elft.net Continuous Application Delivery (~2 weeks)

peter mostert (elft.net)

2023-08-23

Abstract

Founding a new base to build upon.

Contents

About me:

- name: peter mostert
- education: graduated as an Ingenieur in Electronics
- company: elft.net - Digital Experience

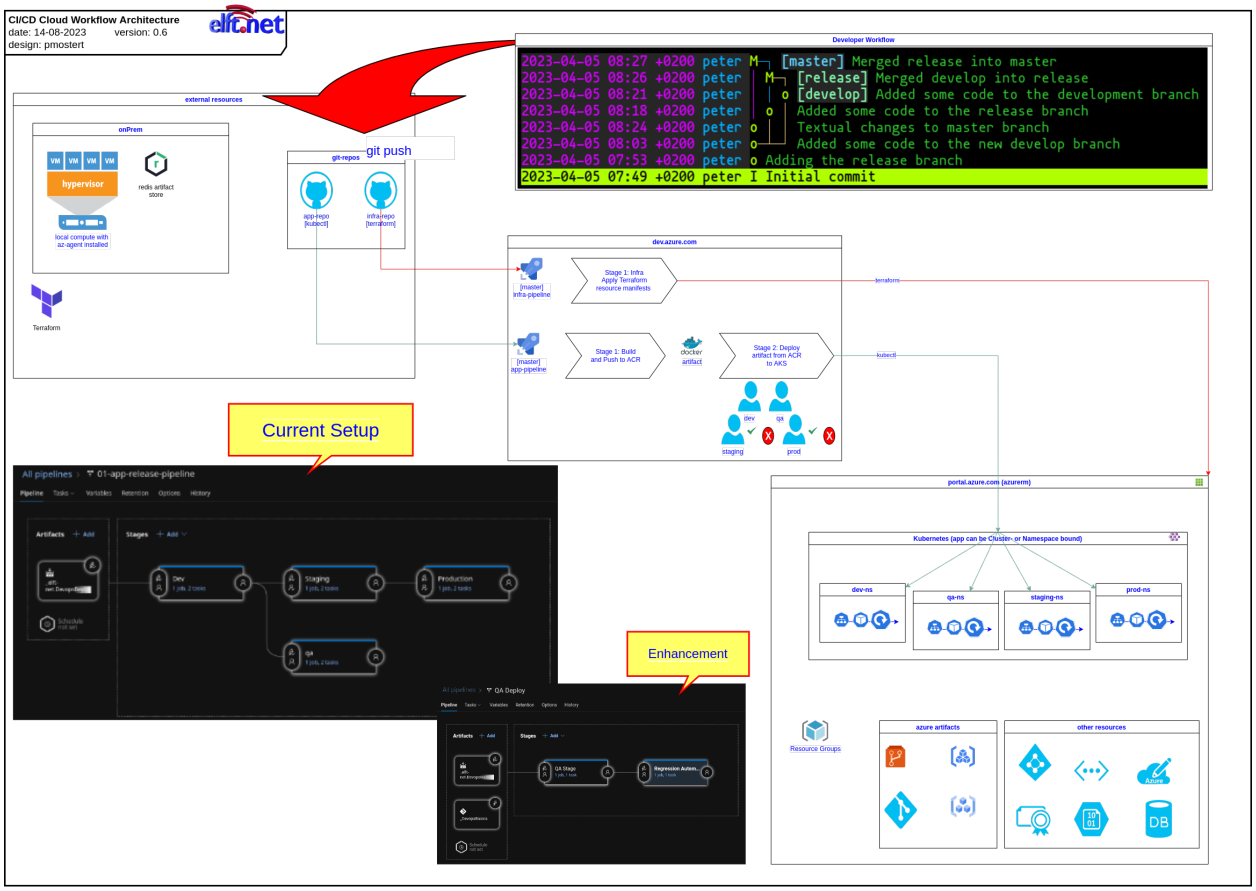
About this assignment

Goal: Having fun playing with technology figuring out where my quality time get spend.

Stakeholders: Just me showing things off for now.

Management survey

Proposed Architecture



```
2023-04-05 08:27 +0200 peter M┐ [master] Merged release into master
2023-04-05 08:26 +0200 peter │ M┐ [release] Merged develop into release
2023-04-05 08:21 +0200 peter │ │ o [develop] Added some code to the development branch
2023-04-05 08:18 +0200 peter │ o │ Added some code to the release branch
2023-04-05 08:24 +0200 peter o │ │ Textual changes to master branch
2023-04-05 08:03 +0200 peter o─┴─┘ Added some code to the new develop branch
2023-04-05 07:53 +0200 peter o Adding the release branch
2023-04-05 07:49 +0200 peter I Initial commit
```

Figure 1: CI/CD workflow

The world is three domains

From an Azure Devops perspective the world is divided into three domains: - portal.azure.com (resource portal): provisions all other Azure Cloud resources (Kubernetes Clusters), IaC can be provided by Terraform. Works if you manage to keep state somewhere save. - dev.azure.com (devops portal): contains resources like projects, service connections and pipelines. Required manual configuration - other resource (external resources): onPrem, github, . . .

Service Connections to hook things up

The domains managed from within dev.azure.com and accessed by configuring "service connections" Currently the dev.azure.com domain has to be manually configured. Trying to force automation did not result in a production grade environment.

Resource Structure

- [external] github contains two repositories
  - infra-repo
  - app-repo
- [devops] within devops 2 projects each containing one pipeline have been defined. Multiple Service Connections have been configured to hook things up.
- [azurerm] resources will be deployed by Terraform as IaC. Basic setup (subscription) should be available.

Infra Deployment

Terraform takes care that the resources within Azure Cloud reflect the definitions within the Terraform manifests. Resource provisioning is handled by IaC. When the infra-repo gets changed/pushed the AKS cluster gets deployed within Azure. Namespaces (dev, qa, staging, prod) are utilized to keep the deployments apart. Only the build-pipeline is used to accomplish this (drawing)

Application Deployment

When a developer pushes code the app-pipeline gets triggered. As a first step the Docker Image is built and stored as an artifact (hub.docker.com). As a second step the artifact get deployed onto the cluster. Continuous deployment has been set up for QA. In the future we could add a step before the qa step that does regression testing. This will improve availability of the code in the qa-namespace. Only if the app passes regression testing it will be deployed on qa. If not then the developer should be made aware of the issue. This is known as Continuous Deployment.

Deployment within the other namespaces (staging, prod) needs to be approved by a stakeholder. This type of deployment is known as continuous delivery.

Sneak Peek into devops portal

Design considerations:

- dev.azure.com (workflow) and portal.azure.com (resources) have literally been divided into two separate domains. Trying automate the generation of pipelines and service connections did not result in a production grade configuration (things crashed). Came across an article later which describes how devops resources (organizations, projects and pipelines) can be managed with Terraform (Aicheh 2022). Up until now the resources within dev.azure.com have been setup and maintained manually.
- The architecture contains a stage labeled enhancement. This is could be the regression automation Cognizant is currently creating. The regression automation stage should be placed before the QA-stage and not after as shown in the architecture.
- Terraform (tf) is used for infra as code. Pushing a code changes to the infra-repo triggers tf to match up the stored state with the resource state in Azure. Terraform is Cloud Agnostic in the sense that it supports many resource providers (Azure, AWS, GCP, vmWare, etc.)
- The terraform manuscripts have been designed to setup 2 AKS Clusters within Azure (dev, prod). The idea here is that different Kubernetes versions can be used within each cluster. Using Separate Clusters will introduce an additional security boundary.
- Environments are contained within namespaces in the Kubernetes Cluster (AKS). Currently the dev, qa, staging and prod namespaces have been defined within one cluster (dev-cluster). The service connection within Azure Devops is leading in where resources will be referenced. Therefore it should be possible to point the service connections for the dev and qa namespace to the dev-cluster while the service connections for the staging and prod namespace point to the prod-cluster.

- Path based routing support: Currently access is based upon IP-number (curl http://20.4.94.183/) . In the end we should look into path based routing => (curl https://elft.net/logistics, https://elft.net/warehousing, https://elft.net/supplychain, https://elft.net/planning). If a workflow is readily available then we should try to merge Azure Devops into the flow.

Workflow

Commands:

- cdd
- cds

Locations:

- Automation Pipeline: DevopsAutomation/Git-Repo-Files/pipeline-backups
- Automation Terraform: DevopsAutomation/terraform-manifests
- Automation Manifest: DevopsAutomation/kube-manifests/01-Deployment-and-LoadBalancer-Service.yml
- App BuildPipeline: DevopsBeyond/01-Azure-DevOps-with-AKS/00-01-Azure-DevOps-BuildandPush-to-ACR/01-build-pipeline.yaml
- App Code Dockerf : DevopsBeyond/01-Azure-DevOps-with-AKS/00-01-Azure-DevOps-BuildandPush-to-ACR/Git-Repository-files/Dockerfile
- App Code Manifest: DevopsAutomation/kube-manifests/01-Deployment-and-LoadBalancer-Service.yml

Next Steps

TBD:

- feedback
- recent events

Azure as a Resource Provider

- AWS SSM (Simple System Manager): Provides many options to access and manage cloud resources in a secure way. Azure Bastion is mentioned as a replacement but isn't.
- Buckets NFS vs Azure Blob storage: First impression Azure Blob storage has less options but that might well be me.
- NFS vs SMB: Working with SMB-4 on Linux VMs was surprisingly transparent (needed for artifacts)
- Azure Support: Issuing Azure Tickets is like moving to /dev/null. Their Support Bots are really bad and annoying.

Aspect	AWS	Azure
Infrastructure	Virtual Private Cloud, Amazon EC2, Amazon Lightsail	Microsoft Azure Virtual Machines, Azure App Services, Azure Functions
Storage	Amazon S3, Amazon EBS, Amazon EFS, Amazon Glacier	Microsoft Azure Storage, Azure Databricks, Azure Data Lake Store
Computing	AWS EC2 Auto Scaling, AWS Lambda, Amazon Elastic Container Service	Azure App Service, Azure Autoscaling, Azure Container Instances, Azure Batch
Database	Amazon RDS, Amazon Aurora, Amazon RedShift	Azure CosmosDB, Azure Database for MySQL, Azure Database for MariaDB, Azure Database for PostgreSQL
Network	Uses VPC	Supports Virtual Networks as well as Express Route
System Management	Uses CloudWatch and CloudFormation	Uses Azure Monitor and Azure Resource Manager
Pricing	Pay-As-You-Go (on-demand instances), Spot Instances, Reserved Instances	Pay-As-You-Go (on-demand instances), Reserved VM Instances, License Included VM Instances, Low Priority VMs

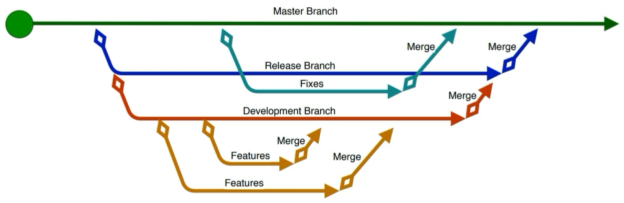
Building

- github or Azure repo:
  - Source code Repository
- Microsoft Azure Resources (https://portal.azure.com)
  - Storage (Artifacts)
  - Container Management (AKS)
- Microsoft Azure DevOps (https://dev.azure.com)
- Terraform
- Documentation: https://azure.microsoft.com, https://devops.microsoft.com, https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs

Testing (TBD)

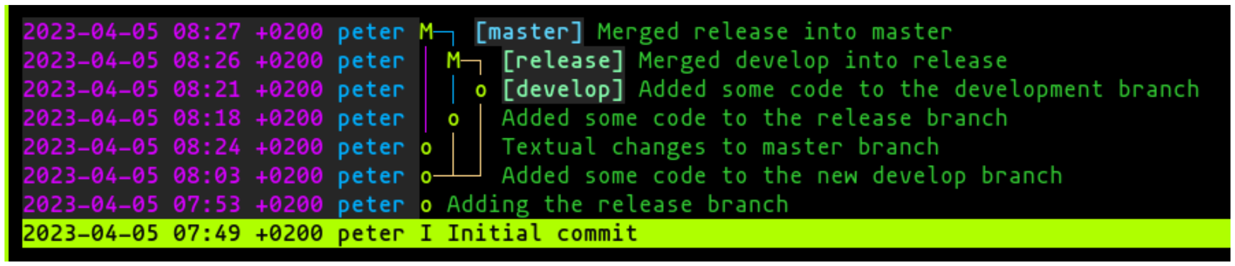
- Unit Testing
- User Interface Testing
- Code Quality Checks
- Security Vulnerability Testing
- Performance Testing
- Integration Testing

Git Branching strategy



Features are build from Stories

Or in real life:



```
2023-04-05 08:27 +0200 peter M─┐  [master] Merged release into master
2023-04-05 08:26 +0200 peter │  M─┐ [release] Merged develop into release
2023-04-05 08:21 +0200 peter │  │ o [develop] Added some code to the development branch
2023-04-05 08:18 +0200 peter │  o │ Added some code to the release branch
2023-04-05 08:24 +0200 peter o │ │ Textual changes to master branch
2023-04-05 08:03 +0200 peter o──┘ Added some code to the new develop branch
2023-04-05 07:53 +0200 peter o Adding the release branch
2023-04-05 07:49 +0200 peter I Initial commit
```

Figure 2: Git Branches Real

- Azure DevOps Architecture

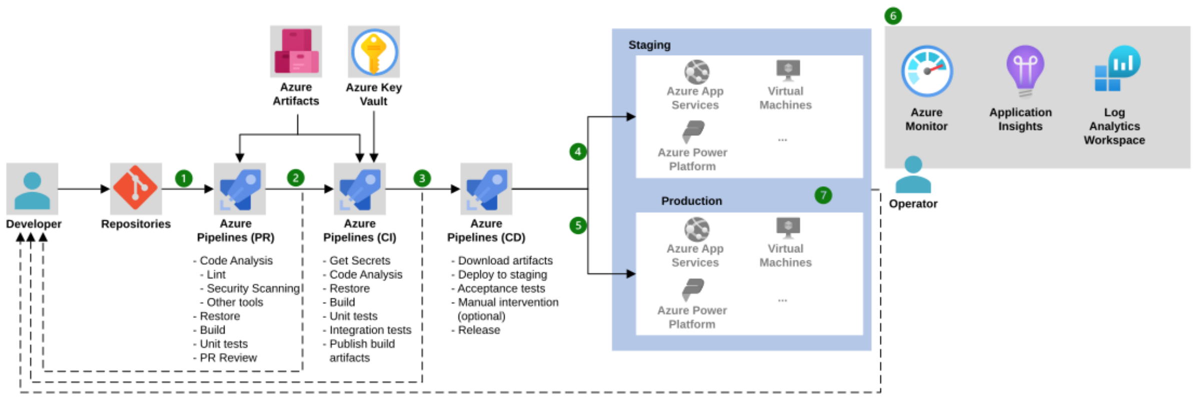
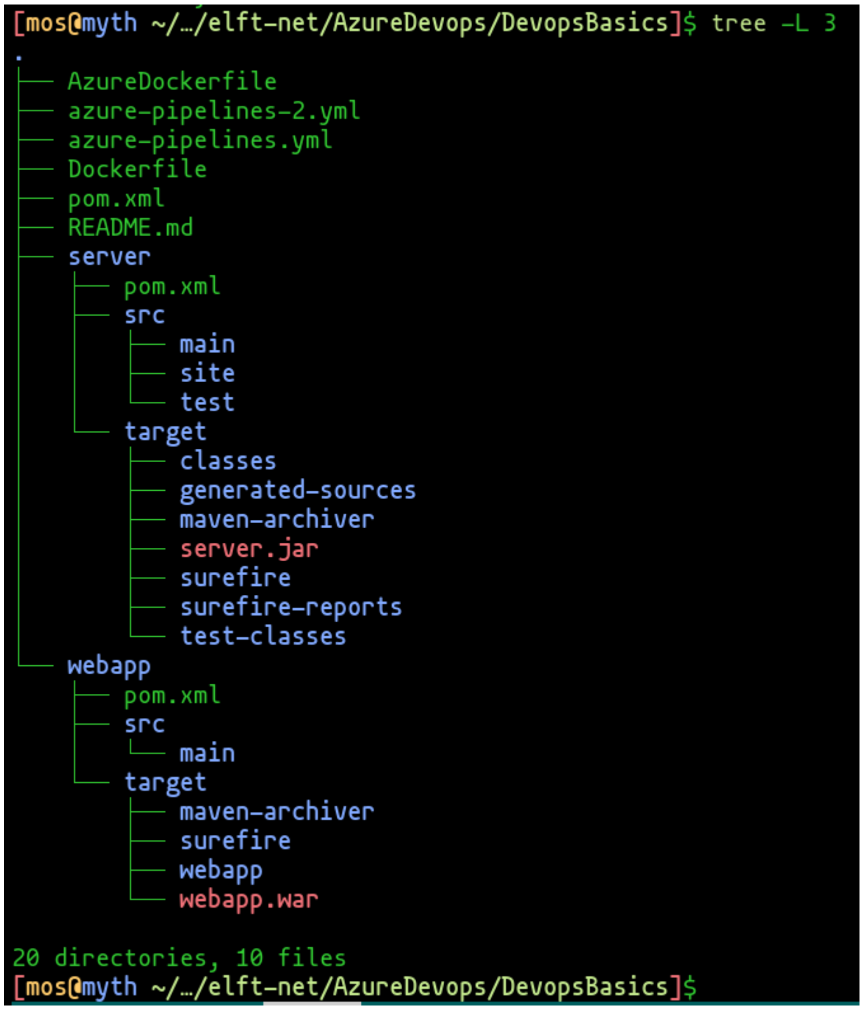


Figure 3: DevOps Workflow

- Azure Container Registry (ACR) replacement for github.
- AppService WebApp
- Docker Image
- WebHooks => Registry level or bound to tag



```
[mos@myth ~/…/elft-net/AzureDevops/DevopsBasics]$ tree -L 3
.
├── AzureDockerfile
├── azure-pipelines-2.yml
├── azure-pipelines.yml
├── Dockerfile
├── pom.xml
├── README.md
├── server
│   ├── pom.xml
│   ├── src
│   │   ├── main
│   │   ├── site
│   │   └── test
│   └── target
│       ├── classes
│       ├── generated-sources
│       ├── maven-archiver
│       ├── server.jar
│       ├── surefire
│       ├── surefire-reports
│       └── test-classes
└── webapp
    ├── pom.xml
    ├── src
    │   └── main
    └── target
        ├── maven-archiver
        ├── surefire
        ├── webapp
        └── webapp.war

20 directories, 10 files
[mos@myth ~/…/elft-net/AzureDevops/DevopsBasics]$
```

Figure 4: Source Code (github)

Hands-on

Lab 1: Azure Devops: Basics

Build Pipelines:

The purpose of the Build Pipeline is to build an Artifact from source. Artifacts are materialized in an Artifact Store to get deployed by the Release Pipelines as a later step. Both Build-Pipelines use Ubuntu in the background for the Azure Build Process.

- First Build Pipeline is transpiling a .war file from source. .war-files are handled by Tomcat => Read an Azure App running Tomcat.
- Second Build Pipeline transpiles the same code and uses a Dockerfile to create a Docker Image that bundles the .was file with Tomcat. The image is externalized to hub.docker.com. The Docker Workflow stops here for now (no release).
- Both Build Pipelines get triggered by Source Code Changes in main.

First Pipeline:

```yaml
# Maven
# Build your Java project and run tests with Apache Maven.
# Add steps that analyze code, save build artifacts, deploy, and more:
# https://d ocs.microsoft.com/azure/devops/pipelines/languages/java

trigger:
- main

pool:
  vmImage: ubuntu-latest

steps:
- task: Maven@3
  inputs:
    mavenPomFile: 'pom.xml'
    mavenOptions: '-Xmx3072m'
    javaHomeOption: 'JDKVersion'
    jdkVersionOption: '1.8'
    jdkArchitectureOption: 'x64'
    publishJUnitResults: true
    testResultsFiles: '**/surefire-reports/TEST-*.xml'
    goals: 'package'
- task: CopyFiles@2
  inputs:
    Contents: '**/*.war'
    TargetFolder: '$(build.artifactstagingdirectory)'
- task: PublishBuildArtifacts@1
  inputs:
    PathtoPublish: '$(Build.ArtifactStagingDirectory)'
    ArtifactName: 'drop'
    publishLocation: 'Container'
```

Second Pipeline:

```yaml
# Maven
# Build your Java project and run tests with Apache Maven.
# Add steps that analyze code, save build artifacts, deploy, and more:
# https://docs.microsoft.com/azure/devops/pipelines/languages/java
```

```
trigger:
- main

pool:
  vmImage: ubuntu-latest

steps:
- task: Maven@3
  inputs:
    mavenPomFile: 'pom.xml'
    mavenOptions: '-Xmx3072m'
    javaHomeOption: 'JDKVersion'
    jdkVersionOption: '1.8'
    jdkArchitectureOption: 'x64'
    publishJUnitResults: true
    testResultsFiles: '**/surefire-reports/TEST-*.xml'
    goals: 'package'
- task: Docker@2
  inputs:
    containerRegistry: 'dockerhub-sc'
    command: 'login'
- task: Docker@2
  inputs:
    containerRegistry: 'dockerhub-sc'
    repository: 'pmostert / devopsbasics1'
    command: 'buildAndPush'
    Dockerfile: '**/AzureDockerfile'
```

Dockerfile to create Container Image:

```
FROM tomcat:latest
COPY ./webapp.war /usr/local/tomcat/webapps
RUN cp -r /usr/local/tomcat/webapps.dist/* /usr/local/tomcat/webapps
```

Release Pipelines:

Two Release Pipeline have been setup.

Workflow

Two release Pipelines have been introduced: - One Deployments for which two Azure App Services are running, one to deploy QA, another one to deploy PROD. Tomcat is running on both App Services, PROD will be deployed only if the Integration Tests Pass (mvn test). - Another pipeline deploys locally to my Linux desktop with tomcat installed. - The pipeline targets can be defined in Azure Pools. The Azure Pipeline Pool is there by default and deploys in Azure. APLmyth01 has been added and points to my workstation (onPrem).

Change, commit and push the index.js from the main branch of the repository and watch things deploy.

QA-Pipeline passed. Latest changes ended up in Production.

Initially the onPrem Pipeline failed caused by the fresh Tomcat install on my workstation. Created an empty webapp.war file owned by mos which can/will be overwritten by the Pipeline process to get things salvaged:

```
[mos@myth /usr/share/tomcat/webapps]$ sudo touch webapp.war
[mos@myth /usr/share/tomcat/webapps]$ sudo chown mos:mos webapp.war
```

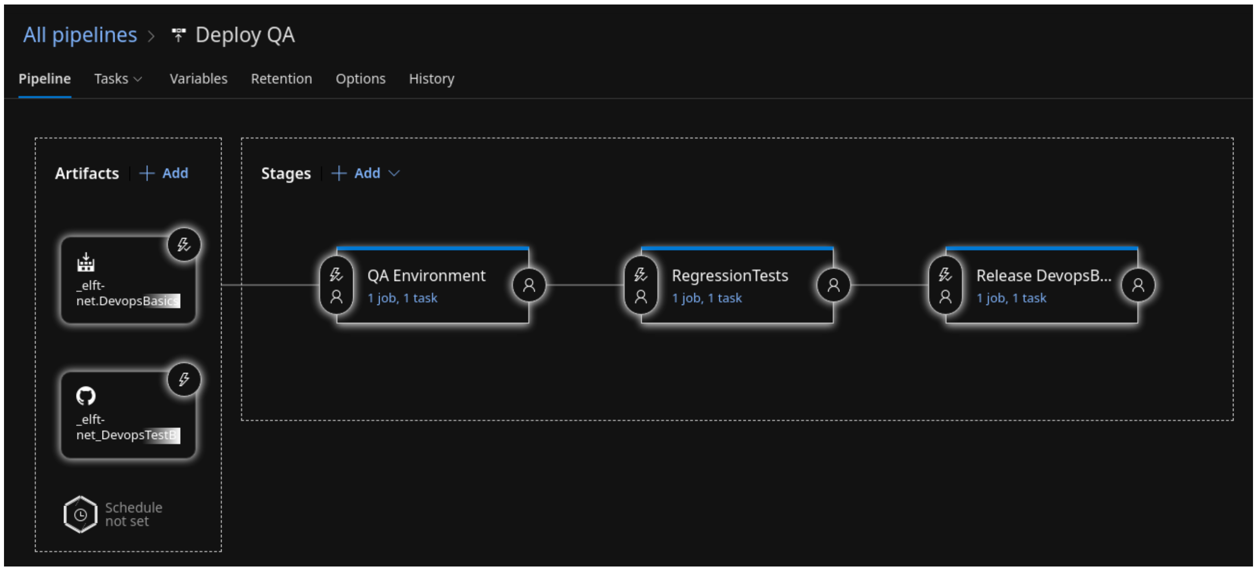


Figure 5: QA Release Pipeline

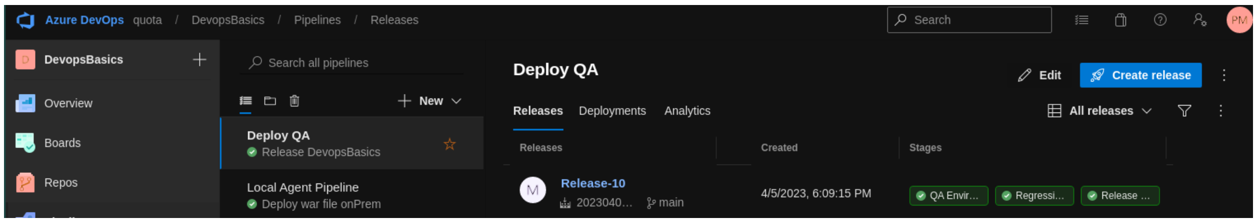


Figure 6: QA Release Pipeline

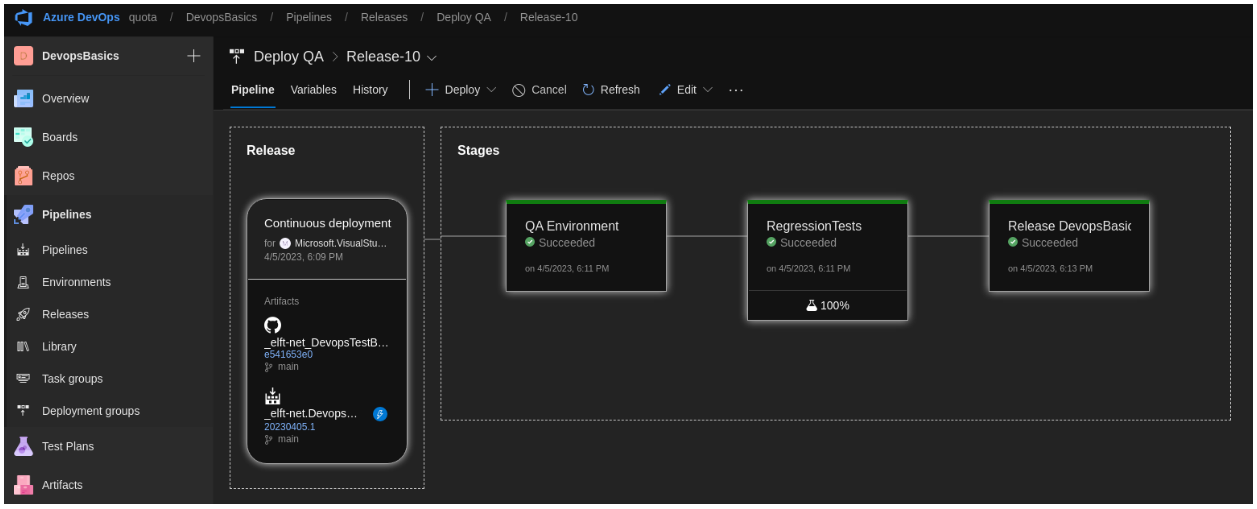


Figure 7: QA Release Pipeline detail

```
[mos@myth /usr/share/tomcat/webapps]$ ls -la
total 0
drwxrwxr-x. 1 root tomcat 20 Apr  5 19:19 .
drwxr-xr-x. 1 root tomcat 14 Apr  5 18:02 ..
-rw-r--r--. 1 mos  mos     0 Apr  5 19:18 webapp.war
[mos@myth /usr/share/tomcat/webapps]$

## Agent output:
[mos@myth ~/.az-agent]$ ./run.sh                                               (:|)  2:07PM
Scanning for tool capabilities.
Connecting to the server.
2023-04-05 12:08:36Z: Listening for Jobs
Error reported in diagnostic logs. Please examine the log for more details.
    - /home/mos/.az-agent/_diag/Agent_20230405-120833-utc.log
2023-04-05 14:56:53Z: Agent connect error: The HTTP request timed out after 00:01:00.. Retrying until r
2023-04-05 14:58:09Z: Agent reconnected.
2023-04-05 17:09:23Z: Running job: Core Agent job
2023-04-05 17:11:21Z: Job Core Agent job completed with result: Failed
2023-04-05 17:24:40Z: Running job: Myth Agent job
2023-04-05 17:26:28Z: Job Myth Agent job completed with result: Succeeded

## Curl
[mos@myth /usr/share/tomcat/webapps]$ curl http://localhost:8080/webapp/
<h1> DevOps Basics Learning Azure</h1>
<h2> Learn QA Automation tools + Devops Tools CI/CD pipelines from Scratch</h2>
<h2> Getting experienced by doing stuff</h2>
<h2> At this stage the CI/CD pipelines are finished. When this change gets committed it should popup in
<h2> Added the Selenium TestCode to the Pipeline and did some minor changes </h2>
<h2> Removed the windows (.exe) binatiies from the repo. Think it was related to the chrome-driver for S
<h2> Selenium: Test should match total text, partial test just fails </h2>
<h2> Added deploy to prod app service to azure and deploy step to pipeline </h2>
<h2> Added a new pipeline that build  a docker image on docker hub.  </h2>
<h2> Added an agent pool that deploys the docker image to my local/onPrem agent/workstation </h2>
<h2> 05-04 at 18:00 Another change while preparing the elft presentation. </h2>
[mos@myth /usr/share/tomcat/webapps]$

## QA App Service
[mos@myth /usr/share/tomcat/webapps]$ curl -L https://devopsbasics-qa.azurewebsites.net
<h1> DevOps Basics Learning Azure</h1>
<h2> Learn QA Automation tools + Devops Tools CI/CD pipelines from Scratch</h2>
<h2> Getting experienced by doing stuff</h2>
<h2> At this stage the CI/CD pipelines are finished. When this change gets committed it should popup in
<h2> Added the Selenium TestCode to the Pipeline and did some minor changes </h2>
<h2> Removed the windows (.exe) binatiies from the repo. Think it was related to the chrome-driver for S
<h2> Selenium: Test should match total text, partial test just fails </h2>
<h2> Added deploy to prod app service to azure and deploy step to pipeline </h2>
<h2> Added a new pipeline that build  a docker image on docker hub.  </h2>
<h2> Added an agent pool that deploys the docker image to my local/onPrem agent/workstation </h2>
<h2> 05-04 at 18:00 Another change while preparing the elft presentation. </h2>
[mos@myth /usr/share/tomcat/webapps]$

## Tests
2023-04-05T16:11:50.5948445Z ----------------------------------------------------------
```

```
2023-04-05T16:11:50.5948737Z  T E S T S
2023-04-05T16:11:50.5949045Z -------------------------------------------------------
2023-04-05T16:11:51.5993678Z
2023-04-05T16:11:51.5994584Z Results :
2023-04-05T16:11:51.5994891Z
2023-04-05T16:11:51.5995389Z Tests run: 0, Failures: 0, Errors: 0, Skipped: 0
2023-04-05T16:11:51.5995665Z
2023-04-05T16:11:51.5996991Z [INFO] ------------------------------------------------------------------------
2023-04-05T16:11:51.5997552Z [INFO] Reactor Summary for Maven Project 1.0-SNAPSHOT:
2023-04-05T16:11:51.5997845Z [INFO]
2023-04-05T16:11:51.6000603Z [INFO] Maven Project ..................................... SUCCESS [  0.0
2023-04-05T16:11:51.6001277Z [INFO] Server ............................................ SUCCESS [ 11.7
2023-04-05T16:11:51.6001913Z [INFO] Webapp ............................................ SUCCESS [  1.5
2023-04-05T16:11:51.6002903Z [INFO] ------------------------------------------------------------------------
2023-04-05T16:11:51.6021379Z [INFO] BUILD SUCCESS
2023-04-05T16:11:51.6021830Z [INFO] ------------------------------------------------------------------------
2023-04-05T16:11:51.6022267Z [INFO] Total time:  13.546 s
2023-04-05T16:11:51.6024020Z [INFO] Finished at: 2023-04-05T16:11:51Z
2023-04-05T16:11:51.6024775Z [INFO] ------------------------------------------------------------------------
2023-04-05T16:11:51.6613471Z Code analysis is disabled outside of the build environment. Could not find
2023-04-05T16:11:52.7574733Z Result Attachments will be stored in LogStore
2023-04-05T16:11:52.7780935Z Run Attachments will be stored in LogStore
2023-04-05T16:11:52.8377923Z ##[section]Async Command Start: Publish test results
2023-04-05T16:11:53.0646863Z Publishing test results to test run '56'.
2023-04-05T16:11:53.0682651Z TestResults To Publish 2, Test run id:56
2023-04-05T16:11:53.0722490Z Test results publishing 2, remaining: 0. Test run id: 56
2023-04-05T16:11:53.8011365Z Published Test Run : https://dev.azure.com/quota/DevopsBasics/_TestManageme
2023-04-05T16:11:53.9003034Z ##[section]Async Command End: Publish test results
2023-04-05T16:11:53.9004516Z ##[section]Finishing: Maven D:\a\r1\a/_elft-net_DevopsTestBasics/pom.xml

## Prod App Service
[mos@myth /usr/share/tomcat/webapps]$ curl -L https://devopsbasics-prod.azurewebsites.net
<h1> DevOps Basics Learning Azure</h1>
<h2> Learn QA Automation tools + Devops Tools CI/CD pipelines from Scratch</h2>
<h2> Getting experienced by doing stuff</h2>
<h2> At this stage the CI/CD pipelines are finished. When this change gets committed it should popup in
<h2> Added the Selenium TestCode to the Pipeline and did some minor changes </h2>
<h2> Removed the windows (.exe) binatiies from the repo. Think it was related to the chrome-driver for S
<h2> Selenium: Test should match total text, partial test just fails </h2>
<h2> Added deploy to prod app service to azure and deploy step to pipeline </h2>
<h2> Added a new pipeline that build  a docker image on docker hub.  </h2>
<h2> Added an agent pool that deploys the docker image to my local/onPrem agent/workstation </h2>
<h2> 05-04 at 18:00 Another change while preparing the elft presentation. </h2>
[mos@myth /usr/share/tomcat/webapps]$
```

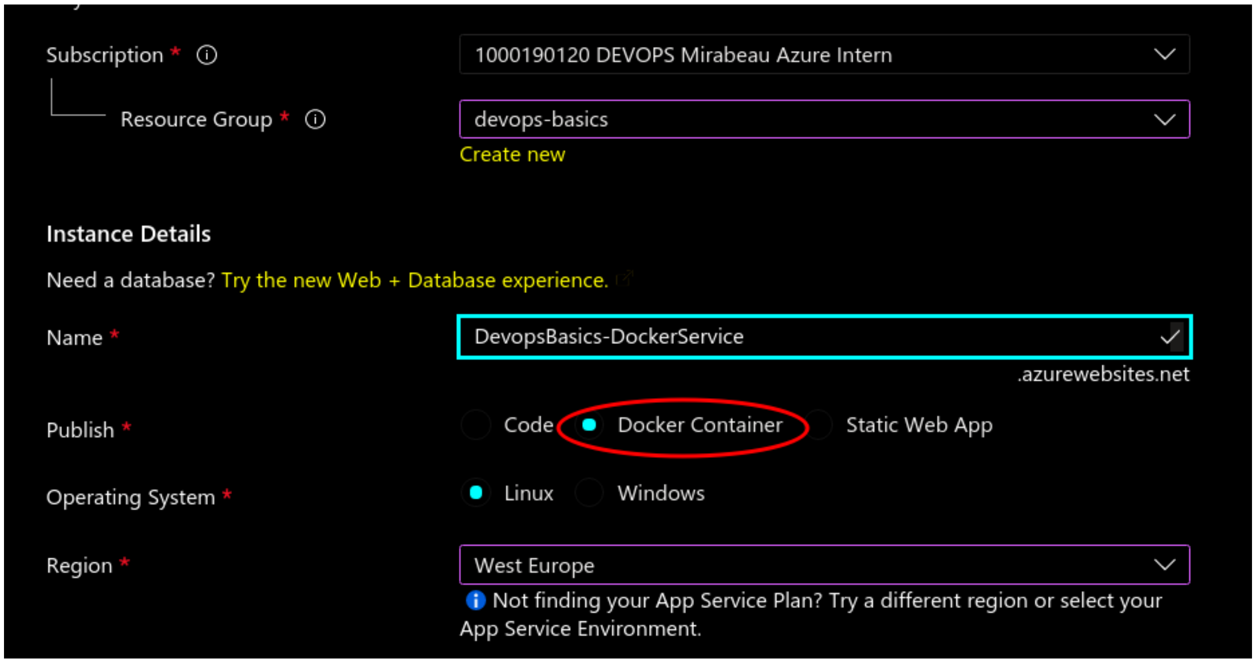
Findings

Azure App Services are way more agile and well integrated into the eco-system than the AWS opponent (EBT= Elastic Beans Talk). Just needed to create an App Service and tell it what stack whould be active (.NET, ASP.NET, Java, Node, PHP, Python or Ruby). Initially I planned to deploy the Docker Image we created above but it doesn't seem to be able to do that «??? - See NOTE:»

There is no yaml file for the release Pipeline which is odd, pitiful and a shame (AWS has one). Within Azure DevOps release pipelines are managed in the Azure Pipelines graphical editor. The YAML files that exist in

Azure DevOps are used to define build pipelines and task definitions. The graphical editor provides a way to manage complex release pipelines that include multiple tasks but automating release pipeline creation might be problematic !!!

NOTE: Later I found that a Docker image can be deployed within an Azure App Service. Azure App Service provide two kinds of web hosting services: App Service Web App for containers and Azure Container Instances (ACI). When deploying a Docker image in App Service, you can select an existing image from Docker Hub, Azure Container Registry, or your own private registry. App Service Web Apps also accept popular container formats like Compose and Kubernetes YAML, so you can use these formats to deploy multi-container applications through App Service. ACI provides a simple and fast way to run containerized applications, without the overhead of virtual machine management. You can deploy a Docker image to ACI without needing to create or manage any infrastructure.



Subscription * : 1000190120 DEVOPS Mirabeau Azure Intern

Resource Group * : devops-basics

Create new

Instance Details

Need a database? Try the new Web + Database experience.

Name * : DevopsBasics-DockerService .azurewebsites.net

Publish * : Code [✓ Docker Container] Static Web App

Operating System * : [✓ Linux] Windows

Region * : West Europe

Not finding your App Service Plan? Try a different region or select your App Service Environment.

Figure 8: Docker not in Code

Might try this later for now let's see if the Docker Image works locally:

```
[mos@myth /usr/share/tomcat/webapps]$ docker run -d pmostert/devopsbasics1:latest
Emulate Docker CLI using podman. Create /etc/containers/nodocker to quiet msg.
 docker.io/pmostert/devopsbasics1:latest
Trying to pull docker.io/pmostert/devopsbasics1:latest...
Error: initializing source docker://pmostert/devopsbasics1:latest: reading manifest latest in docker.io
[mos@myth /usr/share/tomcat/webapps]$ docker images
Emulate Docker CLI using podman. Create /etc/containers/nodocker to quiet msg.
REPOSITORY               TAG         IMAGE ID      CREATED       SIZE
quay.io/podman/hello     latest      26b987a99508  3 months ago  82.6 kB
docker.io/kindest/node   <none>      d8644f660df0  5 months ago  903 MB
[mos@myth /usr/share/tomcat/webapps]$ docker run -d pmostert/devopsbasics1:74
Emulate Docker CLI using podman. Create /etc/containers/nodocker to quiet msg.
 docker.io/pmostert/devopsbasics1:74
Trying to pull docker.io/pmostert/devopsbasics1:74...
Getting image source signatures
Copying blob 9477858ddf22 done
Copying blob 9d52462c5181 done
Copying blob a182a611d05b done
Copying blob ad4fe29a3001 done
Copying blob 74ac377868f8 done
Copying blob ac04a5bb8dd2 done
Copying blob 7f1e52e63d94 done
Copying blob cc2062ac4c6a done
Copying blob 655f794338fe done
Copying config 6fa9836f29 done
Writing manifest to image destination
Storing signatures
0868b7cb4eaf4b3fb7412b2da7e2401fa671018472b30bbc5d3520e3d770f920
[mos@myth /usr/share/tomcat/webapps]$ docker ps
Emulate Docker CLI using podman. Create /etc/containers/nodocker to quiet msg.
CONTAINER ID  IMAGE                                 COMMAND          CREATED        STATUS        PORTS
0868b7cb4eaf  docker.io/pmostert/devopsbasics1:74  catalina.sh run  6 seconds ago  Up 6 seconds

[mos@myth /usr/share/tomcat/webapps]$ curl -L http://localhost:8080/webapp
<h1> DevOps Basics Learning Azure</h1>
<h2> Learn QA Automation tools + Devops Tools CI/CD pipelines from Scratch</h2>
<h2> Getting experienced by doing stuff</h2>
<h2> At this stage the CI/CD pipelines are finished. When this change gets committed it should popup in
<h2> Added the Selenium TestCode to the Pipeline and did some minor changes </h2>
<h2> Removed the windows (.exe) binatiies from the repo. Think it was related to the chrome-driver for S
<h2> Selenium: Test should match total text, partial test just fails </h2>
<h2> Added deploy to prod app service to azure and deploy step to pipeline </h2>
<h2> Added a new pipeline that build  a docker image on docker hub.  </h2>
<h2> Added an agent pool that deploys the docker image to my local/onPrem agent/workstation </h2>
<h2> 05-04 at 18:00 Another change while preparing the elft presentation. </h2>
[mos@myth /usr/share/tomcat/webapps]$
```

Lab 2: AKS Production Setup: BeyondBasics

FallBacks from Lab 1:

- we needed to create the Azure App Services and Pipelines manually, using the graphical editor.
- we had no control over in which VNET the services were placed.
- lack of container support . . . . . well at least it looked like at that point in time. Trying AKS next which stands for Azure Kubernetes Services.
- Kubernetes in general consumes a lot of IP-addresses. Try to prevent depletion by assigning large subnets.
- Azure doesn't have something similar like the AWS ALB. NGINX reverse proxy- or a Traefik POD needs to be deployed as a pod when Host based- or IP based routing is needed. Obviously there need to be more than one is redundancy is required.
- For AWS most EKS clusters can be setup within Private Subnets. In this lab a public subnet will be used for convenience, as a consequence no bastion host will be needed to access the cluster(cost savings). TODO: What is common practice in Azure.
- VNET
  - Default Subnet: 10.240.0.0/16
  - Virtual Nodes Subnet: 10.241.0.0/16
  - AKS Cluster Control Plane (Azure Managed)

    - System Node Pool: Linux
    - Load Balancer
    - Public IP

- Terraform: used to create Cloud Resources (IaC - Infrastructure as Code) Terraform provides a Cloud Agnostic Framework to create Cloud Resources for AWS, Azure, GCP and many other Cloud Providers. One Terraform script can be used to generate Resources within for instance AWS and Azure. A particular script implementing the AWS template can NOT be used to create Resources within Azure (it's not that agile).

    - Providers: Looking at the providers on terraform.io it is striking that AWS has many more downloads than Azure indicating that terraform is more popular on AWS.
    - aliased: tf to terraform and kc to kubectl
    - Commands:
        * tf init
        * tf validate
        * tf plan
        * tf apply
        * tf refresh
        * tf show
        * tf providers

- Terraform and Azure Pipelines:

    - Stage 1: Cluster Creation: DEV, QA, PROD environment clusters should be created alike. It should suffice to change the ENVIRONMENT variable to build a specific environment from scratch. Cost savings might lead to additional changes.
    - Stage 2: Pipeline Creation: Terraform should setup the pipelines for the environment
    - Stage 3: Build and Deploy: The Code (assumption: k8s manifests) is located in in a git repository (fi github or azure repo). A pipeline is bound to a specific repo-branch. When changes in the branch get committed the Pipeline build and deploys the Code.

    TODO: Investigate Terraform IDE-integration (run vsc-code plugin by Charles Zipp) Looks like the plugin needs to be installed in Devops

TODO: Reorder after weekend

1. Create Cluster:

```
[mos@myth ~/…/24-Azure-AKS-Terraform/03-Create-AKS-Cluster/terraform-manifests-aks]$ tf init

Initializing the backend...

Successfully configured the backend "azurerm"! Terraform will automatically
use this backend unless the backend configuration changes.

Initializing provider plugins...
- Finding hashicorp/azuread versions matching "~> 2.3"...
- Finding hashicorp/random versions matching "~> 3.4"...
- Finding hashicorp/azurerm versions matching "~> 3.0"...
- Installing hashicorp/azuread v2.36.0...
- Installed hashicorp/azuread v2.36.0 (signed by HashiCorp)
- Installing hashicorp/random v3.4.3...
- Installed hashicorp/random v3.4.3 (signed by HashiCorp)
- Installing hashicorp/azurerm v3.50.0...
- Installed hashicorp/azurerm v3.50.0 (signed by HashiCorp)

Terraform has created a lock file .terraform.lock.hcl to record the provider
```

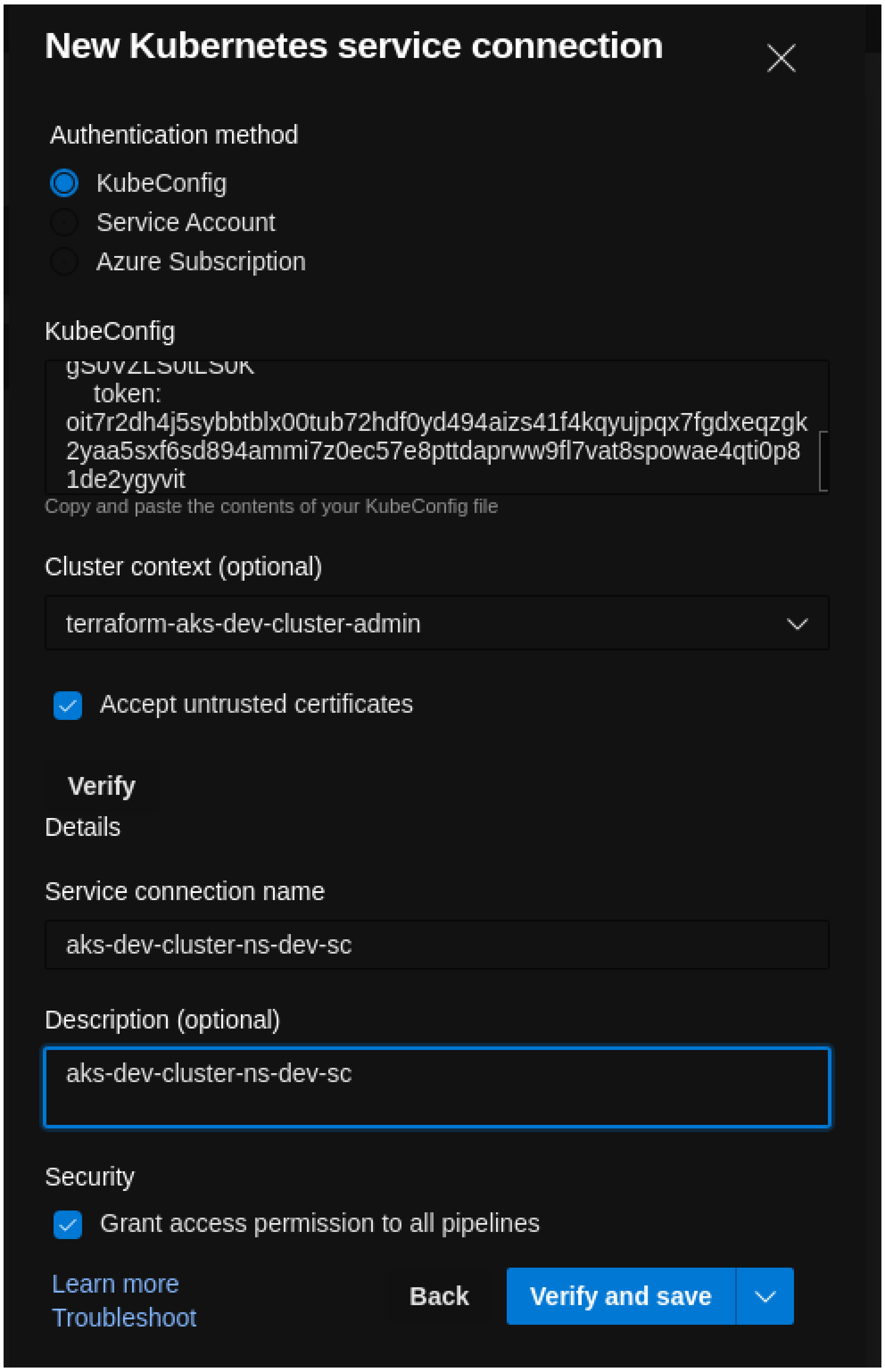
```
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
[mos@myth ~/…/24-Azure-AKS-Terraform/03-Create-AKS-Cluster/terraform-manifests-aks]$ tf apply
azurerm_resource_group.aks_rg: Refreshing state... [id=/subscriptions/058a36cd-ad87-41c0-8431-a09e57423
. . . . .
Apply complete! Resources: 4 added, 0 changed, 0 destroyed.

Outputs:

aks_cluster_id = "/subscriptions/058a36cd-ad87-41c0-8431-a09e57423aee/resourceGroups/terraform-aks-dev/
aks_cluster_kubernetes_version = "1.25.5"
aks_cluster_name = "terraform-aks-dev-cluster"
azure_ad_group_id = "6737337a-54f7-43f2-b614-205ae87da7c7"
azure_ad_group_objectid = "6737337a-54f7-43f2-b614-205ae87da7c7"
latest_version = "1.25.5"
location = "westeurope"
resource_group_id = "/subscriptions/058a36cd-ad87-41c0-8431-a09e57423aee/resourceGroups/terraform-aks-d
resource_group_name = "terraform-aks-dev"
versions = tolist([
  "1.23.12",
  "1.23.15",
  "1.24.6",
  "1.24.9",
  "1.25.4",
  "1.25.5",
])
[mos@myth ~/…/24-Azure-AKS-Terraform/03-Create-AKS-Cluster/terraform-manifests-aks]$ source setenv
[mos@myth ~/…/24-Azure-AKS-Terraform/03-Create-AKS-Cluster/terraform-manifests-aks]$ az aks get-credent

Merged "terraform-aks-dev-cluster-admin" as current context in /home/mos/.kube/config
[mos@myth ~/…/24-Azure-AKS-Terraform/03-Create-AKS-Cluster/terraform-manifests-aks]$ mv ~/.kube/config 
[mos@myth ~/…/24-Azure-AKS-Terraform/03-Create-AKS-Cluster/terraform-manifests-aks]$ kc get no
NAME                                STATUS   ROLES   AGE   VERSION
aks-systempool-15400044-vmss000000   Ready    agent   18m   v1.25.5
[mos@myth ~/…/24-Azure-AKS-Terraform/03-Create-AKS-Cluster/terraform-manifests-aks]$
[mos@myth ~/…/24-Azure-AKS-Terraform/03-Create-AKS-Cluster/terraform-manifests-aks]$ kc apply -f -<<EOF
apiVersion: v1
kind: ServiceAccount
metadata:
   name: dev-admin-sa
   namespace: dev
EOF
```

```
serviceaccount/dev-admin-sa created
[mos@myth ~/…/24-Azure-AKS-Terraform/03-Create-AKS-Cluster/terraform-manifests-aks]$ kc apply -f -<<EOF
apiVersion: v1
kind: ServiceAccount
metadata:
   name: qa-admin-sa
   namespace: qa
EOF

serviceaccount/qa-admin-sa created
[mos@myth ~/…/24-Azure-AKS-Terraform/03-Create-AKS-Cluster/terraform-manifests-aks]$ kc apply -f -<<EOF
apiVersion: v1
kind: ServiceAccount
metadata:
   name: staging-admin-sa
   namespace: staging
EOF

serviceaccount/staging-admin-sa created
[mos@myth ~/…/24-Azure-AKS-Terraform/03-Create-AKS-Cluster/terraform-manifests-aks]$ kc apply -f -<<EOF
apiVersion: v1
kind: ServiceAccount
metadata:
   name: prod-admin-sa
   namespace: prod
EOF

serviceaccount/prod-admin-sa created
[mos@myth ~/…/24-Azure-AKS-Terraform/03-Create-AKS-Cluster/terraform-manifests-aks]$
[mos@myth ~/…/24-Azure-AKS-Terraform/03-Create-AKS-Cluster/terraform-manifests-aks]$ kubectl apply -f -
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: role-for-serviceaccount
  namespace: dev
rules:
- apiGroups: ["*","apps","extensions"]
  resources: ["*"]
  verbs: ["get", "list", "watch", "create", "update", "patch", "delete"]
EOF
role.rbac.authorization.k8s.io/role-for-serviceaccount created
[mos@myth ~/…/24-Azure-AKS-Terraform/03-Create-AKS-Cluster/terraform-manifests-aks]$
 . . . . .
```

Normal service connection (using Service Principle on subscription level) does NOT work. => creating kubernetes sc fails with message: Could not find any secrets associated with the Service Account. This is due to an integration issue between Azure and Devops. Had to the above for ALL namespaces AND configure the following to resolve:

Struggles

Got an hour-glass while invoking commands from within devops portal, digging deep revealed AD-connection issues: GetUserAccessToken: Failed to obtain an access token of identity 4586b572-9dc9-72db-b5e0-dc96d9582059. AAD returned silent failure.



New Kubernetes service connection

Authentication method

[✓ KubeConfig]
Service Account
Azure Subscription

KubeConfig

```
gS0VZLS0tLS0K
    token:
oit7r2dh4j5sybbtblx00tub72hdf0yd494aizs41f4kqyujpqx7fgdxeqzgk
2yaa5sxf6sd894ammi7z0ec57e8pttdaprww9fl7vat8spowae4qti0p8
1de2ygyvit
```

Copy and paste the contents of your KubeConfig file

Cluster context (optional)

terraform-aks-dev-cluster-admin

[✓] Accept untrusted certificates

Verify

Details

Service connection name

aks-dev-cluster-ns-dev-sc

Description (optional)

aks-dev-cluster-ns-dev-sc

Security

[✓] Grant access permission to all pipelines

Figure 9: AKS integration issue

Known issue in AKS 1.24 => caused by: Service Credentials fail to create implicitly behind the scenes.

- Lab 3: Python Scripting in Azure

TODO: Make some Python scripts to play with Azure integrations

Storage: - Upload, Download, Copy, List, Delete files from/to Azure Storage Instance: - Menu with List Running, Access, Stop . . . Azure Instances AKS: - List, Change and Manage Environments on a high level Serverless: - Gernerate static site from markdown (Hugo) and put it in a publicly shared Azure Blob (Shouldn't need Python).

- Lab 4: Java Springboot

Java Spring Boot is a popular open-source framework used to develop applications and services on the Java platform. Spring Boot simplifies application development by providing a vast array of features and tools for rapid application development, such as its "starter projects" feature and its "spring-boot-maven-plugin." The framework helps developers create applications using a "convention over configuration" approach that reduces tedious and time-consuming configuration and setup tasks. Additionally, Spring Boot supports various servers, including Tomcat and Jetty, as well as a variety of databases and frameworks, including Hibernate and Spring MVC.

```
[mos@myth ~/…/03-Azure-AKS-Terraform/03-Create-AKS-Cluster/terraform-manifests-aks]$ cat setenv
# az aks get-credentials --name MyManagedCluster --overwrite-existing --resource-group MyResourceGroup
# trigger `source setenv` command to apply.
export AKS_CLUSTER=terraform-aks-dev-cluster
export AKS_RESOURCE_GROUP=terraform-aks-dev
# az aks get-credentials --name ${AKS_CLUSTER}  --resource-group ${AKS_RESOURCE_GROUP} --admin
[mos@myth ~/…/03-Azure-AKS-Terraform/03-Create-AKS-Cluster/terraform-manifests-aks]$

source setenv
az aks get-credentials --name ${AKS_CLUSTER}  --resource-group ${AKS_RESOURCE_GROUP}

dev-stack:
AKS_CLUSTER=devops-tfstate-dev-cluster
AKS_RESOURCE_GROUP=devops-tfstate-dev
az aks get-credentials --name ${AKS_CLUSTER}  --resource-group ${AKS_RESOURCE_GROUP} --admin
```

Python use case: TODO: use az ask command to build a list of clusters, resource groups and namespaces. Create a separate config file for the cluster if none exists (currently clusters will be merged into one config file. Show a menu where a specfic cluster, namespace can be selected and switch to it.

Addendum

TBD

References

Aicheh, Damien. 2022. "Configure Your Azure DevOps Projects Automatically with Terraform." https://stackoverflow.com/questions/69721907/nexus-repository-azure-devops-release-pipeline.