Release Management Exercises

1. Browse to the VSTS portal at <https://microsoftleap.visualstudio.com> and click on the project for your team
2. You should already have a personal remote branch in VSTS from the previous VSTS exercises.
   1. If you don’t, clone the master branch from the Test repo to a new remote branch called “<yourAlias>-dev”

# Build

## Creating build definitions

1. Click on Build and Release
2. Click on Builds
3. Click on New definition
4. Select .NET Desktop and click on Apply
5. Change the build definition name to your alias
6. In the Agent queue dropdown, select Hosted
7. In the left pane, click on the Get Sources step
8. In the Branch dropdown, select your personal remote branch
9. Click on Save and Queue -> Save
10. Click on Save
11. Congrats, you now have a build definition!

## Triggering builds manually

1. Click on Build and Release in the top bar
2. Click on the link to your build definition
3. Click on Queue new build in the top right
4. Click on Queue
5. Click on the build number in the green bar near the top left to watch the progress

## Viewing build progress, history, logs, and artifacts

1. Click on Builds the top pane on the left
2. Click on your build definition. This page shows the history of builds for this definition
3. Click on the latest completed build
4. Click on Artifacts to view build artifacts
5. Click on Tests to view unit test results (there are no tests for this dummy code)
6. Click on any step in the left pane to view logs for that step

## Triggering builds on commit

1. Browse to your build definition
2. Click on edit in the top right
3. Click on the Triggers pane
4. Click on Enable continuous integration
5. Make sure the branch filters say “Include” and your branch name
6. Click on Save and Queue -> Save in the top left
7. Make a change in your local branch and commit it
8. Push the change to your tracked branch
9. Go to the builds page and watch your build start automatically

# Release

## Creating release definitions

1. Click on Build and Release in the top pane and click on Releases
2. Click on New definition
3. Click on Empty Process in the top right
4. In the top left, rename New Release Definition to the same name as your build definition from the previous section
5. Under Artifacts, click on Add artifact
6. Select the build definition you created in the previous section and click on Add
   1. This allows artifacts (outputs) from your build to be used in your release
7. In the top pane, click on Tasks
8. Change the environment name to Dev
9. In the Agent Phase section in the left pane, click on the “+” symbol to add a task
10. In the search box, type Command. Click on Command Line and click on Add three times
    1. This will add 3 steps to the release
11. In the left pane, click on the first step. Use the following values:
    1. Display name: Mount Azure File Share
    2. Tool: net
    3. Arguments: use $(Drive) \\leap.file.core.windows.net\test /u:$(AzureUser) $(AzurePassword)
12. In the left pane, click on the second step. Use the following values:
    1. Display name: Copy artifacts
    2. Tool: xcopy
    3. Arguments: "$(Source)" "$(Dest)" /Y /S /I
13. In the left pane, click on the third step. Use the following values:
    1. Display name: Unmount Azure File Share
    2. Tool: net
    3. Arguments: use $(Drive) /delete
14. Notice the steps above use variables that look like $(this). These are environment variables that we will set now.
15. Click on the Variables tab. Add the following key value pairs:

|  |  |
| --- | --- |
| Name | Value |
| AzurePassword | Y9m0DQpWqoAwxUWRFhsNxwieR4D9p/gUEX6sWN/E77rRw/INBkgcTlIwZDXCbVbR/WNyc1Y0ytUZXgVGfF02+g== |
| AzureUser | AZURE\leap |
| Dest | $(Drive)\$(Release.DefinitionName)\$(Release.EnvironmentName)\$(Release.ReleaseName) |
| Drive | Z: |
| Source | $(System.DefaultWorkingDirectory)\$(Build.DefinitionName)\drop\HelloWorld\HelloWorld\bin\Release |

1. Click on the Pipeline tab. You should see the artifacts your release will use as well as a single environment called Dev.
2. Click on the oval on the left side of the Dev environment (Pre-deployment conditions). Under triggers, select Manual only.
3. Mouse over the Dev environment and click on Clone
   1. Note that the tasks for Dev and Prod do not have to be identical, but they are for this exercise.
4. Click on Copy of Dev and change the name to Prod
5. Click on the oval on the left side of the Prod environment (Pre-deployment conditions). Under triggers, select Manual only.
6. Click on Save at the top right. Click on OK.
7. Congratulations, you now have a release definition linked to your build definition.
   1. The release has two environments called Dev and Prod, and both use manual deployments.
   2. The only thing this release does is copy some files to an Azure file share

## Triggering releases manually

1. Click on +Release in the top right, and click on Create release
2. Click on Create
3. You should get a success message in a green bar near the top of the page, something like “Release Release-1 has been created”
4. Click on the Release-1 link
5. Notice the release has been created, but no deployment has occurred. You can see both environments, but the status should be NOT DEPLOYED for both of them.
6. To the right of the Dev environment, click on the “…” under Actions and click on Deploy. Click on Deploy.
7. Click on Logs at the top to watch the Dev environment tasks while they run.
8. If the deployment completes successfully, artifacts from your build will be copied to an azure file share
9. To see the files, open a command prompt and run the following command
   1. net use Z: \\leap.file.core.windows.net\test /u:AZURE\leap Y9m0DQpWqoAwxUWRFhsNxwieR4D9p/gUEX6sWN/E77rRw/INBkgcTlIwZDXCbVbR/WNyc1Y0ytUZXgVGfF02+g==
10. Browse to your released files
    1. Open windows explorer and browse to the Z drive
    2. Double click on the name of your release definition.
    3. Double click on the name of your environment (Dev, in this case).
    4. Double click on the release number (Probably Release-1, in this case).
    5. Your build artifacts should be in this folder

## Viewing release progress and history

1. Click on the Build and Release tab and click on the Releases tab
2. Click on your release definition in the left pane
3. The right pane will show the history of all the releases for your definition, as well as the status of the deployment to each environment
4. Double click on the release title to drill down into that release
5. The summary tab shows the release from a high level
6. The logs tab shows each environment, each step for that environment, and the logs for each step

## Triggering releases on build success

1. Creating a release manually is great, but it doesn’t take advantage of continuous deployment.
2. Open your release definition and click on the Pipeline tab
3. Under artifacts, click on the lightning bolt icon
4. In the right pane, toggle the Continuous deployment trigger to Enabled
5. Click on Save at the top right and click on OK.
6. Manually kick off a new build from your build definition
7. Once the build completes successfully, notice that a new release will be created for this build.
8. Congratulations, you now have releases that get created automatically!

## Triggering environment deployment on release creation

1. Creating a release automatically is great, but it doesn’t take advantage of continuous delivery.
2. Open your release definition and click on the pipeline tab.
3. Click on the oval on the left side of the Dev environment (Pre-deployment conditions). Under triggers, select After release.
4. Click on Save at the top right and click on OK.
5. Manually kick off a new build from your build definition.
6. Once the build completes successfully, notice that a new release will be created for this build.
7. After the release is created, notice that the Dev environment will be deployed automatically.
8. Congratulations, your release now implements continuous delivery. Your changes now flow from commit to build to release to deployment automatically! Well done!
   1. **Every lab we do in this class should be set up in this way before any additional lab work is done!**