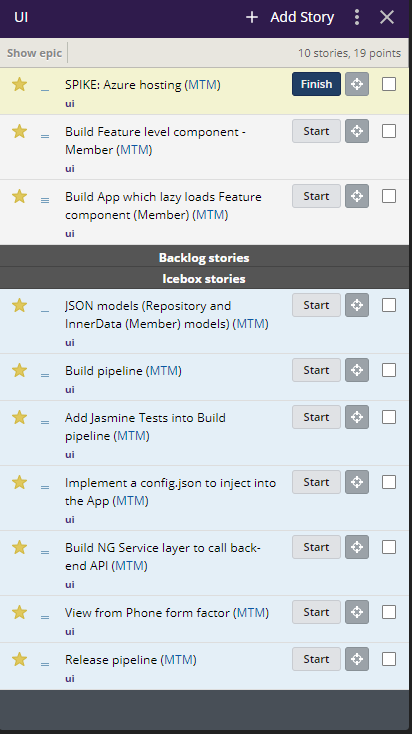
# Narrative

## Stories – from Objectives & Requirements

First, I wrote up the user stories in PivatolTracker for [project NG10E2E](https://www.pivotaltracker.com/n/projects/2463171). The UAT’s will be written or implied as I code. Tasks will also be added. GitHub check-ins will be tied to a PivotalTracker story.

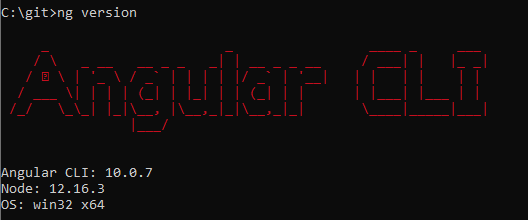
Initial Backlogs on 8/26/2020 are,

For the Microservice (SVC) epic:  


For the UI epic:  


## Environment Latest Tools

I’ll use Visual Studio 2019 Preview edition, which also installs the .NET 5 Preview. I’ll also install the latest Angular CLI (10.0.7).

1. I updated NG to 10.0.7  
   
2. I updated VS 2019 to v 16.8 Preview 2  
   
   1. Checked to verify the latest .NET 5 version is installed  
      

## Getting Started

### First story is a spike: How to host on Azure the NG App

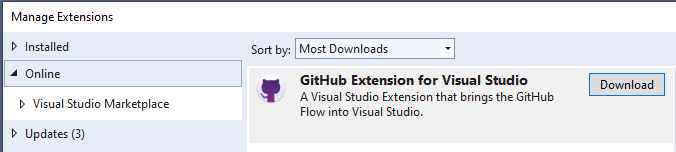
So here I guess I want to build the App with the CLI, then also learn the WebPack things I need to understand in one fell swoop.

Once I understand Webpack, I imagine the method of hosting will be easy to implement upon research of how to host on Azure.

Thesis: 1) WebApp is going to be easy or ok to use Webpack with  
2) There will be other ways, some obvious and some not. Blob URL has been mentioned, but it’s not a server (but can it act as one, given a URI to the Webpack? 3) smallest served Linux front-end server with Nginx on it, should also be considered.

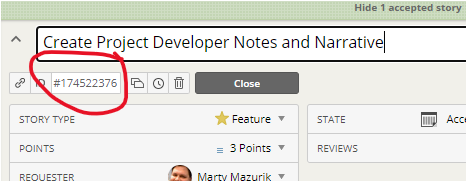
#### Unscheduled work: put dev document into Github from a Pull Request (PR) into PivotalTracker

I found I didn’t have the ability to Connect in VS2019 to Github, looked for extension: bingo !



#### Branching strategy

I will create a branch, off of trunk (Main) using the Story ID (we will be at the story level on check-ins and merge requests; at issue is we should then ‘stash’ in between code (daily) for safekeeping. And with Stash is there any other work to do. (If I’m not religious about stash, will it kill me? Only if my local HD dies)



I create then a branch for the work **\174522376** gets created linked to the story.

And issue pull request for a merge. I’ll do this out of habit, although it is just me right now.

#### Conclusion of spike: Use the WebApp for now – wrong. Nope!

Rationale: there are enough capabilities to host the website NG \dist folder as a WebApp to warrant strong consideration

1. Customers will be most familiar with WebApp as its name implies the Azure purpose
2. Deployment slots : permit a blue/green deploy with staging/production or just call them green (idle) and blue (active). It doesn’t seem to matter in blue/green you just use the other for deployment based on which one is “live” (seems dangerous from a communication standpoint).  
   See [CloudFoundry Doc on blue/green here](https://docs.cloudfoundry.org/devguide/deploy-apps/blue-green.html#:~:text=Blue%2Dgreen%20deployment%20is%20a,live%20and%20Green%20is%20idle.).

For that reason: we’ll stick with “staging” and “production”.

1. DDOS Protection can be applied (t.b.d) easily
2. Does scale (App Service Plan : [Scale Up more here](https://docs.microsoft.com/en-us/azure/app-service/manage-scale-up#:~:text=In%20your%20browser%2C%20open%20the,options%20to%20show%20more%20tiers.) also can scale out (up to 30 instances) depending on your plan)

STOP! WebApp is terrible IaC from Terraform, and not really how I wanted to deply. Go with NG in a container (with nginx) solution. Solution: using ng in a container. The UI is containerized.

Following this post:  [7 Steps to Containerize …](https://medium.com/better-programming/7-steps-to-dockerize-your-angular-9-app-with-nginx-915f0f5acac)

### Another Story: IaC Code - Terraform

We want the infrastructure that the Release pipeline needs, to be available prior to Release.

The question is whether the Release Azure Pipeline contains the Iac (Terraform) step to run as part of release? Would be kind of neat, but then there is also “teardown” needed.

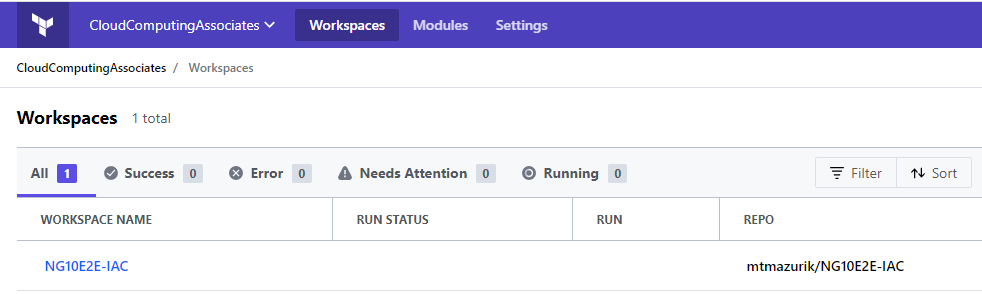
Vision: a single Terraform script that both creates and tears-down the infrastructure.

#### I set up a Terraform Cloud Account

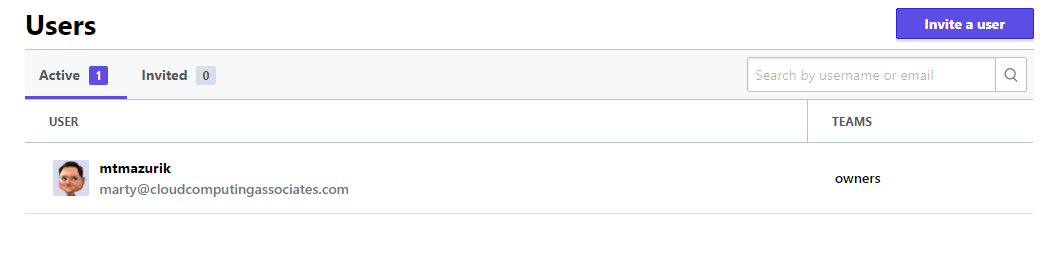
U: mtmazurik  
Email: [marty@cloudcomputingassociates.com](mailto:marty@cloudcomputingassociates.com)  
Pwd: {usual}

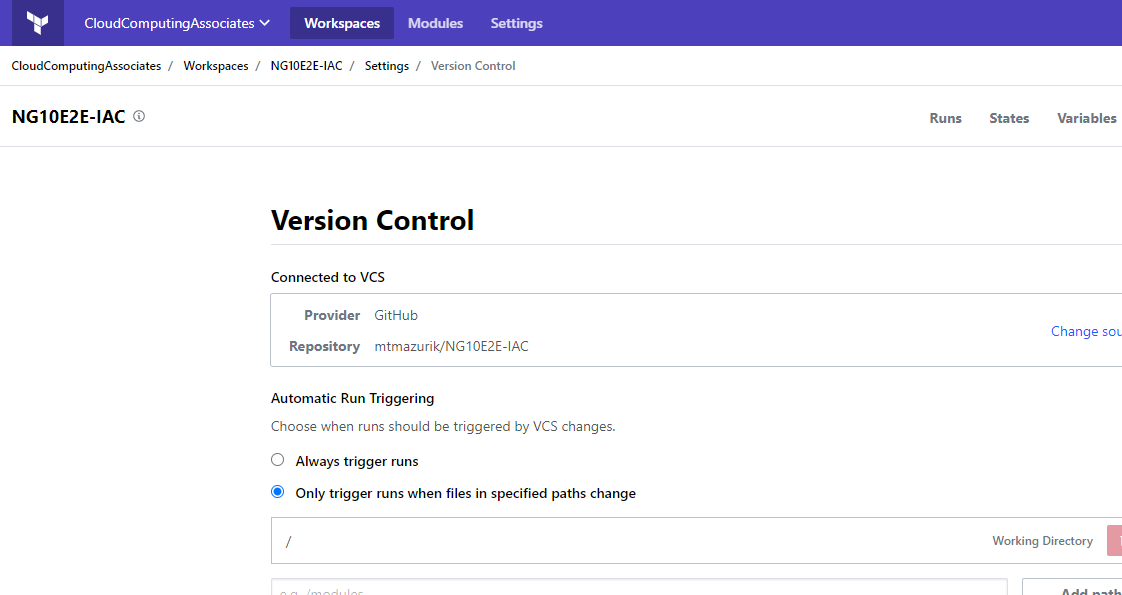
<https://app.terraform.io/app/CloudComputingAssociates/workspaces>

Then I set up a Workspace for the project, from a Github repo I created: /mtmazurik/NG10E2E-IAC of the same name:



I am the only user currently:



Under Workspaces and Version Control you can see the trigger and tie-in to VCS (Version Control System):  


**Thesis:** changes in the IAC directory will trigger the running of the Workspace NG10E2E-IAC code.

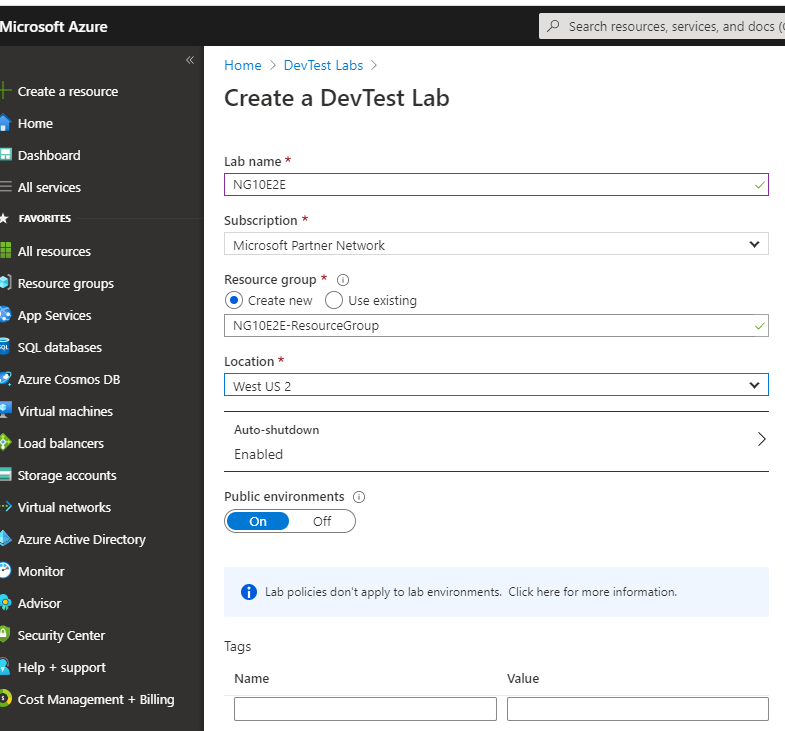
### Strategy: using Terraform for effective IaC creation/deletion

Learning to write Terraform I used this link first, as I want to use Azure DevTestLabs (so they auto-cleanup) <https://www.terraform.io/docs/providers/azurerm/r/dev_test_lab.html>

Choose whether to leverage *Azure Dev Test Labs*: [read here](https://docs.microsoft.com/en-us/azure/devtest-labs/devtest-lab-overview)about it, Tutorial: [click here](https://docs.microsoft.com/en-us/azure/devtest-labs/tutorial-create-custom-lab)

**Thesis:** may not be a good idea; seems very VM focused re: shutdown (release of resources), the only reason to consider, given we want to have shutdown scripts … this is cheating? (or maybe comprehensive, like what Google Cloud Projects buy you, totality in releasing expensive resources.)  
Thesis proved. *Through research, it looks VERY VM oriented for classrooms and playing with instances of VM’s (either Windows 10, Server or Linux flavors) and allowing auto shutdown to free up resources.*

#### Investigate: Create a DevTest Lab – nope!



! STOP – wrong path – we will use the regular Azure Subscription without DevTestLabs as it appears to be fodder for setting up VMs and VPC’s and VPN’s and other rudimentary things for “students” to play with.

Why it won’t work: we are using Kubernetes Service (AKS) with containers, and ACR (azure container registry) and eventually AKV (Azure Key Vault) these don’t seem represented in Dev Test Labs very well from what I can see.

#### Terraform IaC (two levels concept)

1. We’ll use an IaC for persistent services (like Vaults, Registries) that need to be set up once-and-only-once. Implying also, they are not torn down.  
     
   Call it CCA-PERSISTENT, and create an Azure RG (Resource Group) of the same name, then if you need to delete use the RG to tear-down within the Azure Portal. (manual, not scripted at this time).
2. Create a 2nd Terraform project for the system resources that will be created by the “Release Pipeline” during release, this pipeline and RG will be named: ng10e2e  
     
   Candidates for creation here during the release are:
   1. AKS (Azure Kubernetes Service)  
      Yes! The whole of the Kubernetes Service will be created, and torn down (manually via the resource group).
   2. Load Balancer ? We would like to have the Load Balancer in place and test having 2, 3 or more of the ng10e2e-ui hit by a load balancer route that is working and test, with images showing something about the environment they are running in (maybe some inner ID we can poll is easier than different images?)

To this end, we will have two different IaC modules written in Terraform.

We will create the one for release pipeline and run it ourselves, and then as part of the release pipeline.

The 1st Terraform module (persistent services) is written and working, here : <https://github.com/mtmazurik/terraform/tree/master/Azure/CCA-PERSISTENT>

A breakout on Terraform itself, the creation of the above code and the online training/notes document in   
L:\CloudComputingAssociates\Technical\Terraform\Training-Implementing Terraform on Microsoft Azure- course notes.docx.  
Note: |  
I leveraged an Udemy training course, only up to the point of initial project-related understanding was attained.   
  
TO-DO: Complete the Udemy training course for fuller knowledge on Terraform coding.

### Story (#174518606): Build App (basic)

Every App typically is an App (and not a service) because it has a face, called a User Interface, or UI for short.

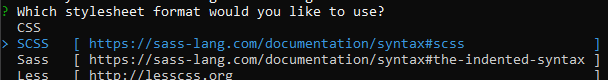
We’ve decided (earlier) on NG10 for this app, installed the appropriate tools on our Windows Developer machine, we stay with Windows predominately because ( a) we document in Word, and b) other windows tools like OneNote for clipping things, and c) everything is faster with the Windows desktop, responsiveness wise. … developer cycles are everything.)

Recapping what we’ll do in this initial story:

* NG 10 CLI App
* Containerize it, with nginx
* Test it locally (Docker for Windows)
* Push to ACR (Azure Container Registry) manually

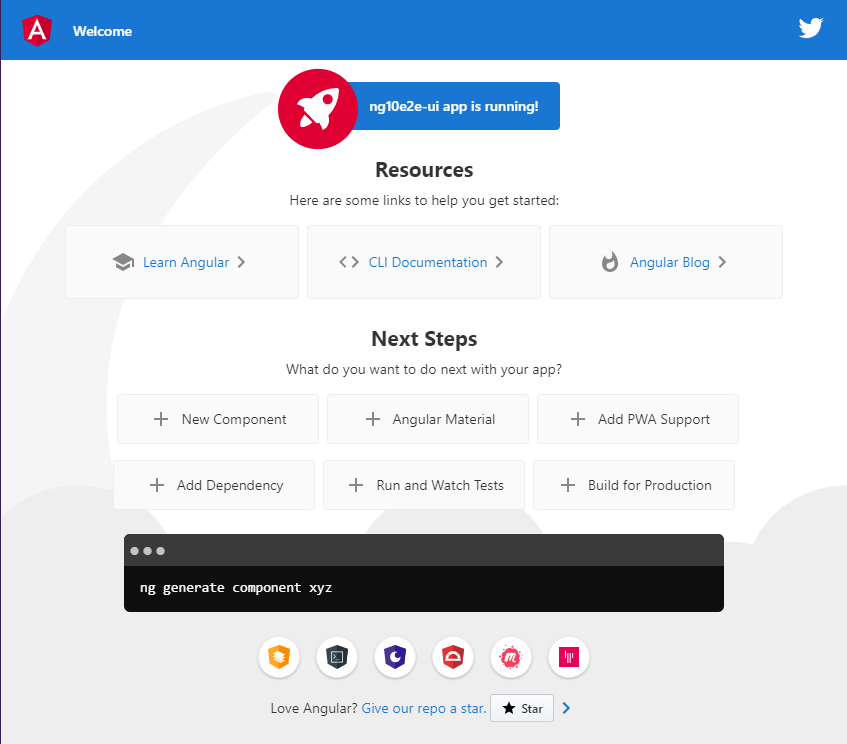
#### Angular CLI – Create the App

Angular official documentation [Getting Started](https://cli.angular.io/)

1. > ng new ng10e2e-ui
   1.   
      I answered ‘y’ (yes)
   2. Stylesheet: SCSS  
      
2. Fixup the .gitignore (if one exists) by the new .gitignore that Angular CLI produces, to have all the angular/node specific information in your .gitignore file.
3. Test it out: > ng serve  
   (then, open browser windows to: <http://localhost:4200/> )

Observation: while having a spanking new ‘default’ App page for the SPA, with twitter bird and such, I don’t see info on current version. That would be nice to know.

? How to tell what version of ng is running for a website?



#### Containerize using a Dockerfile & push to ACR

Here we containerize an ng app to use nginx and Linux container as the serving image of our App UI.

1. Create a Multi-Stage Docker Image. I did this in an example RepositoryNook-UI github project back in January 2020, see here : <https://github.com/mtmazurik/RepositoryNook-UI/blob/master/dockerfile>

Credit to Michael Herman, [“Dockerizing an Angular App”,](https://mherman.org/blog/dockerizing-an-angular-app/) May 20, 2019

*NOTE ON ‘No initial AUTOMATED TESTING’ taking place:d****I’m not currently automate testin****g : use* [*Jasmine*](https://jasmine.github.io/) *(Behavior-driven Development BDD),* ***Jasmine*** *is installed with****$ ng test***

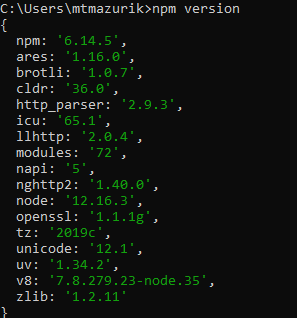
*Also, see* [*ng Testing*](https://angular.io/guide/testing) *and Protractor. Learn* [*E2E testing*](https://stackshare.io/stackups/jasmine-vs-protractor) *with* ***Protractor****.*

***Karma,*** *what is it****?***

*Karma just launches an HTTP server, and generates the test runner HTML file you probably already know from your favourite testing framework. Karma is not a testing framework, nor an assertion library.*

***Added Story: Created Story in PivitolTracker for TESTING knowledge and implementation***

1. *Learn how, to use Jasmine (BDD) and Protractor (E2E)*
2. ***Extend the Automated Build in Azure DevOps to include Jasmine and/or Protractor test(s).***
3. Update the Dockerfile in root of project, follow the github example for mtmazurik\RepositoryNook-UI
   1. Update node line



Example: node: 12.16.3

* 1. Update line run npm install -g @angular/cli@8.3.2 to version you are using  
     example: @angular/cli@10.0.7

1. Create a \nginx subdirectory  
   create a default.conf file (follow format in repository example: mtmazurik\RepositoryNook-UI)
2. Test creation of Docker image locally, with tag name to push to :   
   registrycca.azurecr.io

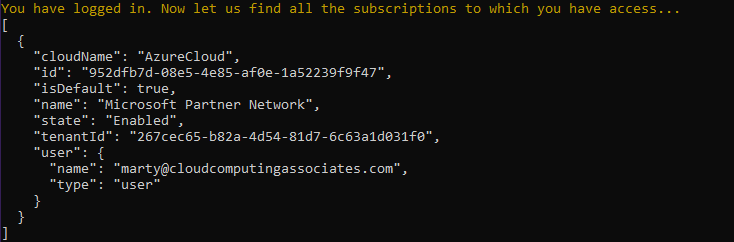
docker build -t registrycca.azurecr.io/ng10e2e-ui:latest .

1. Docker push

docker push registrycca.azurecr.io/ng10e2e-ui:latest

error – AZ authentication required

az login  
  
u: [marty@cloucomputingassociates.com](mailto:marty@cloucomputingassociates.com)

p: {usual}234!  
  


error – nope;

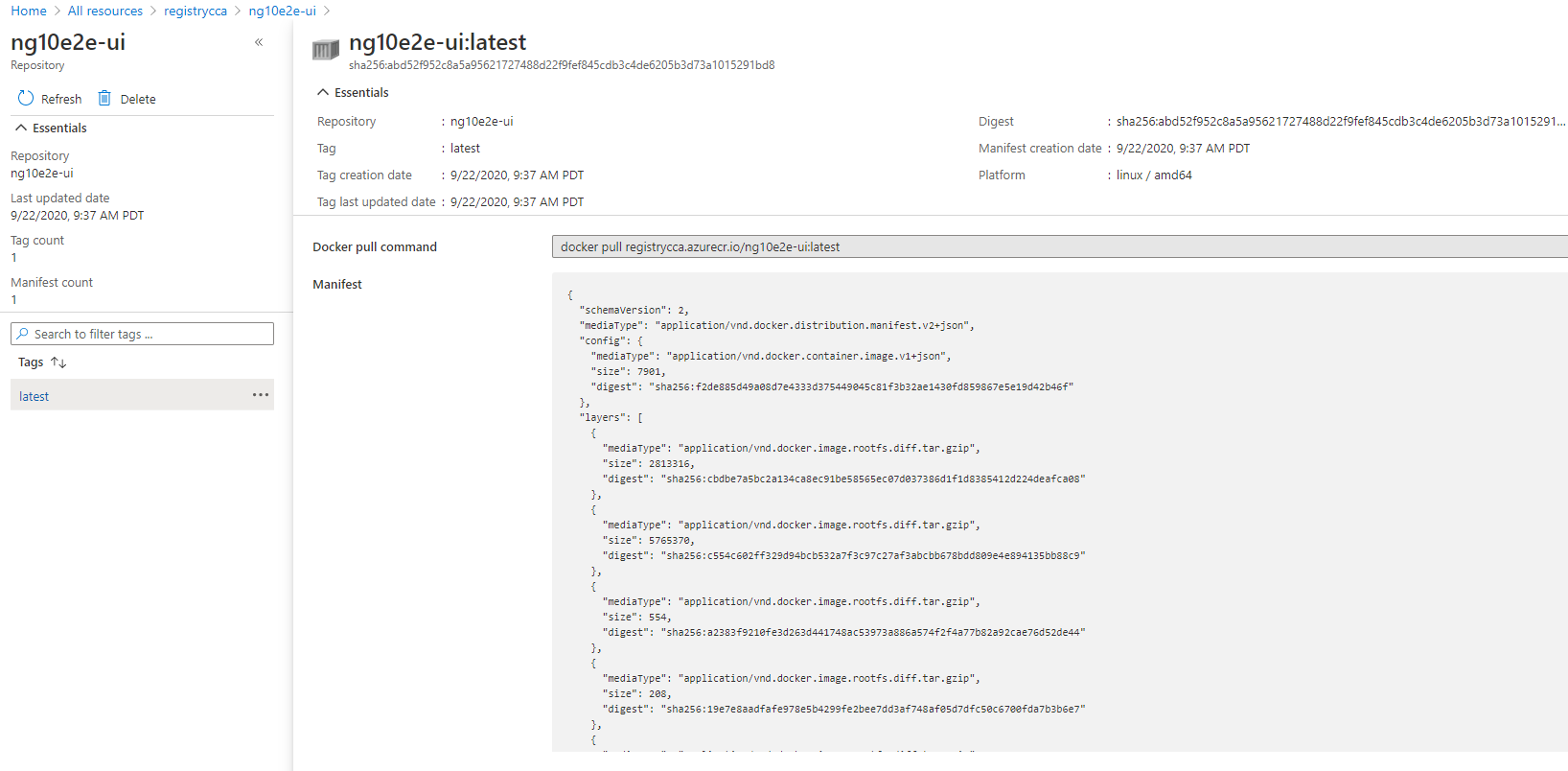
checked that my login IS part of IAM security group: cca-developer and that role is *contributor* access on the registrycca service. Hmm.

* 1. Log in to ACR (registrycca itself) with:

az acr login –name registrycca



Worked!

* 1. Looking around we see the details of the image in the ACR  
       
     

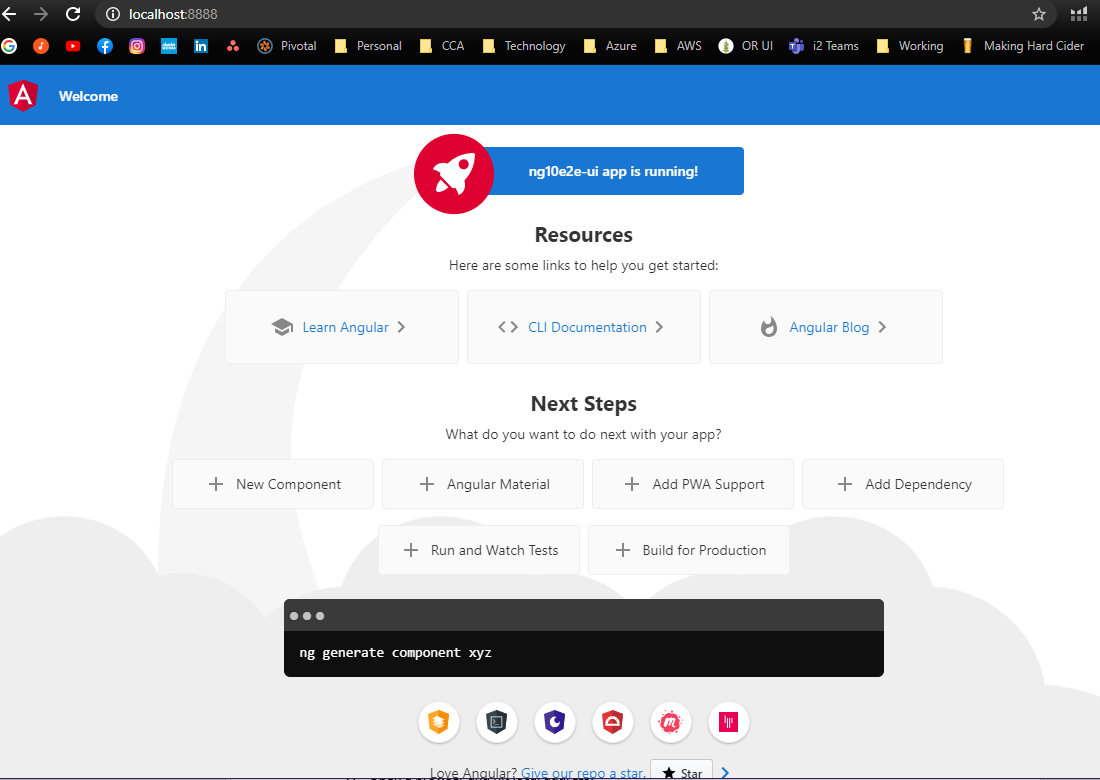
1. Run locally & sanity check website is accessible from container on port 80
   1. We’ll map port 80 to port :8888 just in case port:80 is in use:

docker run -d -p:8888:80 --name registrycca.azurecr.io/ng10e2e-ui:latest

1. Open a browser and hit local address:

<http://localhost:8888/>

Success!



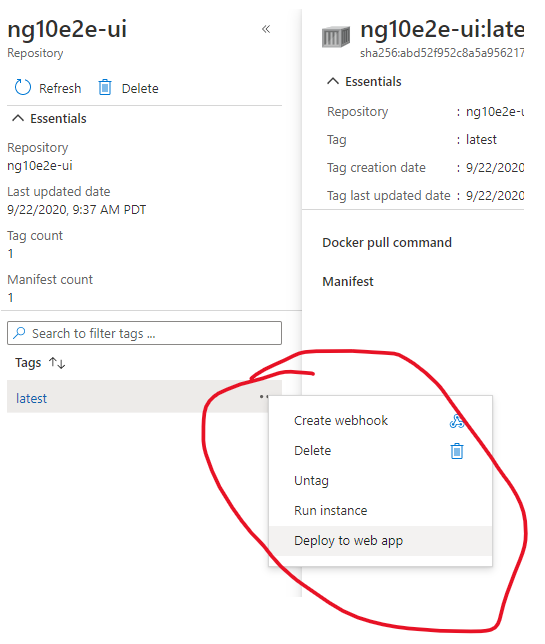
1. Azure Container Registry findings

Poking around this submenu implies:

1. Webhooks can ‘do something’ with image  
   no real practical examples on MS Site, [Using Azure Container Registry webhooks](https://docs.microsoft.com/en-us/azure/container-registry/container-registry-webhook)

And thinking about it “do I want this inner webhook to launch helm releases – automatically after a build is successful? No release ‘gate’ or human intervention.  
And if so, is this the right way to wire-it-up (or is a triggered ‘release pipeline’ a better bet?)  
These feel akin to writing table triggers and stored procs (inside of SQL server db) where the code is not as accessible for modification/debugging versus in this case using:  
**a release pipeline**

1. Run Instance  
   The service is called: *Azure Container Instance* service, [more here](https://docs.microsoft.com/en-us/azure/container-instances/container-instances-using-azure-container-registry)  
   Seems pretty cool, if all you want to do is throw it up into an Azure Container Instance  
   (not k8s). Won’t test it as it is a “Microsoft only play”, not cross cloud possible.
2. Deploy to web app  
   Similar to b) , I won’t be testing this out.



Conclusion of ng10e2e hosted in a container image with nginx server app creation and insertion into ACR registry.

Next focus:   
Release Engineering:  
  
Terraform module (#2) for creating AKS instance inside of specific (NG10E2E resource group) and teardown.

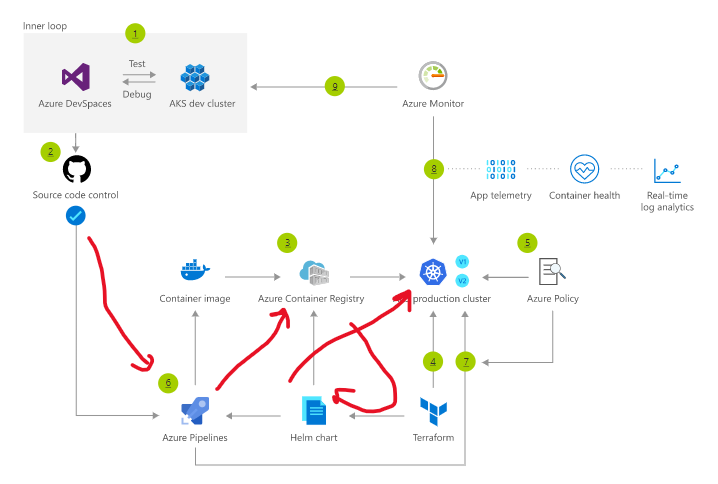
Refactor to get image deployed out to k8s (AKS instance) using helm.

### Terraform Module #2 ( release engineering )

This terraform module should establish the k8s cluster, any other services (like load balancer) and Virtual Network (depending on best practices for exposing an AKS based front-end )

#### Best practices Cloud configuration for UI hosted on Azure Kubernetes Service (AKS)

[More here](https://azure.microsoft.com/en-us/services/kubernetes-service/)

This diagram is doing a lot,   


I might do better than my red arrows and rearrange, but near as I can tell:

1. -------------------------
2. Check in your code
3. The trigger of an Azure Build pipeline (creates target: docker image )
4. Docker image into ACR
5. Which can have a - Webhook to terraform process kicking off (if fully automated CI/CD)

Or

1. Separate Release Pipeline is created and run manually (no hooks): human intervention step.  
     
   **Azure Policy** is news to me (Governance) [more on Azure Policy, click here](https://docs.microsoft.com/en-us/azure/governance/policy/overview)  
   setting up here scale (# of instances) -type values, for aks, is why I think it shows up here.