



TEMPLATE-BASED WEB APP AND DEPLOYMENT PIPELINE ON AZURE IN AN ENTERPRISE ENVIRONMENT

Dr. Johannes Schöck

Senior Expert Data Science, NKD Group GmbH

*Nürnberg Data Science & AI Meetup
@Nürnberg Digital Festival
01.07.2025*

ABOUT ME // CODE AND SLIDES

Johannes Schöck

- Studied molecular nano science, PhD in physics
- Self-taught coding and data science skills
- Transitioned into data science since 2019
- Data Scientist at NKD since 2023



 Find me on LinkedIn

Code and slides @GitHub



 [https://github.com/JSchoeck/
talk_webapp_template_and_pipeline_on_azure](https://github.com/JSchoeck/talk_webapp_template_and_pipeline_on_azure)

NKD





NKD GROUP

NKD Online Shop

<https://www.nkd.de/>

NKD Career

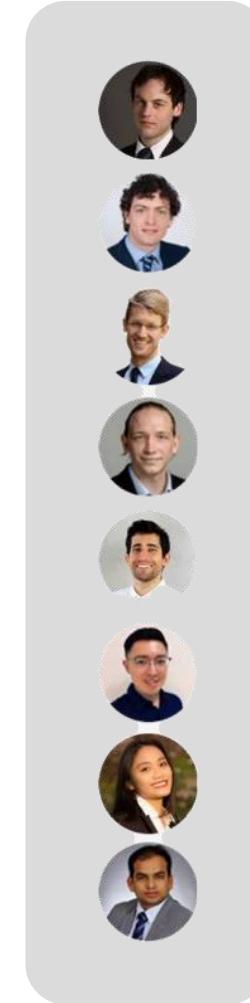
<https://www.nkdgroup.com/>

NKD Data Science Team

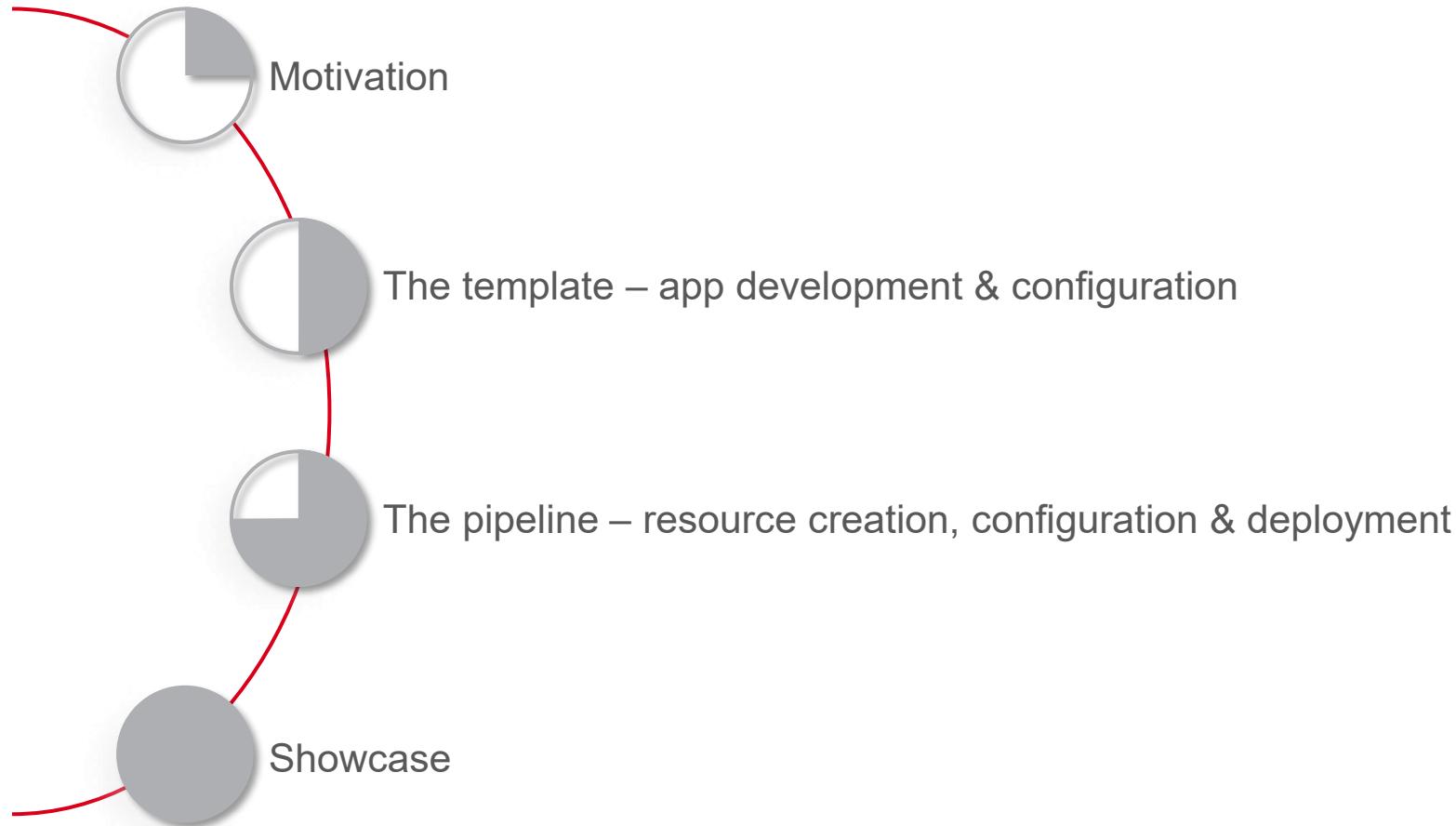
- 6 Data Scientists
- 1 Cloud Engineer
- 1 Working Student

+ MIS Team for data and database topics

Pipeline and template co-developed with [Luis Cuervo](#) & [Sai-Ram Nallamothu](#)

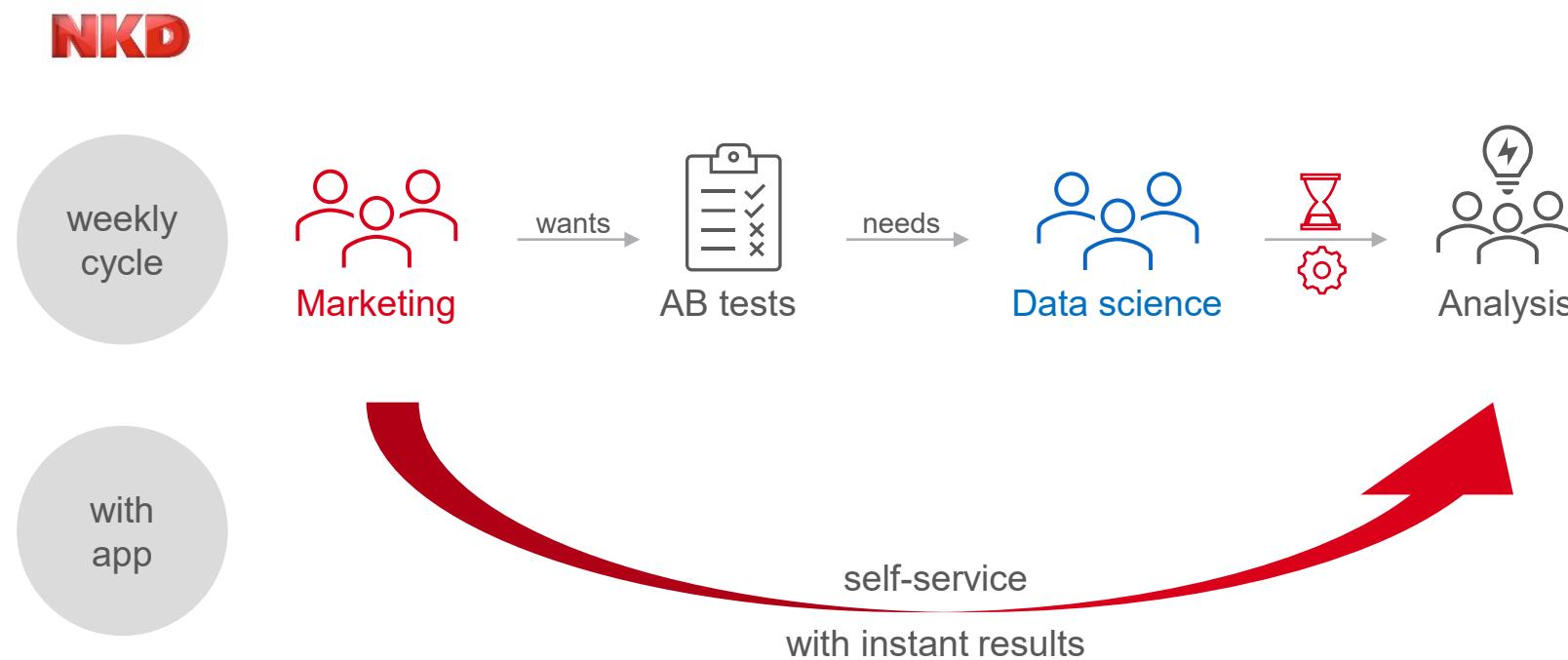


OUTLINE



MOTIVATION

- To increase the **impact of data science teams**, they should not work on user requests
- Instead, companies should **enable business users** to get the data and predictions they need for a specific task on their own
- Building **data-driven applications** empowers business users to be faster, more informed, and deciding better



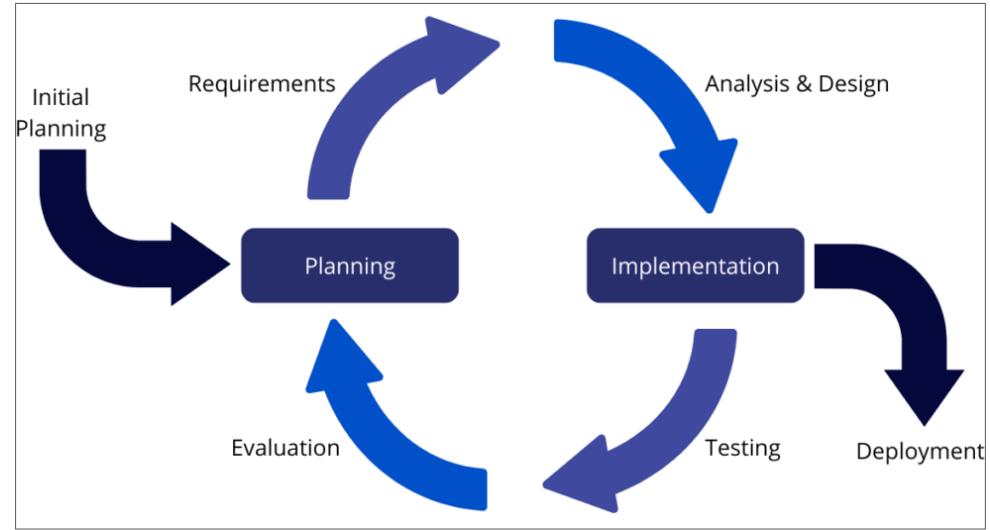
MOTIVATION

Developing a data-driven app – requirements

- Solving the problem at the core of the task
- Using the right data
- Understanding the data

Why it's hard to get user feedback early...

- Complex data interactions: filter, sort, aggregate, time shift...
- Nothing to show to the business users during backend development
- Business users often have limited knowledge about data-driven app peculiarities



...and why that is problematic

- No feedback loop for iterative improvement
- Changing the development goal increases development time and complexity
- Building the wrong thing
- Business rules impact data, software and UI design decisions, but often are not communicated up front
- Poor user experience
- Lack of trust
- Low adoption rate

MOTIVATION

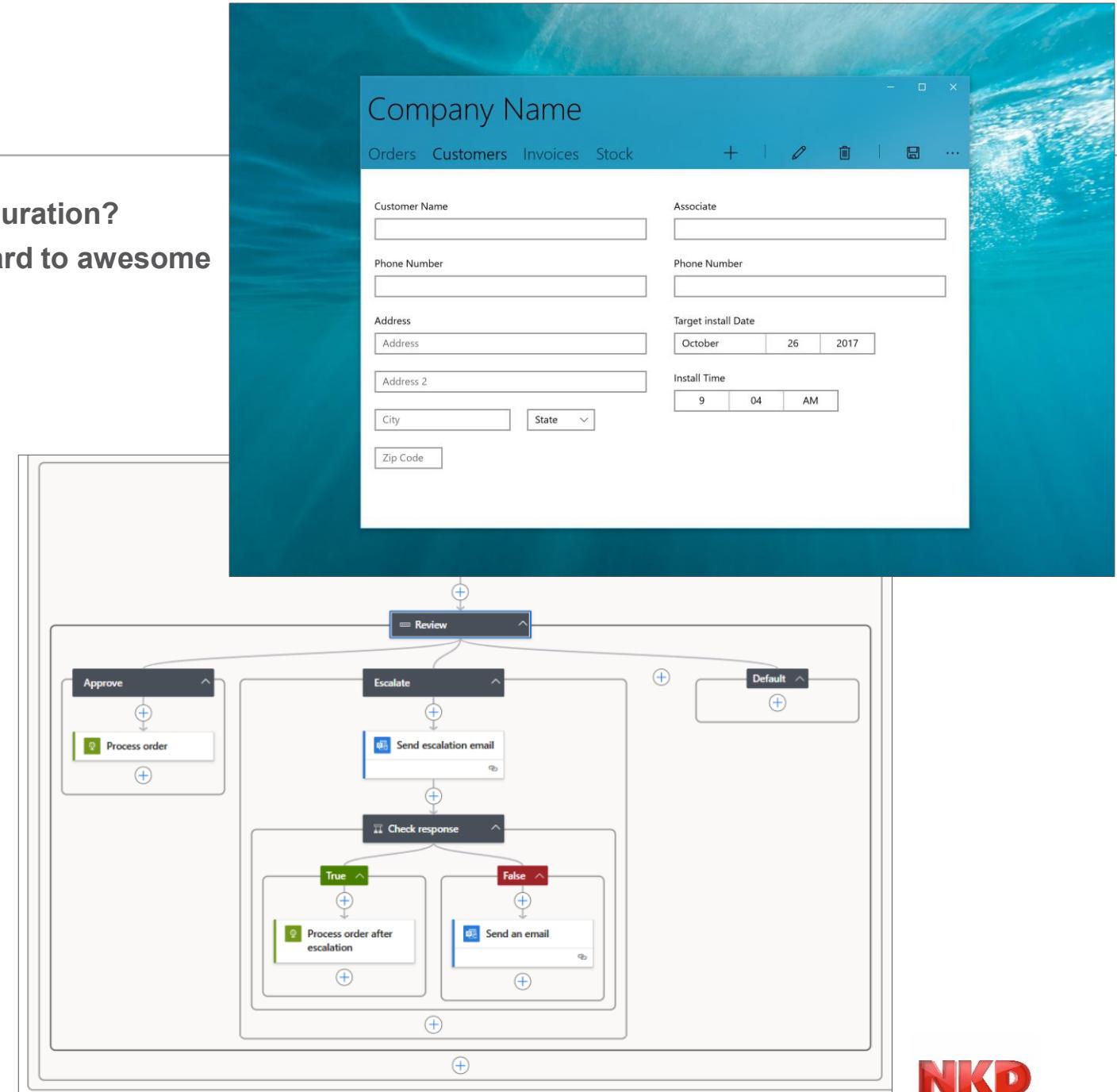
Why it's hard to get a real application running early

- With business users, discussing an app based on its code is usually not possible
→ UI required
- Setting up an app locally limits how business users can interact with it
→ lowers buy-in, does not scale later
- Deploying an app in the cloud leads to additional issues
 - Dependencies
 - Budget approval / costs
 - Knowledge to setup infrastructure
 - Permissions to setup infrastructure
 - Effort to setup infrastructure
 - User authentication
 - User authorization
 - Interaction of data, backend, and frontend

MOTIVATION

What if we could automate app deployment and configuration?
Or, how the NKD data science teams went from awkward to awesome

- Starting point
 - Development in Jupyter notebooks on dev machines
 - Most apps run on request from business users
 - Interactive apps using MS Forms and Logic apps
 - No version control
 - Not file based
 - Hard to parametrize
 - Hard to backup and setup after incident
 - Hard to modularize
 - Hard to debug
 - Outdated user experience



MOTIVATION

What if we could automate app deployment and configuration?

Or, how the NKD data science teams went from awkward to awesome

- Step by step
 - Common coding rules, using a linter and formatter
 - Work with repositories
 - Integrate common code into utility library
 - Build PoC web app with streamlit
 - Learn about pipelines
- Final stage
 - Common development environment as default
 - Template app, based on internal utility library
 - Central pipeline repository with versioning



A screenshot of a Streamlit application interface. At the top left is the NKD Data Science logo. The main title is "AB Test Group Definition". Below the title, there are several input fields and dropdown menus. One dropdown menu is open, showing options like "Gruppenzuordnung", "Kontrollgruppe", "Testgruppe", and "Gruppenzuordnung". There are also other input fields for "Gruppenanzahl", "Gruppenanzahl", and "Gruppenanzahl". The interface has a clean, modern design with a light gray background and red highlights for certain elements.

CHALLENGES

- Collaboration between developers
- Virtual environment and dependencies across different systems
- Many different possible solutions to build data apps on cloud services
- Large variety of involved systems and technologies
 - Code repository
 - CI/CD pipeline
 - Authentication
 - Authorization
 - Backend
 - Frontend
 - Web server
 - Database
 - File storage
 - Resource permissions
 - Package / artifact index
 - Secret management
- Complex details make it hard to get it right
 - Connecting services
 - Many config options
 - Not total control over systems
- Pipeline delay during development
- When not actively developing pipeline, knowledge is lost again
- Updates to the pipeline during project development leads to fragmented work, which must be re-done multiple times

The screenshot shows the Microsoft Azure portal interface. On the left, there is a sidebar menu with the following items:

- Cloud Run
- Cloud Run-Funktionen
- App Engine
- API Gateway
- Endpunkte

The main content area displays a section titled "Popular Azure services" with a "See more" link. It lists several services with their icons and "Create" buttons:

- Function App
- Web App
- Logic App
- Azure AI Search
- App Service Plan
- Container App

On the far right, there is a vertical column of service cards:

- Datenverarbeitung
- Amazon EC2
Virtuelle Server in der Cloud
- Amazon EC2 Auto Scaling
Rechenkapazität skalieren, um Anforderungen zu erfüllen
- Amazon Lightsail
Virtual Private Server starten und verwalten
- AWS App Runner
Erstellen und Ausführen von containerisierten Webanwendungen im großen Maßstab
- AWS Batch
Ausführen von Batch-Jobs beliebiger Größe
- AWS Elastic Beanstalk
Web-Apps ausführen und verwalten
- AWS Lambda
Code ausführen, ohne an Server zu denken

TOOLS

- Dependency Manager – poetry 
- Code Formatter and Linter – ruff 
- Cloud Resource Management – Azure CLI 
- Project Configuration – pyproject.toml 

More details in
this talk

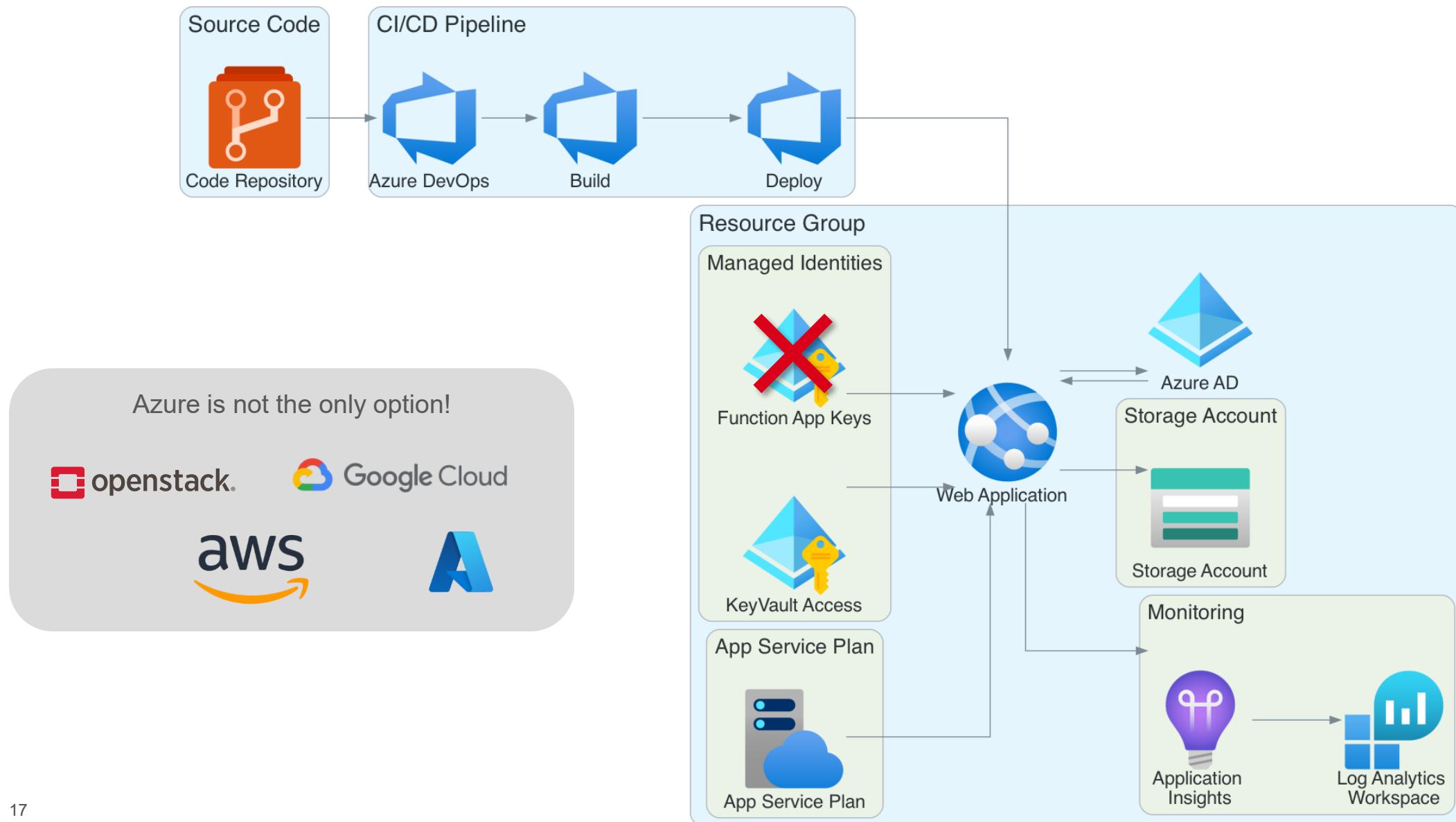


🔗 https://github.com/JSchoeck/talk_python_ruff-poetry
2024 @ Python User Group Nürnberg

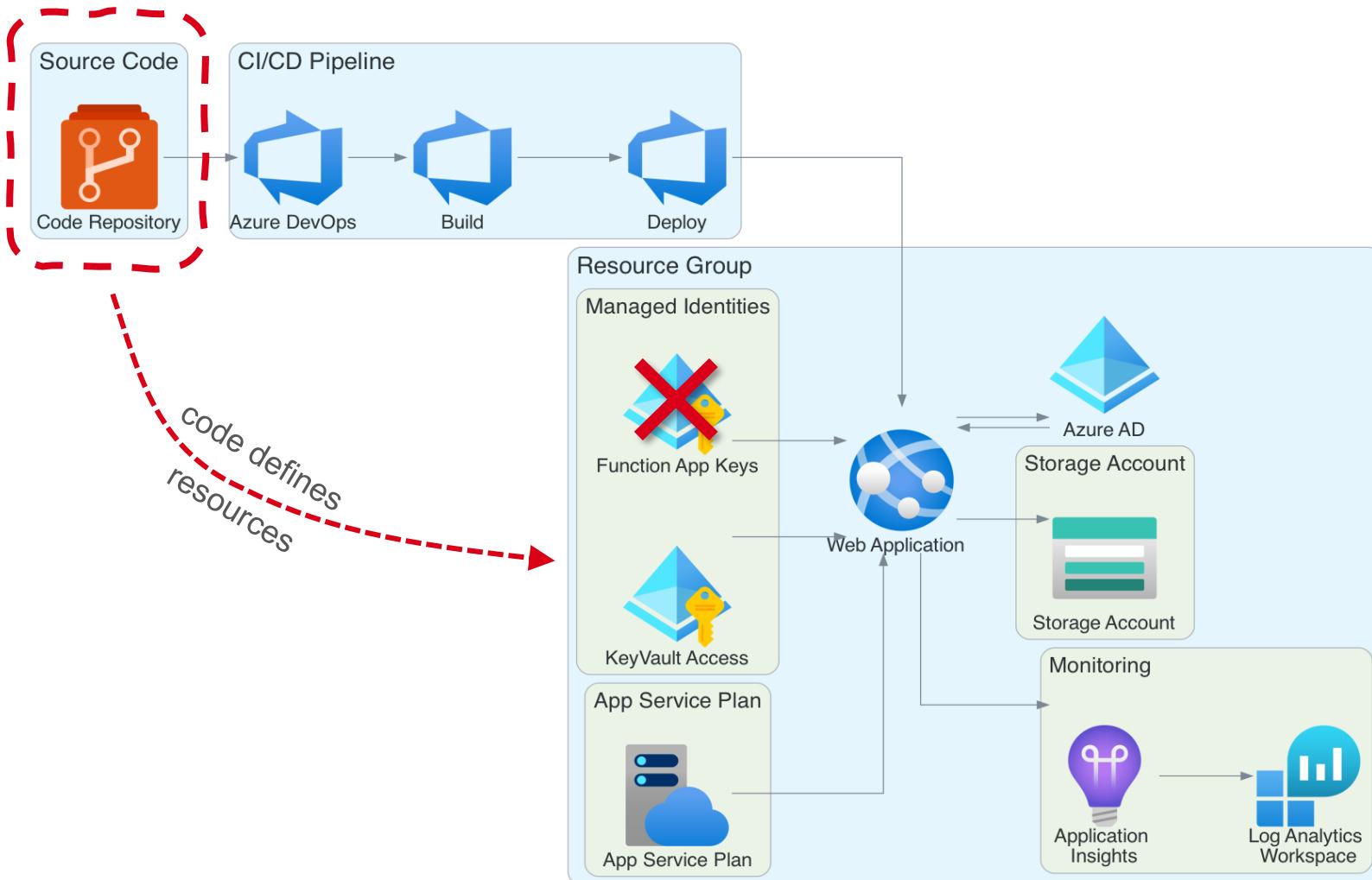
```
1 [project]
2 # TODO: replace with actual project name, version, description, and author(s)
3 name = "TEC_002_Web_App_Template" # Name must have this structure: [type in 3 letters: TEC/PAS/PRO]
4 version = "2.0.0"
5 description = "This is a template for Python projects that are hosted as an Azure App Service (i.e.
6 authors = [
7     {name = "Johannes Schöck", email = "Johannes.Schoeck@nkd.de"}, 
8     {name = "Luis Cuervo", email = "Luis.Cuervo@nkd.de"}, 
9     {name = "Sai Nallamothu", email = "SaiRam.Nallamothu@nkd.de"}, 
10 ]
11 readme = "README.md"
12 requires-python = ">=3.11"
13 dependencies = []
14
15 [tool.poetry]
16 package-mode = false
```



AZURE INFRASTRUCTURE



AZURE INFRASTRUCTURE



THE TEMPLATE – APP DEVELOPMENT & CONFIGURATION

- Poetry configuration in poetry.toml

```

1 [virtualenvs]
2 in-project = true
3
4 [virtualenvs.options]
5 always-copy = true

```

- Streamlit configuration in .streamlit\config.toml

```

1 [theme]
2 base="light"
3 primaryColor="#F63366"
4 backgroundColor="#FFFFFF"
5 secondaryBackgroundColor="#FADEDE" # production: F0F2F6; development: FADEDE
6 textColor="#262730"
7 font="sans-serif"
8
9 [client]
10 toolbarMode="minimal" # https://docs.streamlit.io/develop/concepts/architecture
11 showSidebarNavigation = false # Controls whether the default sidebar page
12 # Alternatively build custom navigation: https://docs.streamlit.io/develop
13
14 [browser]
15 gatherUsageStats = false # always set to false, otherwise user data will

```

- Running & debugging streamlit



```

1 {
2   "version": "0.2.0",
3   "configurations": [
4     {
5       "name": "Streamlit app.py",
6       "type": "debugpy",
7       "request": "launch",
8       "module": "streamlit",
9       "env": {
10         "STREAMLIT_APP": "\\\$src\\\app.py",
11         "PYTHONPATH": "\${workspaceRoot}",
12         "PYDEVD_WARN_SLOW_RESOLVE_TIMEOUT": "30"
13       },
14       "args": [
15         "run",
16         "\${workspaceRoot}\\\$\{STREAMLIT_APP\}"
17       ],
18       "redirectOutput": true,
19       "preLaunchTask": "prepare environment",
20     }
21   ]
22 }

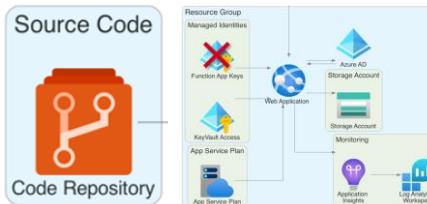
```

```

1 {
2   "label": "poetry install",
3   "type": "shell",
4   // Install dependencies from lock file into environment, without installing
5   "command": "poetry install --no-root",
6   "dependsOn": "poetry lock refresh"
7 }

```

THE TEMPLATE – DEPLOY.BICEP



- Fetch KeyVault ID, to avoid hardcoding it

```
1 resource keyVault 'Microsoft.KeyVault/vaults@2023-07-01' existing = {
2   scope: resourceGroup(subscriptionIdProd, utilitiesResourceGroupName )
3   name: 'kv-dataservice-general'
4
5 }
```

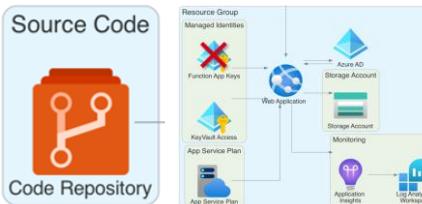
- Create new resource group, with standardized name

```
1 resource newRG 'Microsoft.Resources/resourceGroups@2022-09-01' = {
2   name: 'rg-${projectId}-${ projectName}'
3   location: location
4   tags: {
5     owner: 'data science'
6   }
7 }
```

- Pass parameters to resource creation bicep file

```
1 module resources 'resources.bicep' = {
2   name: 'resources'
3   scope: newRG
4   params: {
5     projectId: projectId
6     projectName: projectName
7     location: location
8     utilitiesResourceGroupName: utilitiesResourceGroupName
9     pathToApp: pathToApp
10    pythonVersion: pythonVersion
11    runFromPackage: runFromPackage
12    blobStorage: blobStorage
13    skuProd: skuProd
14    skuDevtest: skuDevtest
15    existingASPname: existingASPname
16    mGeneralKeyvaultAccess: dsManagedIdentityKeyvaultAccessClientId
17    subscriptionIdProd: subscriptionIdProd
18  }
19 }
```

THE TEMPLATE – RESOURCES.BICEP



1. Define default parameters
2. Set log analytics workspace to existing shared workspace
3. Set app service plan (ASP) to existing shared compute resource
4. “Optional”: create storage account, blob service, and blob container and set accessibility
5. Create application insights, targeting log analytics workspace
6. Create ASP, if not using the shared one
7. Create web application
 - Assign managed IDs to access KeyVaults
 - Set startup command for streamlit
 - Define environment variables to connect services and configure app
8. Create basic blocking identity provider

Compute & costs

- ASPs have monthly fixed cost
- Web apps are not in active use most of the time
- More CPU and RAM → faster app flow
- Sharing compute resources allows high performance with low cost

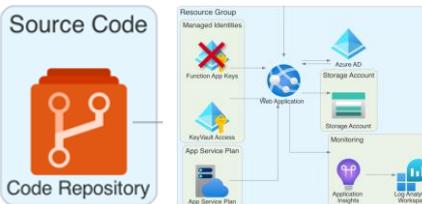


Naming

- Resources have fixed naming schemes
- Split into production and development environment is represented in most resource names by adding “-dev”

```
1 resource webApplication 'Microsoft.Web/sites@2024-04-01' = {
2   name: environment == 'production' ? 'wa-${projectFullName}' : 'wa-${projectFullName}-dev'
```

THE TEMPLATE – RESOURCES.BICEP



1. Define default parameters
2. Set log analytics workspace to existing shared workspace
3. Set app service plan (ASP) to existing shared compute resource
4. “Optional”: create storage account, blob service, and blob container a
5. Create application insights, targeting log analytics workspace
6. Create ASP, if not using the shared one
7. Create web application
 - Assign managed IDs to access KeyVaults
 - Set startup command for streamlit
 - Define environment variables to connect services and configure ap
8. Create basic blocking identity provider

Resource interdependence

- Some resources depend on the existence of others
- Creation order can be enforced using *dependsOn*

```
1 dependsOn: empty(existingASPName) ? [appServicePlan] : [existingAppServicePlan]
```

- Settings to allow resource access to other components are stored as application variables in the web app component

```
1 {
2   name: 'AZURE_STORAGE_ACCOUNT_NAME'
3   value: blobStorage ? storageAccount.name : ''
4 }
```



Web Application

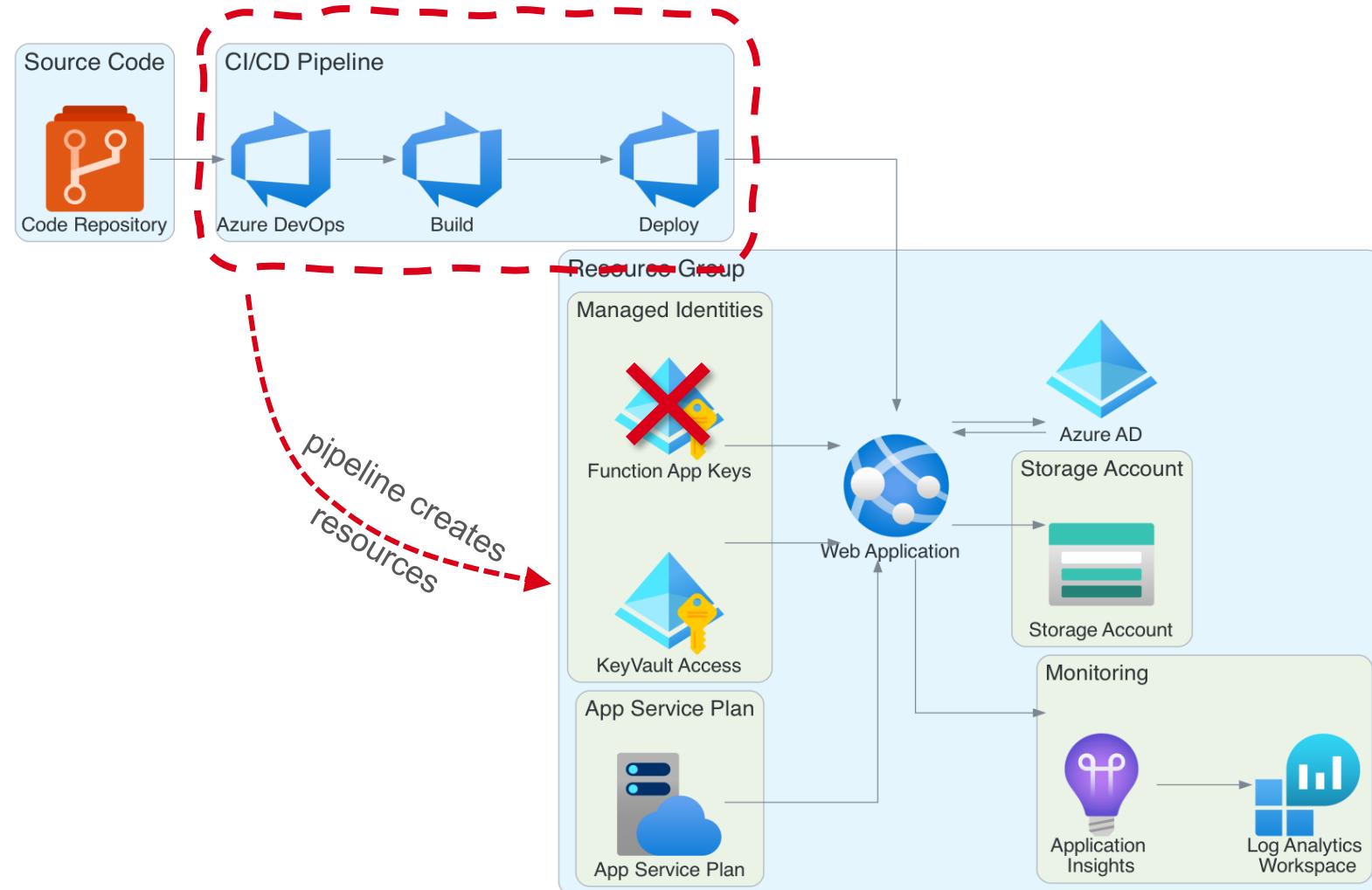
Streamlit configuration



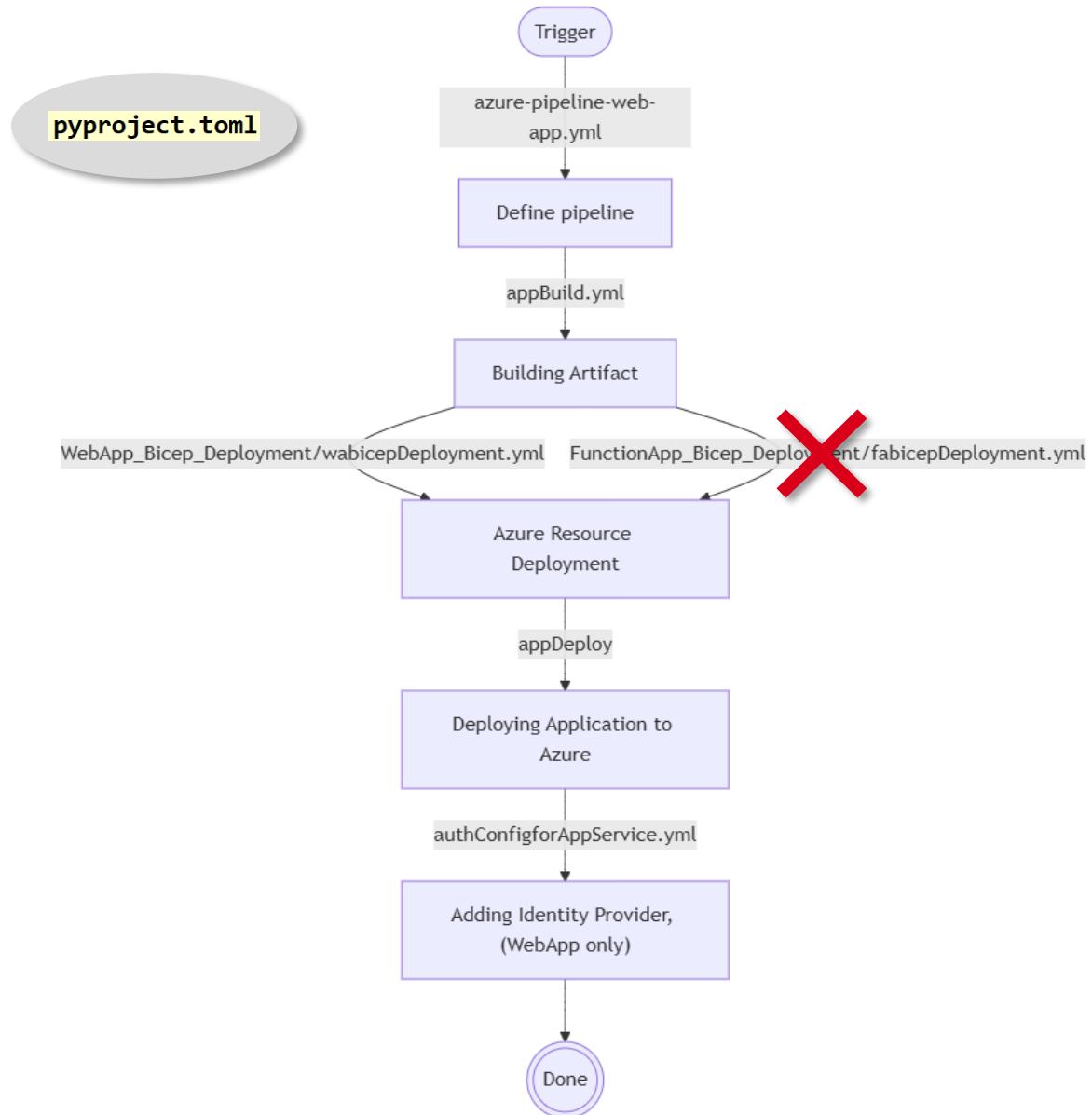
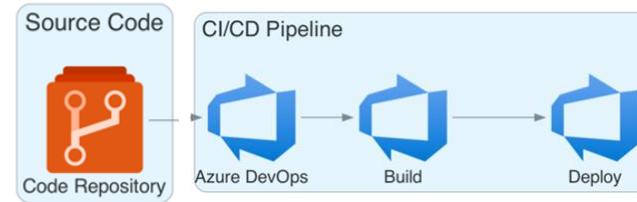
- Start-up command needs to be customized, as only Django & Flask support is integrated
- XSRF Protection needs to be disabled, otherwise Azure components from different servers cannot interact properly (authentication, storage)

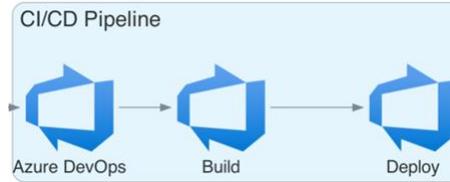
```
1 appCommandLine: 'python${pythonVersion} -m streamlit run /home/site/wwwroot/${pathToApp} --server.port 8000 --server.address 0.0.0.0 --server.enableXsrfProtection=false'
```

AZURE INFRASTRUCTURE



THE PIPELINE - STRUCTURE OF THE STAGES





THE PIPELINE – AZURE-PIPELINE-WEB-APP.YML

Declaration of pipeline stages

1. Resources:

Define repository for the centralized pipeline template files

2. Trigger:

Define default triggers for main and development branch

Optional: include / exclude paths, files, or file types

3. Variables:

Import Azure DevOps pipeline variable groups
Set constants and initiate variables

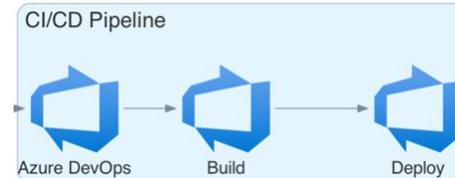
4. Stages:

Defines stages of the pipeline, which template pipeline file to use for it, and which parameters to pass to each stage

```
1 variables:
2   - name: isMain
3     value: $[eq(variables['Build.SourceBranchName'], 'main')]
```

→ Matching branch and target server

```
1 stages:
2   - template: appBuild.yml@Shared_Templates
```



THE PIPELINE – PARAMETERS

Minimizing manual configuration with file parsing and bicep

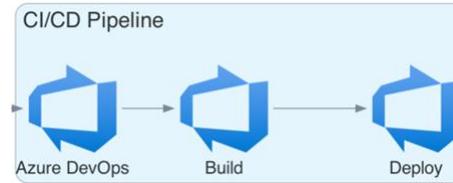
- Azure resources already defined in the bicep template
- But the pipeline also needs some information
- Solution:
 - Parse parameters from our one and only source of adaptable configuration: `pyproject.toml`
 - Modifications and error-handling trivial to implement
 - Set parameters as environment variables in the pipeline

ado_pipeline_repo\WebApp_Bicep_Deployment\deploy\nkd_pipeline_utils.py

```

1 def get_var_from_pyproject(var_name: str, path: Path | None = None) -> None:
2     if path is None:
3         path = Path.cwd().joinpath("pyproject.toml")
4     config = toml.load(path)
5     if var_name == "pythonVersion":
6         var_value = clean_version_string(config["project"]["requires-python"])
7     elif var_name == "nkd_utils_lib_version":
8         var_value = clean_version_string(config["tool"]["poetry"]["dependencies"]["nkd-utils-lib"]["version"])
9     else:
10        print(f"Error: Variable {var_name} not found in pyproject.toml file.")
11        var_value = None
12    set_pipeline_variable(var_name, var_value)

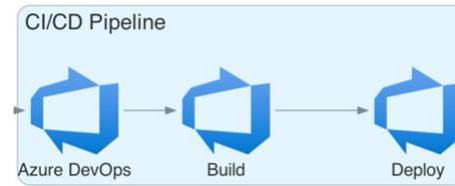
```



THE PIPELINE – APPBUILD.YML

Steps of build stage

1. Find `pyproject.toml` and set path as environment variable
 2. Extract project variables to set them globally
 3. Clean up pipeline template and structure app folder
 4. Authenticate to Azure DevOps artifact storage
 5. Install poetry
 6. Update lock file and export dependencies
 7. Install requirements
 8. Remove unneeded files
 9. Create zip archive from virtual environment and web app files
 10. Upload zip file from build to pipeline server
- Exporting to `requirements.txt` and pip installing from there to handle missing native support for poetry
 - Installation of requirements into target environment's *site-packages* folder
 - Deleting files that are not required to run the web app decreases deployed file size and avoids accidental publication

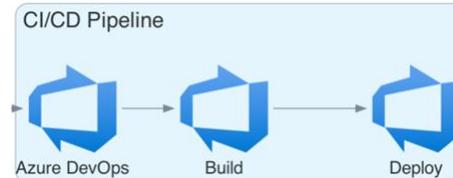


THE PIPELINE – WABICEPDEPLOYMENT.YML

Steps of Azure resource creation stage

1. Define variables for pipeline (most from params)
2. Read subscription ID and put it in variable
3. Download bicep template files
4. Set override parameters from app config
5. Deploy Azure Resources based on bicep template

- Define all parameters to receive from calling pipeline
- Starting the stage depends on success of previous build stage
- Resource definition happens in .bicep file
- Default settings for resources are defined in .bicepparam file
- Azure CLI can be used in pipeline bash scripts



THE PIPELINE – APPDEPLOY.YML

Steps of deployment stage

1. Define variables for pipeline
2. Deploy zip artifact with AzureWebApp task

- Define all parameters to receive from calling pipeline
- Starting the stage depends on success of all previous stages

```

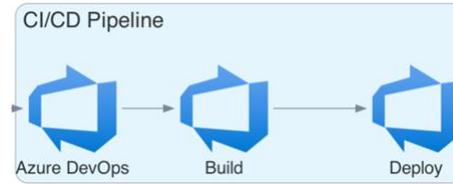
1 steps:
2 - task: AzureWebApp@1
3   displayName: 'Deploy Azure Web App'
4   inputs:
5     azureSubscription: ${{ variables.armconnection }}
6     appName: ${{ parameters.appName }}
7     appType: 'webAppLinux' # $(appType) it should be always webAppLinux
8     package: $(Pipeline.Workspace)/drop/${{Build.BuildId}}.zip

```

```

1 parameters:
2   - name: armconnection
3     type: string
4   - name: appName
5     type: string
6   - name: appType
7     type: string
8   - name: projectId
9     type: string
10  - name: projectName
11    type: string
12
13 stages:
14 - stage: DeployApp
15   displayName: 'Deploy to azure environment'
16   dependsOn:
17     - Build
18     - DeployBicep
19   variables:
20     armconnection: ${{ parameters.armconnection }}
21     appName: ${{ parameters.appName }}
22     appType: ${{ parameters.appType }}
23     isMain: ${{ variables.isMain }}
24     projectId: ${{ parameters.projectId }}
25     projectName: ${{ parameters.projectName }}

```

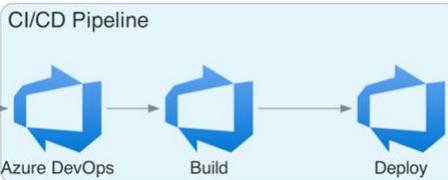


THE PIPELINE – AUTHCONFIGFORAPPSERVICE.YML

Steps of authorization configuration stage

1. Define variables for pipeline (from params)
2. Execute AzureCLI script on pipeline server
 1. Check if app registration exists in subscription
 2. If it does not yet exist, create app registration
 3. Add client secret from ADO
 4. Add authentication extension to Azure CLI
 5. Limit access to users from same tenant
 6. Enable and configure web app authentication
 7. Assign app registration to web app
 8. Configure app service to only accept connections from matching app registration
 9. If it does not yet exist, create Service Principal for app registration
 10. Set authorization by adding users / groups to service principal

- Define all parameters to receive from calling pipeline
- Starting the stage depends on success of all previous stages



THE PIPELINE – CREATION IN ADO

- Manual pipeline creation – required once
 - Pipelines → New pipeline → Azure Repos Git
 - Select repository with app code
 - Existing Azure Pipelines YAML
 - Select azure-pipeline-web-app.yml

Azure DevOps nkdit / Data Science / Pipelines

Where is your code?

- Azure Repos Git (YAML)
- Bitbucket Cloud (YAML)
- GitHub (YAML)
- GitHub Enterprise Server (YAML)

PAS_000_demo_app

✓ Connect ✓ Select **Configure** Review

New pipeline

Configure your pipeline

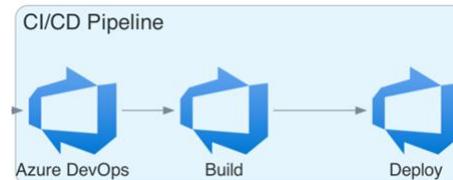
- Python package
- Python to Linux Web App on Azure
- Starter pipeline
- Existing Azure Pipelines YAML file

Select an existing YAML file

Branch: main

Path: /azure-pipeline-web-app.yml

PAS_000_demo_app ↗



THE PIPELINE – CREATION IN ADO

- After starting first pipeline run, pipeline stops... 😞
- Permission required to access centralized pipeline repository and Azure “environment”

Azure DevOps nkdit / Data Science / Pipelines / PAS_000_demo_app / 20250630.1

Data Science

- Overview
- Boards
- Repos
- Pipelines
- Pipelines

Jobs in run #20250630.1
PAS_000_demo_app

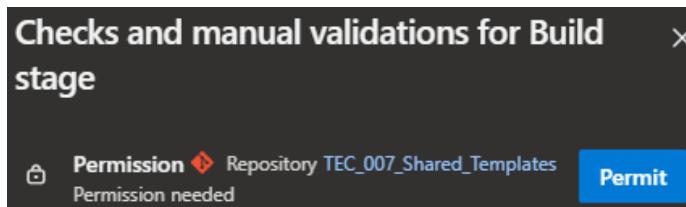
Build stage

- BuildJob
- Deploy Bicep Template
- DeployBicepJob

BuildJob

This pipeline needs permission to access a resource before this run can continue to Build stage

1 Job is pending...

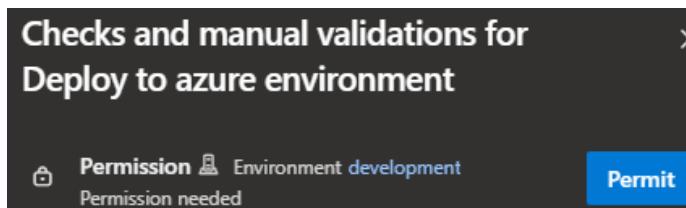


Stages Jobs

- Build stage: 1 job completed, 4m 38s, 1 artifact
- Deploy Bicep Template: 1 job completed, 2m 22s
- Deploy to azure environment: Waiting

This pipeline needs permission to access a resource before this run can continue to Deploy to azure environment

Permission needed



THE PIPELINE – DEBUGGING THE DEPLOYMENT

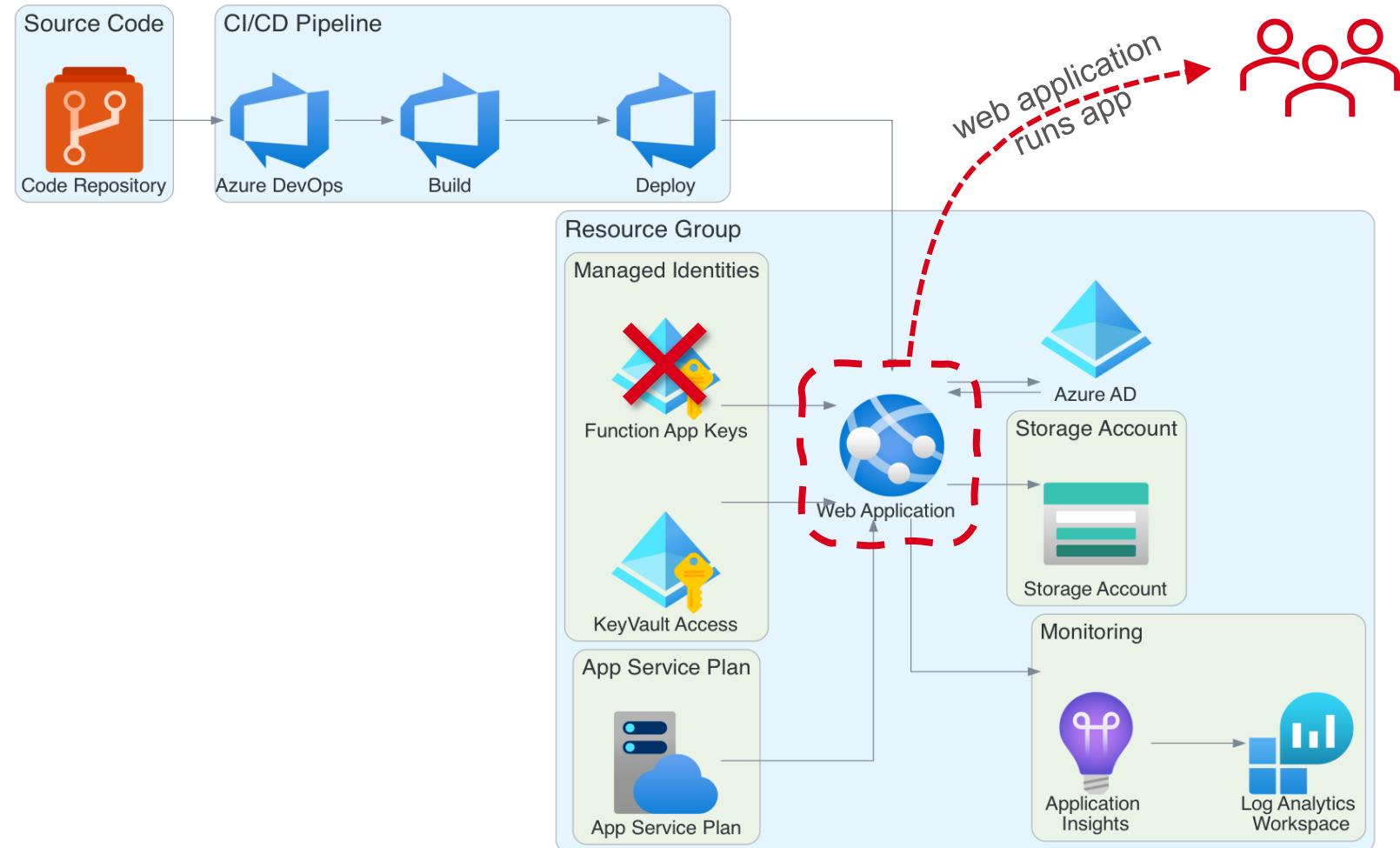
- Debugging deployment errors:

Azure Monitor – Activity Log

The screenshot shows the Microsoft Azure Monitor Activity Log interface. On the left, there's a sidebar with various monitoring options like Overview, Activity log, Alerts, Metrics, Logs, Change Analysis, Service health, Workbooks, and Dashboards with Grafana (preview). Below that is an Insights section listing Applications, Virtual Machines, Storage accounts, Containers, Networks, Azure Cosmos DB, Key Vaults, Azure Cache for Redis, and Azure Data Explorer Clusters. The main area is titled "Monitor | Activity log" and shows a list of activity logs. The logs are listed vertically with icons indicating their type (informational, warning, error). A red circle with a white exclamation mark is overlaid on the bottom right of the log list, drawing attention to the most recent entry.

Activity
Create Deployment
Create Deployment
Update insights component
Update Storage Account Create
⚠ 'audit' Policy action.
Update insights component
⚠ 'auditIfNotExists' Policy action.
Update Storage Account Create
Update Storage Account Create
Get deployment operation status
List Storage Account Keys
List Storage Account Keys
Put blob service properties
Put blob service properties
Put blob container
Get deployment operation status
Put blob container
Update website
⚠ Update website
Get deployment operation status

AZURE INFRASTRUCTURE



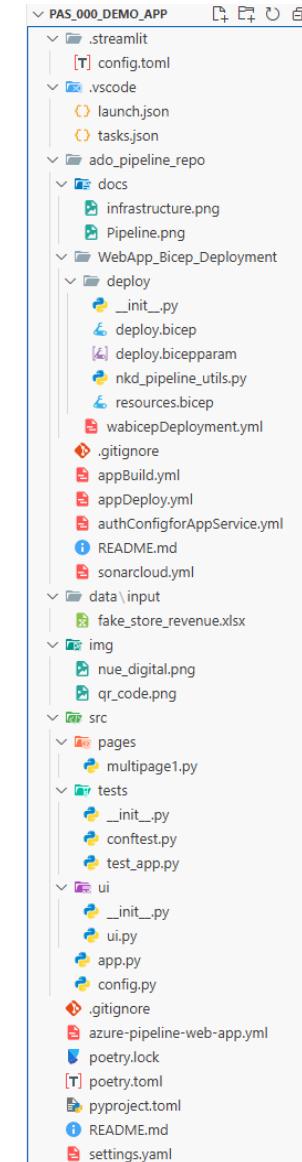
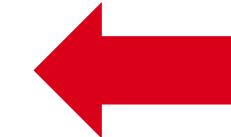
SHOW CASE

- Deploying an app live

... hopefully, we will see this in a few minutes!

```
✓ Deploy Azure Web App

1 Starting: Deploy Azure Web App
2 =====
3 Task      : Azure Web App
4 Description : Deploy an Azure Web App for Linux or Windows
5 Version    : 1.257.0
6 Author     : Microsoft Corporation
7 Help       : https://aka.ms/azurewebapptroubleshooting
8 =====
9 Got service connection details for Azure App Service:'wa-PAS-000-demo-app'
10 Package deployment using ZIP Deploy initiated.
11 Deploy logs can be viewed at https://wa-pas-000-demo-app.scm.azurewebsites.net/api/deployments/4ec51e29-7f6c-49b8-a394-d80cdd1f15dc/log
12 Successfully deployed web package to App Service.
13 Successfully updated App Service configuration details
14 Successfully added release annotation to the Application Insight : wa-PAS-000-demo-app-appinsights
15 Successfully updated deployment History at https://wa-pas-000-demo-app.scm.azurewebsites.net/api/deployments/231381751290632708
16 App Service Application URL: https://wa-pas-000-demo-app.azurewebsites.net
17 Finishing: Deploy Azure Web App
```



LESSONS LEARNED

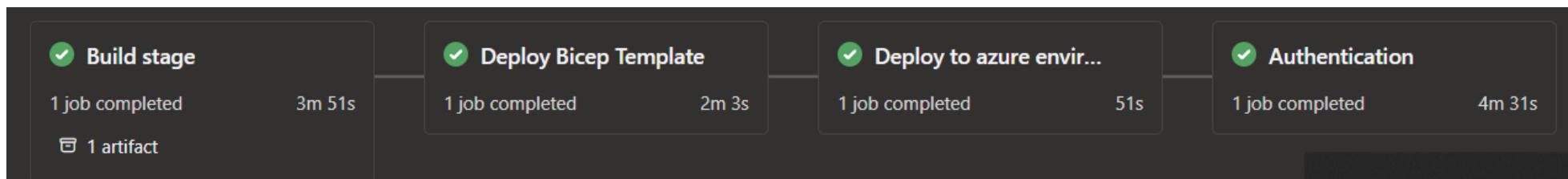
- Standardize early
- Avoid developing template and pipeline features in multiple projects
- Use as few configuration files as possible
- Use fewer tools, if you can achieve your task with a smaller selection
- Create template files, automations, and scripts
- Centralizing pipeline templates enables to perform changes in only one place
 - But: requires versioning of pipeline template!



LESSONS LEARNED

- Archive (zip) deployment is a must, otherwise deployment time explodes (5 vs. 45 minutes)
 - But: its read-only file system requires using external storage (e.g., blob storage account on Azure)
- Environment variables can only be used in the step *after* they were set
- Environment variables are lost between blocks, unless set with 'task.setvariable'
- There is more than one virtual environment on the web app server. Sometimes dependencies are accidentally overridden with a different package version by Azure components
- Use terminal commands for easy tasks
- Python scripts can be used as part of the pipeline to handle more complex tasks

```
python -c "from deploy import nkd_pipeline_utils; nkd_pipeline_utils.get_all_var_from_pyproject(path='${pyprojectPath}')"
```
- The reward for learning these lessons:



YOU WANT MORE?

- Setup script
 - Ask user questions about new project
 - Copy template app folder
 - Create local virtual environment (in .venv)
 - Create repository
 - Add pipeline to repository
- Automate pipeline creation
 - Authorization to use pipeline template repo in pipeline
 - Adding pipeline to repo
- Other ideas:
 - Enforce formatting and linting via pre-commit
 - Test scripts
 - Replace anaconda & poetry with uv
- Already implemented:
 - Deploy Azure Function Apps with same code base
 - Code quality and security checks with SonarCloud
 - Common development environment in VS Code
 - Linting
 - Formatting
 - Virtual environment with important dependencies
 - Install tools (Azure CLI, poetry, VS code extensions)
 - Authentication to Azure DevOps artifact feed

```
1  `` `bash
2  curl -u :<PAT> \
3  -X POST \
4  -H "Content-Type: application/json" \
5  -d '{
6    "folder": "\\",
7    "name": "MyPipeline",
8    "configuration": {
9      "type": "yaml",
10      "path": "azure-pipelines.yml"
11    }
12  }' \
13  https://dev.azure.com/{organization}/{project}/_apis/pipelines?api-version=6.0-preview.1
14  `` `
```



QUESTIONS?

COMMENTS?

SUGGESTIONS?

Code and slides @GitHub



🔗 [https://github.com/JSchoeck/
talk_webapp_template_and_pipeline_on_azure](https://github.com/JSchoeck/talk_webapp_template_and_pipeline_on_azure)



[Find me on
LinkedIn](#)