

Lab 3.1

Deploying the Basic App with Azure DevOps

Toepasbare Cloud Security voor IT-Professionals

Contents

1. Deploying the Basic App with Azure DevOps	3
1.1. Context	3
1.2. Learning goals	3
1.3. Estimated time	3
1.4. Before you start	4
2. Review Day 3 Infrastructure Inputs	5
2.1. Navigate to the Day 3 Terraform Folder	5
2.2. What the Pipeline Will Run	5
2.3. Create a Service Connection	5
2.3.1. Prerequisites for Service Connection	6
2.3.2. Create the Managed Identity	6
2.3.2.1. Fill in the form	6
2.3.3. Creating the service connection for Azure DevOps	6
2.3.3.1. Fill in the form	7
2.4. Verify Service Connection Permissions	9
2.5. Configure the Starter Pipeline for Your Prefix	9
2.6. Link Pipeline to Repository	10
2.7. Run the Starter Pipeline	11
3. Part 3: Review the Pipeline YAML Structure	14
3.1. Identify the Stages	14
3.2. What Is Missing From the Starter Pipeline?	14
3.3. Understanding Identity Separation	15
3.3.1. Pipeline Identity (Service Connection)	15
3.3.2. Application Identity (Managed Identity)	15
3.3.3. Compare the Two Identities	16
3.4. Trace the Artifact Promotion Flow	16
3.4.1. Build Stage Metadata	16
3.4.2. Artifact Publishing	17
3.4.3. Deployment Stages Download the Artifact	17
3.5. Reflection Questions	18
3.5.1. Question 1: Least Privilege for Pipeline Identity	18
3.5.2. Question 2: Approval Gates	18
3.5.3. Question 3: Time Drift Risk	18
3.5.4. Question 4: Traceability in Incident Response	19
4. Summary	20
5. Next Steps	21
6. Small Challenge	22

1. Deploying the Basic App with Azure DevOps

1.1. Context

Day 2 resources were deleted at the end of the previous day. Day 3 starts fresh: you deploy new infrastructure, create an Azure DevOps pipeline, and use it to deploy the basic backend app.

This lab introduces the Azure DevOps pipeline structure that will be extended throughout Day 3. You'll explore:

- Pipeline stages and jobs
- Service connections (pipeline identity) vs. managed identity (application identity)
- Basic build-and-deploy flow
- Build-once-promote-many artifact pattern (introduced here, hardened later)
- Traceability metadata (commit SHA, build ID)

The starter pipeline builds the app, publishes an artifact, deploys it to the Day 3 dev App Service, and runs a smoke test. Later labs add quality gates, security scans, IaC scanning, ZAP, promotion, and rollback.

1.2. Learning goals

By the end of this lab, you should be able to explain:

- The difference between pipeline identity (service connection) and application identity (managed identity).
- How Day 3 infrastructure is deployed and how Terraform outputs feed the pipeline.
- Why the pipeline has separate stages (Build, Deploy, and later Validate) instead of one combined script.
- How the build-once-promote-many pattern works and why it's more secure than rebuilding per environment.
- What metadata is embedded in the published artifact and why traceability matters.

1.3. Estimated time

90 minutes

Suggested timing:

Time	Activity
10 minutes	Azure DevOps project and repository setup
15 minutes	Create service connection (using pre-existing managed identity)
30 minutes	Run pipeline to deploy Day 3 infrastructure and capture outputs
10 minutes	Review pipeline YAML structure
15 minutes	Understand identity separation (service connection vs managed identity)
10 minutes	Trace artifact promotion flow

1.4. Before you start

Make sure you have:

- [] Azure DevOps organization created (or ready to create at <https://dev.azure.com>)
- [] Azure subscription with Contributor access
- [] Repository cloned locally with `azure-pipelines.yml` present
- [] Git installed and configured (run `git config --global user.name "Your Name"` and `git config --global user.email "your.email@example.com"`)

Note

Day 2 resources were deleted at the end of Day 2. In this lab, you deploy fresh Day 3 infrastructure using a pre-created `devops-example` resource group and managed identity. The pipeline will provision the infrastructure.

2. Review Day 3 Infrastructure Inputs

Day 3 starts from a clean Azure subscription. The pipeline will provision the Day 3 infrastructure with Terraform, then build and deploy the backend app.

2.1. Navigate to the Day 3 Terraform Folder

```
cd infrastructure/day-3 && code .
```

Open `dev.tfvars` and confirm it contains:

```
environment = "dev"
app_service_sku = "B1" # or another if B1 is not available in your region
enable_private_endpoints = false
enable_diagnostic_settings = false
```

Note

Dev keeps public access enabled so the pipeline, browser, and OWASP ZAP baseline scan can reach the API. Test/prod use stricter networking and are optional for this day.

Warning

Do not run `terraform apply` manually in this lab. The pipeline provisions the dev infrastructure so you can see infrastructure and application delivery in one Azure DevOps run.

2.2. What the Pipeline Will Run

The starter pipeline uses the Azure DevOps service connection to authenticate to Azure, then runs:

```
cd infrastructure/day-3
terraform init
terraform apply -auto-approve -var-file="dev.tfvars" -
var="student_prefix=<YOUR_PREFIX>"
```

The pipeline also creates a small Terraform state storage account before `terraform init`. This makes repeated pipeline runs safe: later labs update the same dev environment instead of trying to recreate every resource from scratch.

After Terraform completes, the pipeline captures these outputs and passes them to the deployment stage:

```
resource_group_name
public_backend_app_name
public_backend_url
```

2.3. Create a Service Connection

A service connection allows Azure DevOps pipelines to authenticate to your Azure subscription.

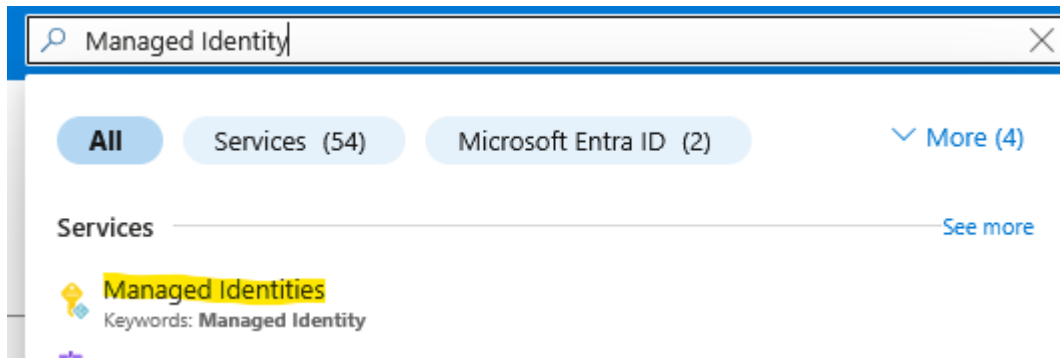
The pipeline uses a pre-created managed identity that has workload identity federation configured with Azure DevOps. This managed identity is scoped to the (pre-existing) `rg-devops-<student_prefix>` resource group.

2.3.1. Prerequisites for Service Connection

1. The rg-devops-<student_prefix> resource group should already exist in your Azure subscription (if not, create it)
2. This managed identity will have identity federation configured with Azure DevOps

2.3.2. Create the Managed Identity

1. Search for “Managed Identities” in the Azure Portal
2. Click “Create”



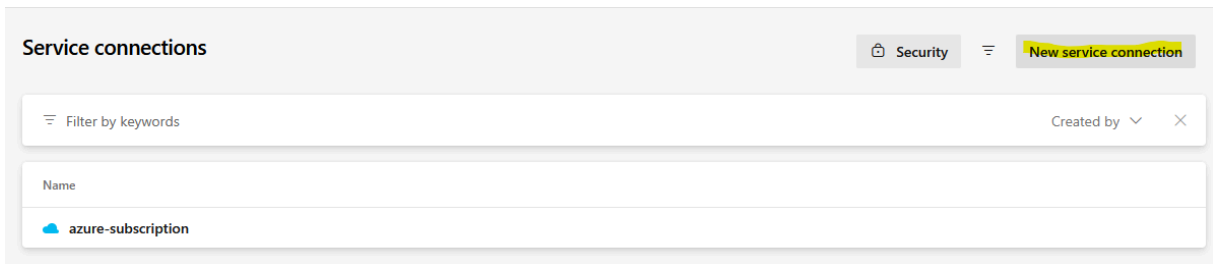
2.3.2.1. Fill in the form

1. Fill in the form:
 - Subscription: Your Azure subscription
 - Resource group: rg-devops-<student_prefix>
 - Name: rg-devops-<student_prefix>-mi
 - Region: Same as your resource group (e.g. West Europe)
 - Isolation Scope: None
1. Click “Review + Create”
2. Click “Create”

2.3.3. Creating the service connection for Azure DevOps

1. In Azure DevOps project, go to **Project Settings** (bottom-left corner)
2. Navigate to **Service connections** (under Pipelines section)
3. Click “Create service connection”
4. Select **Azure Resource Manager**
5. Click “Next”

2.3.3.1. Fill in the form



The screenshot shows a web interface for managing service connections. At the top, there's a header bar with the title 'Service connections' on the left. On the right side of the header, there's a 'Security' tab with a shield icon, followed by a hamburger menu icon, and a button labeled 'New service connection' which is highlighted with a yellow background. Below the header, there's a search bar with the placeholder text 'Filter by keywords' and a 'Created by' dropdown menu with a close button (X). The main content area is a table with a single row. The table has a header row with the label 'Name'. The data row contains a blue cloud icon followed by the text 'azure-subscription'.

Name
 azure-subscription

1. See steps on next page

New Azure service connection



Azure Resource Manager

Identity type

Managed identity



[Need help choosing a connection type?](#)

Step 1: Managed Identity details

Select the managed identity you want to use for workload identity federation.

Subscription for managed identity.

Visual Studio Professional Subscription (522f89cc-91af-4383-9...

Resource group for managed identity.

rg-devops-mmommers

Managed Identity

rg-devops-mmommers-mi

Step 2: Azure scope

Scope level for service connection.

- ☒ Subscription
- ☐ Management Group
- ☐ Machine Learning Workspace

Subscription for service connection.

Visual Studio Professional Subscription (522f89cc-91af-4383-9...

If you can't find subscription you need, please [create manually](#).

Resource group for Service connection (optional)

rg-devops-mmommers

Specify to limit access to the chosen resource group only.

Step 3: Service connection details

Service Connection Name

For security reasons, the service connection name can't be changed.

azure-subscription

Description (optional)

Security

- ☒ Grant access permission to all pipelines

[Learn more](#)
[Troubleshoot](#)

Back

Save

Security consideration

The pre-created managed identity with workload identity federation is a modern authentication approach that avoids long-lived credentials. In production, you would scope the managed identity to specific resources and use workload identity federation. For this lab, the managed identity is scoped to the `rg-devops-<student\_prefix>` resource group for simplicity, but in a real environment, you would follow least privilege principles and scope it more narrowly as seen in day-1.

2.4. Verify Service Connection Permissions

The service connection uses workload identity federation to authenticate to Azure. The managed identity has permissions scoped to the `rg-devops-<student_prefix>` resource group:

1. In Azure DevOps, open your service connection (click on it in the list)
2. The service connection is linked to the pre-created managed identity with workload identity federation
3. Navigate to **Resource Groups** → your resource group → **Access control (IAM)** → **Role assignments**
4. Search for the managed identity name

What role does the service principal have?

Your answer (should be “Contributor” by default):

Question

Is Contributor the least-privilege role needed for deploying App Services and updating configurations? What would be a more restrictive alternative?

Make sure to add another role if not present: `User Access Administrator` on the subscription. The pipeline needs this to assign permissions to the managed identity during deployment.

Your answer:

2.5. Configure the Starter Pipeline for Your Prefix

Open `azure-pipelines.yml` in the repository root. At the top of the file, update these variables:

```
variables:  
  azureLocation: 'westeurope'  
  azureSubscription: 'azure-subscription'  
  studentPrefix: '<YOUR_PREFIX>'
```

Example:

```
variables:
  azureLocation: 'westeurope'
  azureSubscription: 'azure-subscription'
  studentPrefix: 'jdoe'
```

Warning

Use the same prefix you chose earlier. The pipeline uses this prefix when it provisions Day 3 infrastructure.

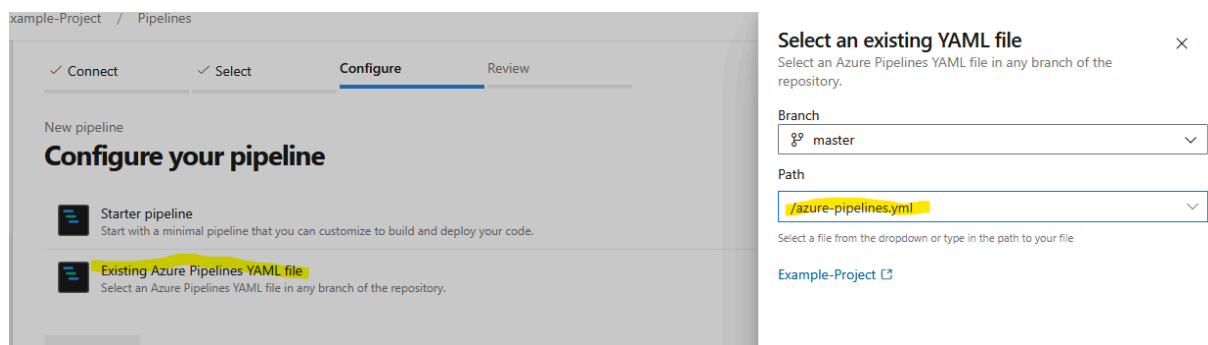
Commit and push the starter pipeline update:

```
git add azure-pipelines.yml
git commit -m "feat: configure day 3 starter pipeline"
git push azure main
```

2.6. Link Pipeline to Repository

Now that the repository and service connection are ready, create the pipeline:

1. In Azure DevOps project, go to **Pipelines** → **Pipelines**
2. Click “Create pipeline”
3. Where is your code? → **Azure Repos Git**
4. Select your repository (cloudsec-course)



2.7. Run the Starter Pipeline

In Lab 3.1, you run the pipeline once to provision infrastructure and deploy the basic backend app. Later labs will add quality gates, security scanning, IaC scanning, ZAP, and test promotion.

1. Click “Run”
2. Watch the pipeline execute the ProvisionDevInfra stage
3. Watch the pipeline execute the Build stage
4. Watch the pipeline execute the DeployDev stage
5. Open the Terraform logs and record the resource group, App Service name, and backend URL
6. Open the smoke test logs and confirm /api/health returns HTTP 200

If requested you have to grant access to the environment the first time you run the pipeline. This is a security feature in Azure DevOps to prevent unauthorized deployments.

The screenshot displays the Azure DevOps pipeline interface for a run titled "#20260603.1 • Starting from scratch". The interface is divided into several sections:

- Summary:** Shows the pipeline was manually run by Mitch Mommers. It lists the repository as "Example-Project" on the "master" branch with commit "51c2fe54". The run started at 8:50 PM and has elapsed 9m 5s. It shows 0 work items and 2 published items.
- Warnings:** Two warnings are listed: "Task 'Node.js tool installer' version 0 (NodeTool@0) is deprecated." and "This task is deprecated and will no longer receive updates. Please use UseNodeV1 as a replacement."
- Permission:** A yellow banner states: "This pipeline needs permission to access a resource before this run can continue to Deploy Basic App to Dev".
- Stages:** The pipeline consists of three stages:
 - Provision Day 3 Dev I...:** 1 job completed, 7m 41s, 1 artifact.
 - Build Basic App:** 1 job completed, 20s, 1 artifact.
 - Deploy Basic App to ...:** Currently in a "Waiting" state with a "Permission needed" message.

If all goes well you should see:

The screenshot displays an Azure DevOps pipeline run for 'Example-Project' with the title '#20260603.2 • Fix in the app build (missing node_modules)'. A green checkmark icon indicates a successful run. A message states: 'This run is being retained as one of 3 recent runs by master (Branch)'. The 'Summary' tab is active, showing the run was 'Manually run by Mitch Mommers'. Details include: Repository and version (Example-Project, master, 4a9caa78), Time started and elapsed (Today at 9:08 PM, 5m 23s), and Related items (0 work items, 2 published). The 'Warnings' section shows two messages about deprecated tasks. The 'Stages' section shows three stages: 'Provision Day 3 Dev I...', 'Build Basic App', and 'Deploy Basic App to ...', each with 1 job completed and 1 artifact.

✓ #20260603.2 • Fix in the app build (missing node_modules)
Example-Project

ⓘ This run is being retained as one of 3 recent runs by master (Branch).

Summary Environments Code Coverage

Manually run by Mitch Mommers

Repository and version	Time started and elapsed	Related
Example-Project	Today at 9:08 PM	0 work items
master 4a9caa78	5m 23s	2 published

Warnings 2

- Task 'Node.js tool installer' version 0 (NodeTool@0) is deprecated.
Build Basic App • Build and Package App • Initialize job
- This task is deprecated and will no longer receive updates. Please use UseNodeV1 as a replacement.
Build Basic App • Build and Package App • Initialize job

Stages Jobs

✓ Provision Day 3 Dev I... 1 job completed 3m 5s 1 artifact	✓ Build Basic App 1 job completed 27s 1 artifact	✓ Deploy Basic App to ... 1 job completed 1m 33s
---	--	---

Open the deployed backend URL from Terraform output:

```
curl <PUBLIC_BACKEND_URL>/api/health
```

What response do you get?

Your answer:

You now have a working Day 3 baseline: fresh infrastructure, deployed code, and a pipeline that can be extended safely.

What is the pipeline name shown in Azure DevOps?

Your answer:

3. Part 3: Review the Pipeline YAML Structure

Open `azure-pipelines.yml` in the repository root. For the lab work, use `azure-pipelines.yml`; `azure-pipelines.yml` is the full reference pipeline.

3.1. Identify the Stages

The starter pipeline has 3 stages. List them:

Stage Name	Purpose (in your own words)
1.	
2.	
3.	

3.2. What Is Missing From the Starter Pipeline?

The starter pipeline deploys the app, but it does not yet prove that the app is typed, tested, scanned, or safely promoted. Fill in the table with what later labs will add:

Future Addition	Lab	Why it matters
Quality gates	Lab 3.2	
SAST/ dependency/ SBOM	Lab 3.3	
IaC scanning	Lab 3.4	
ZAP baseline	Lab 3.5	
Test promotion/ rollback	Lab 3.6	

Question

Why do quality gates run in a separate job from security scans? Why not combine them into one job?

Your answer:

3.3. Understanding Identity Separation

3.3.1. Pipeline Identity (Service Connection)

A service connection is an Azure AD service principal that the pipeline uses to deploy resources, publish artifacts, and access Azure APIs.

Navigate to Azure DevOps → Project Settings → Service connections.

Do you see a service connection for your Azure subscription?

Your answer (Yes/No and name):

What permissions does this service connection have? (Check Azure Portal → Resource Groups → Access control (IAM) → Role assignments, filter by the service principal name)

Your answer:

> Make sure the Identity has both Contributor and User Access Administrator roles on the subscription. The pipeline needs User Access Administrator to assign permissions to the managed identity during deployment.

3.3.2. Application Identity (Managed Identity)

The backend application (running in Azure App Service) uses a managed identity to access Key Vault and other Azure resources at runtime.

Navigate to Azure Portal → App Service (app-...-api) → Identity.

Is system-assigned managed identity enabled?

Your answer (Yes/No):

What permissions does this managed identity have? (Check Key Vault → Access control (IAM) or Access policies)

Your answer:

3.3.3. Compare the Two Identities

Fill in the comparison table:

Aspect	Service Connection (Pipeline Identity)	Managed Identity (App Identity)
When it's used		
What it accesses		
Scope of permissions		
Risk if compromised		

Question

Why should the pipeline identity and application identity be different? What would happen if the backend application used the same identity as the pipeline?

Your answer:

3.4. Trace the Artifact Promotion Flow

3.4.1. Build Stage Metadata

In the Build stage, the pipeline creates metadata files before publishing the artifact.

Find the step that creates metadata in `azure-pipelines.yml` (look for echo commands writing to `BUILD_VERSION`, `BUILD_COMMIT`, `BUILD_NUMBER` files).

What information is embedded in the artifact?

Metadata File	What it contains
<code>BUILD_VERSION</code>	
<code>BUILD_COMMIT</code>	
<code>BUILD_NUMBER</code>	

Question

Why is this metadata important? How does it help with traceability?

Your answer:

3.4.2. Artifact Publishing

Find the PublishPipelineArtifact@1 task in the Build stage.

What is the artifact name?

Your answer:

What directory is being published?

Your answer:

3.4.3. Deployment Stages Download the Artifact

Find the artifact download step in the DeployDev stage.

Does this stage rebuild the application (look for `npm run build` or similar)?

Your answer (Yes/No):

What artifact does it download?

Your answer:

Question

Why is it important that deployment stages download the artifact instead of rebuilding? What could go wrong if each environment rebuilt the application?

3.5. Reflection Questions

3.5.1. Question 1: Least Privilege for Pipeline Identity

The service connection currently has Contributor role on the entire subscription.

Is this least privilege? What would be a better scope?

Your answer:

3.5.2. Question 2: Approval Gates

The DeployTest stage includes environment: 'test' which requires manual approval in Azure DevOps.

Why would you require manual approval for test but not for dev?

Your answer:

3.5.3. Question 3: Time Drift Risk

Imagine this scenario:

- Monday: Pipeline builds artifact from commit abc123, tests pass, deploys to dev
- Tuesday: A new version of express is published to npm with a critical CVE
- Friday: You manually run `npm install` & `npm run build` in production

Does production deploy the same code that was tested on Monday?

Your answer:

3.5.4. Question 4: Traceability in Incident Response

A security incident is detected in production. You need to answer: “Which commit is currently deployed?”

How do you find this information using the artifact metadata?

Your answer:

4. Summary

In this lab, you:

- Created Azure DevOps organization and project
- Imported repository to Azure DevOps
- Created service connection for pipeline authentication
- Linked pipeline to azure-pipelines.yml
- Ran the starter pipeline to provision dev infrastructure, build, and deploy the basic app
- Reviewed the Azure DevOps pipeline structure (ProvisionDevInfra, Build, and DeployDev stages; Validate is added next)
- Distinguished pipeline identity (service connection) from application identity (managed identity)
- Traced the build-once-promote-many artifact flow
- Understood why metadata (commit SHA, build ID) enables traceability

Note

In the next lab (Lab 3.2), you will add quality gates (type checking and automated tests) before the Build stage, then run the pipeline again.

5. Next Steps

- ☐ Understand the difference between service connection and managed identity
- ☐ Know what metadata is embedded in the artifact
- ☐ Explain why deployment stages download artifacts instead of rebuilding
- ☐ Review the reflection questions with your team

Continue to **Lab 3.2: Adding Quality Gates**.

6. Small Challenge

- You might have noticed that the pipeline has warnings regarding the Task version. Can you update the pipeline to use the latest versions of the tasks? (Hint: check the Azure DevOps documentation for the latest task versions, and check the warnings)
 - <https://learn.microsoft.com/en-us/azure/devops/pipelines/tasks/reference/use-node-v1?view=azure-pipelines>