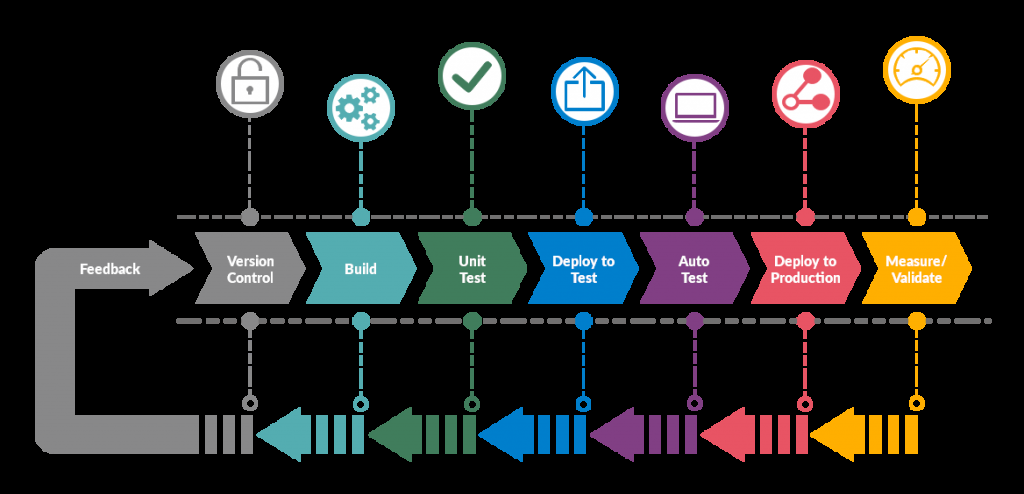
**Acceptance Criteria**

**1.Deployable code base**

ADO is the Azure DevOps Platform. An ADO Pipeline is the coded yaml pipeline feature of ADO. Ansible is the core ansible package (not ansible tower). An AnsibleRole is mostly one of the roles found in the AnsibleRoles project, but could also be a local role in a consumer solution. There may also be local roles in the pipeline framework. yaml files are used substantially to define the entire framework and its capabilities. Both ADO and Ansible use yaml as a syntax, so we will be specific when we are talking about one or the other. Where possible, we will separate ADO and Ansible yaml by folder, so there is no naming confusion. Consumer is someone using the framework. Specifically, it will be a build in another project, referencing an application or service that is to be built/tested/deployed, etc. The Pipeline Framework is this repository, and the associated workflows and tool chain products. There is one pipeline framework for all of the CRA work. The idea is that this framework should meet the needs of all of the current migration, digital, and greenfield applications and services. A Pipeline is a top-level ADO yaml file that is exposed to Consumers. A Pipeline Component is a unit of functionality from which top-level pipelines can be assembled.  
  


**2.Solution deployed to a cloud environment**

We define our pipeline in a YAML file called 'azure-pipelines.yml' with the rest of our app.The pipeline is versioned with our code. It follows the same branching structure. we get validation of our changes through code reviews in pull requests and branch build policies.

Every branch you use can modify the build policy by modifying the azure-pipelines.yml file.  
A change to the build process might cause a break or result in an unexpected outcome. Because the change is in version control with the rest of your codebase, we can more easily identify the issue.

Follow these basic steps:

* After using the project setup in Service Now you will have an azure-pipelines.yml in your repo
* Edit your azure-pipelines.yml file to define your build.
* Push our code to your version control repository. This action kicks off the default trigger to build and deploy and then monitor the results.
* our code is now updated, built, tested, and packaged. It can be deployed to any target.

**azure-pipelines.yml**

resources:

repositories:

- repository: pf

type: git

name: CVX-DevOps/ADOPipeline

ref: refs/tags/4.0.0

########################################################

# WARNING: Removing/Modifying the following Parameters

# may cause errors with your pipeline!

########################################################

parameters:

- name: forceUpdate

displayName: Update Pipeline

type: boolean

default: false

- name: deployEnvironment

displayName: Environment

type: string

default: dev

########################################################

trigger: none

stages:

- template: pipeline.yml@pf

parameters:

${{ insert }}: ${{ parameters }} # DO NOT MODIFY

serviceId: 99999

projects:

- name: MyProject

type: dotnet.core

filePath: /path/to/project/MyProject.sln

buildConfiguration: 'Release'

buildPlatform: 'Any CPU'

playbook: /path/to/main.yml

**Parameters**

The following parameters are project-specific and used in the azure-pipelines.yml file:

###################################################

# Required parameters.

###################################################

name: '' # Must be unique within the list of projects.

type: dotnet.core

filePath: '' # Project Path & File (e.g., /path/to/file/solution.sln, /path/to/file/project.csproj)

###################################################

# Optional parameters.

###################################################

dependsOn: [] # Used to specify a dependency on another project.

coreVersion: 3.1.100

outputProject: '' # Specifies the source project to use for the deploy binary (in cases where a solution has many projects).

buildConfiguration: 'Release' # Used with .NET builds.

buildPlatform: 'AnyCPU' # Used with .NET builds. NOTE: For solution-based builds, add a space (e.g., 'Any CPU').

buildArguments: '' # Used with .NET build step as additional arguments needed.

buildOutputProjectPath: '' # Used to specify the expected (project) output folder of the build for solutions. If building multiple projects in a solution, this will force the one to use as the final binary.

customArtifactPattern: '' # Pattern to use for files published to Artifacts (ex. '\*\*/\*.tgz').

3.**How will you deploy this solution (in code or as a todo list if time is limited). i.e. how and where will this run? Details of CI/CD pieplines etc.**

To build the code and deploy we are using Pipelines Framework and Ansible Roles.Ansible playbook is just a set of ansible tasks to execute.

we have 3 basic ansible playbooks: main.yml, provisioning.yml and deployment.yml.

This playbook just executes one of two playbooks based on the value which we passed to the build\_tags pipeline variable.

infra: executes provisioning.yml playbook. It performs basic infrastructure creation (storage account, Cosmos DB, Azure Databricks, etc.)

code: executes deployment.yml playbook. It performs deployment of the services.

For example, ansible-role-azure-databricks in provisioning just creates an Azure Databricks service, whereas ansible-role-azure-databricks-deploy in deployment deploys

notebooks from specified path (notebook/refined and notebook/produced) to the already created Databricks service.

4. **How will you manage any infrastructure needed?**

We have used Aansible Playbook and in the provisioning.yml file we add the infra as below:  
  
infra: executes provisioning.yml playbook. It performs basic infrastructure creation (storage account, Cosmos DB, Azure Databricks, etc.)

code: executes deployment.yml playbook. It performs deployment of the services.

For example, ansible-role-azure-databricks in provisioning just creates an Azure Databricks service, whereas ansible-role-azure-databricks-deploy in deployment deploys

notebooks from specified path (notebook/refined and notebook/produced) to the already created Databricks service.

**5. Delivered as a feature branch in the repo fork**

Yes, Due to permission denied I am unable to do so.

6. **Any DevOps/Cicd components that would support this feature in a production setting**

Artifacts and environment variable