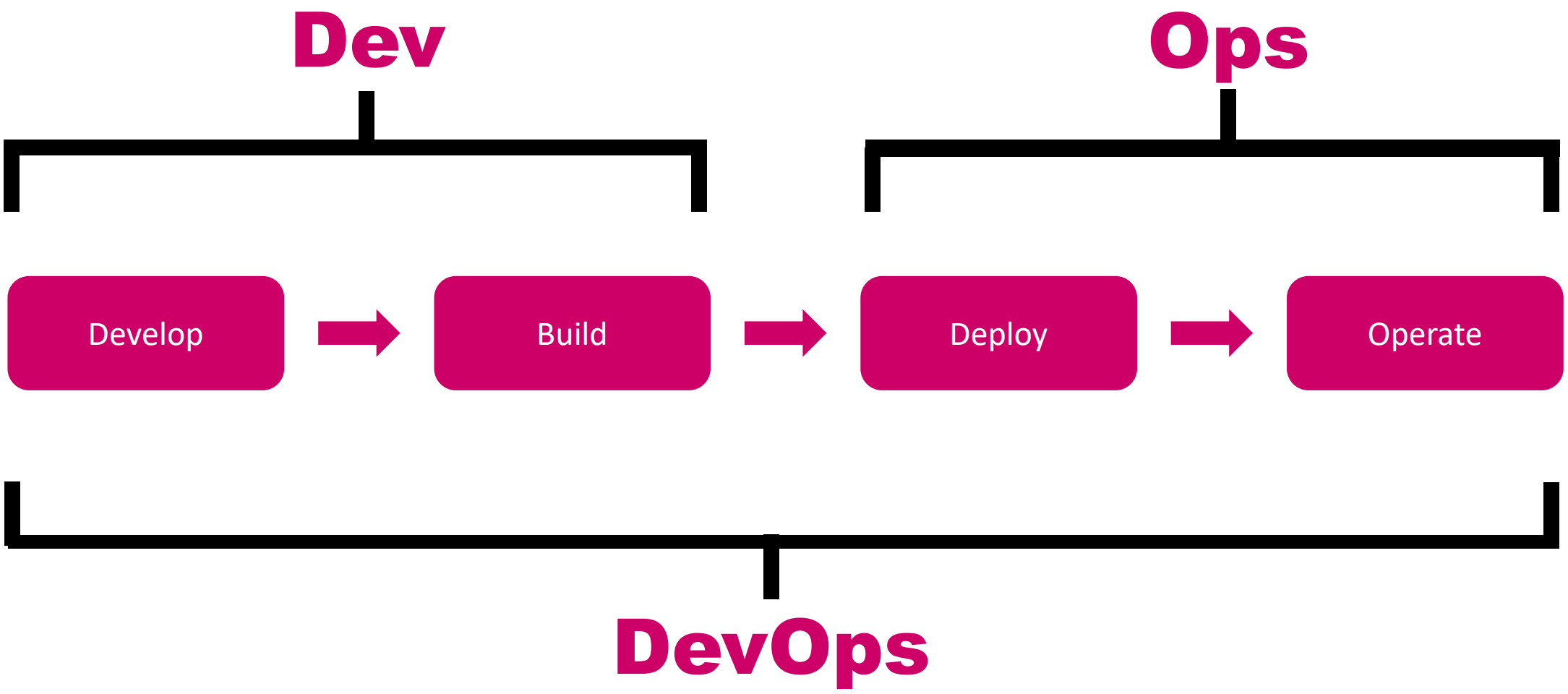
SECURE DEVELOPMENT
KEEPING YOUR SECRETS PRIVATE

Henry Been

So…

WHO DOES DEVOPS?

THE NEED FOR SECRET MANAGEMENT



Access and secret management goals

No shared credentials

Principle of
least privilege

No team member needs
production access

Decouple authentication
from authorization

So…

LET'S GO ON A JOURNEY

Approach 1

Approach 2

Approach 3

Approach 4

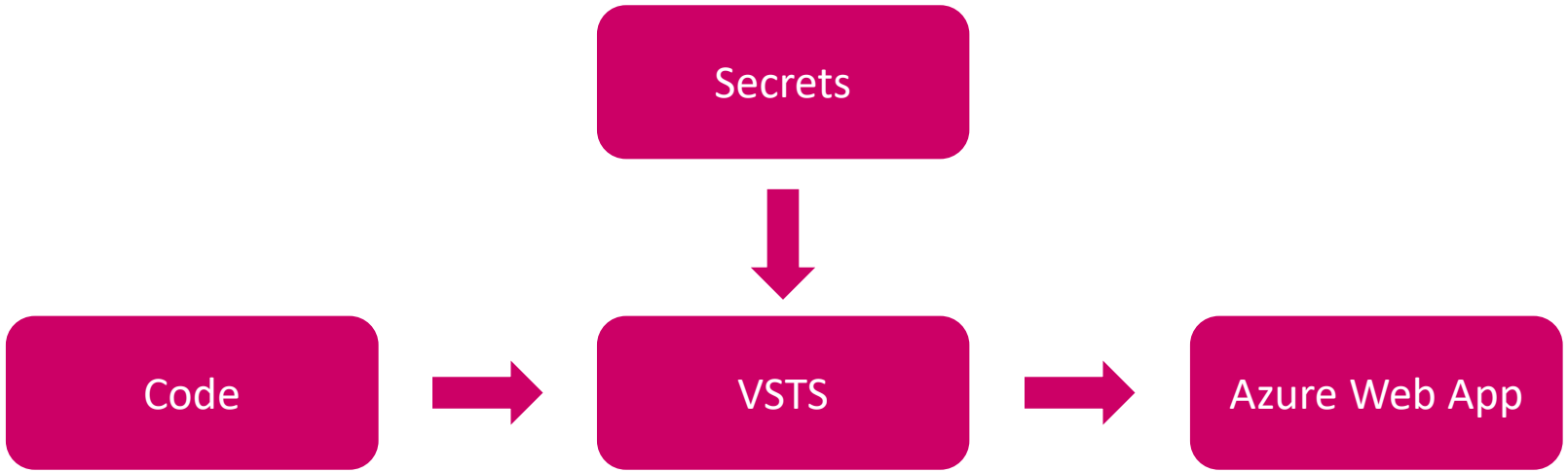
WONDERING **WHO** IS THAT GUY?

HENRY BEEN
Independent Devops & Azure Architect

E: consultancy@henrybeen.nl
T: @henry_been
L: linkedin.com/in/henrybeen
W: henrybeen.nl

Approach 1

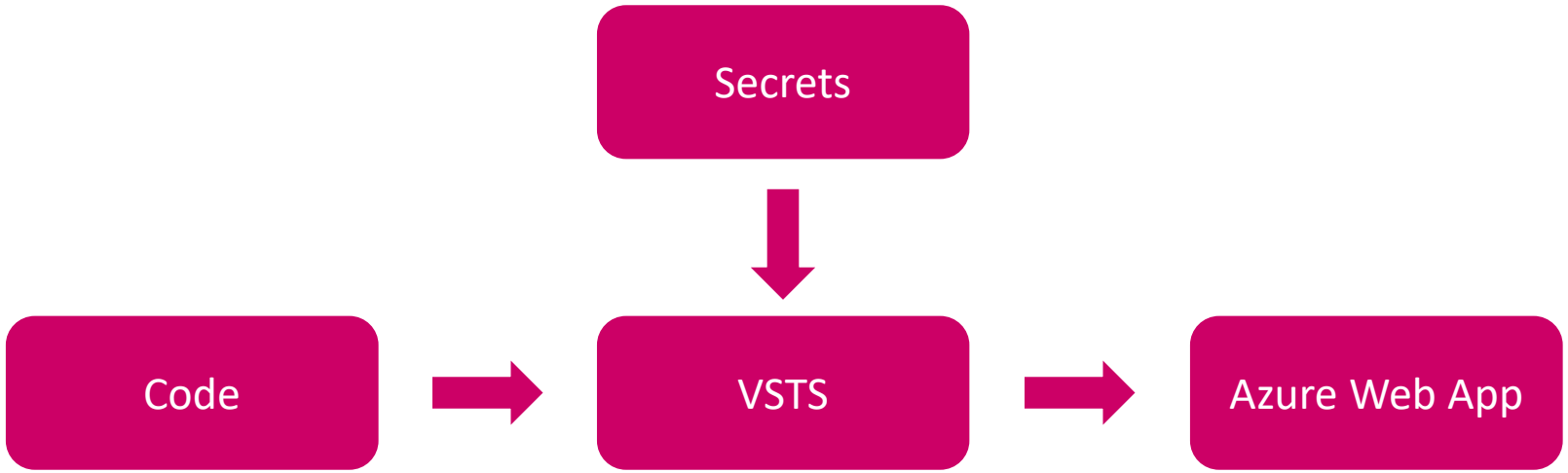
USING RELEASE ORCHESTRATOR



USING RELEASE ORCHESTRATOR

The approach:

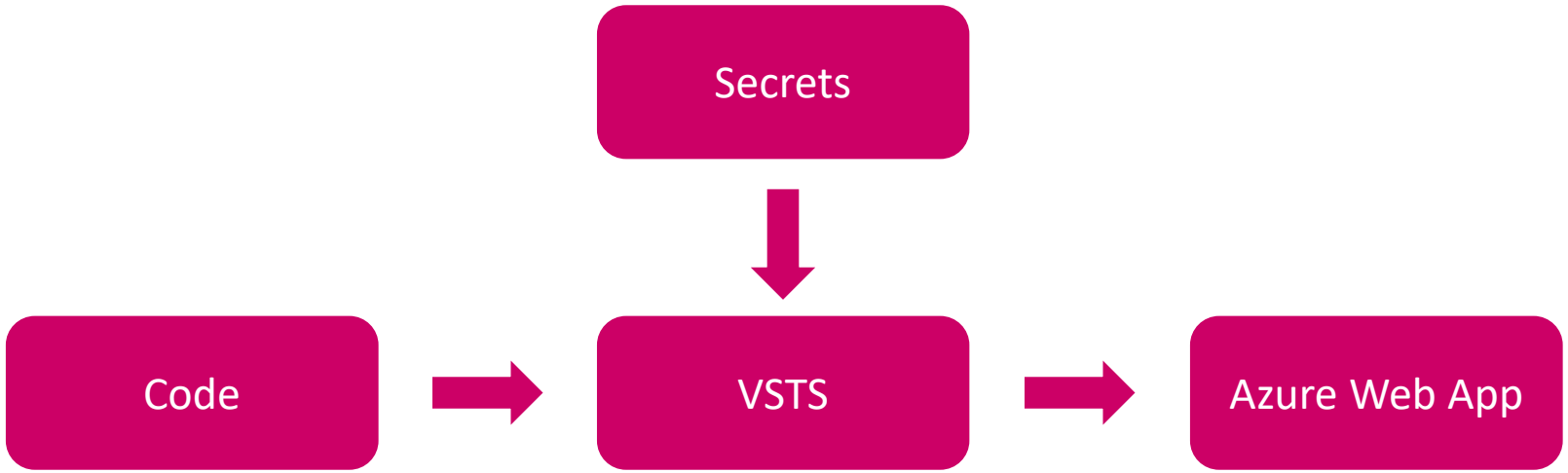
- Save secrets into release orchestrator
- Inject secrets into configuration at release time



USING RELEASE ORCHESTRATOR

Scenario:

- As a team you do CD of your code
- You do not do CD of your infrastructure



DEMO **TIME!**
USING RELEASE ORCHESTRATOR

USING RELEASE ORCHESTRATOR

Pros

- Secrets are pretty secure
- Easy to start with
- Fits existing situations

Cons

- You see and copy secrets
- Secrets visible in portal
- Duplication of secrets

- Cannot roll secrets easily

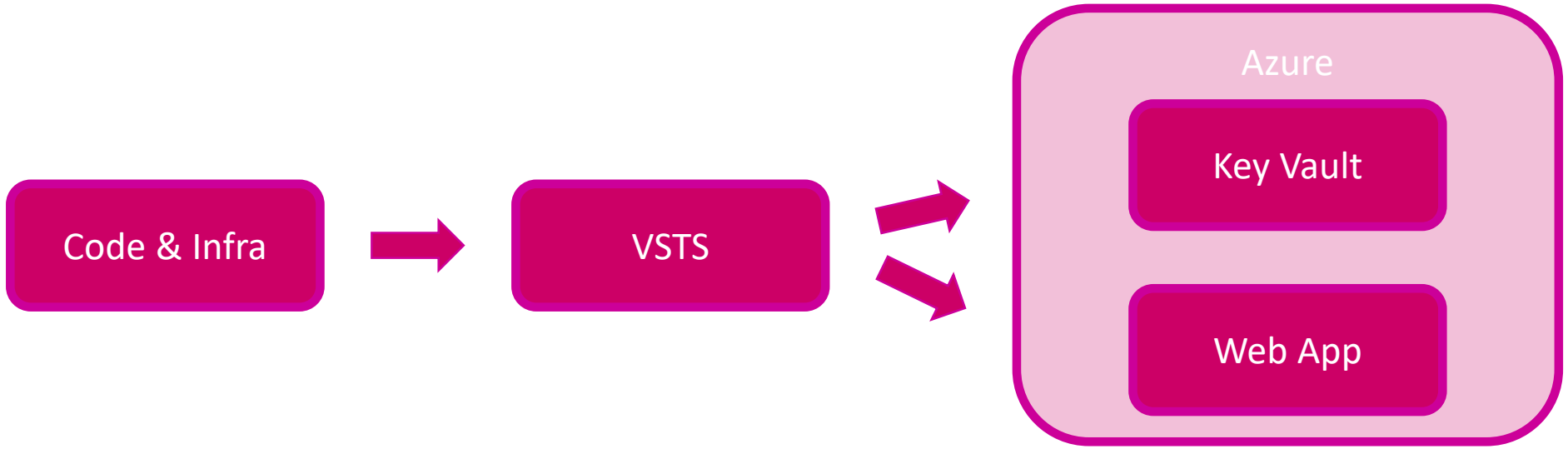
Intermezzo: Roll a secret

Prerequisite: Have primary & secondary secrets

Approach

1. Change the secret in release orchestrator to secondary secret
2. Release
3. Roll primary secret
4. Change the secret in release orchestrator to primary secret
5. Release
6. Roll secondary secret

Approach 2
USING ARM TEMPLATES



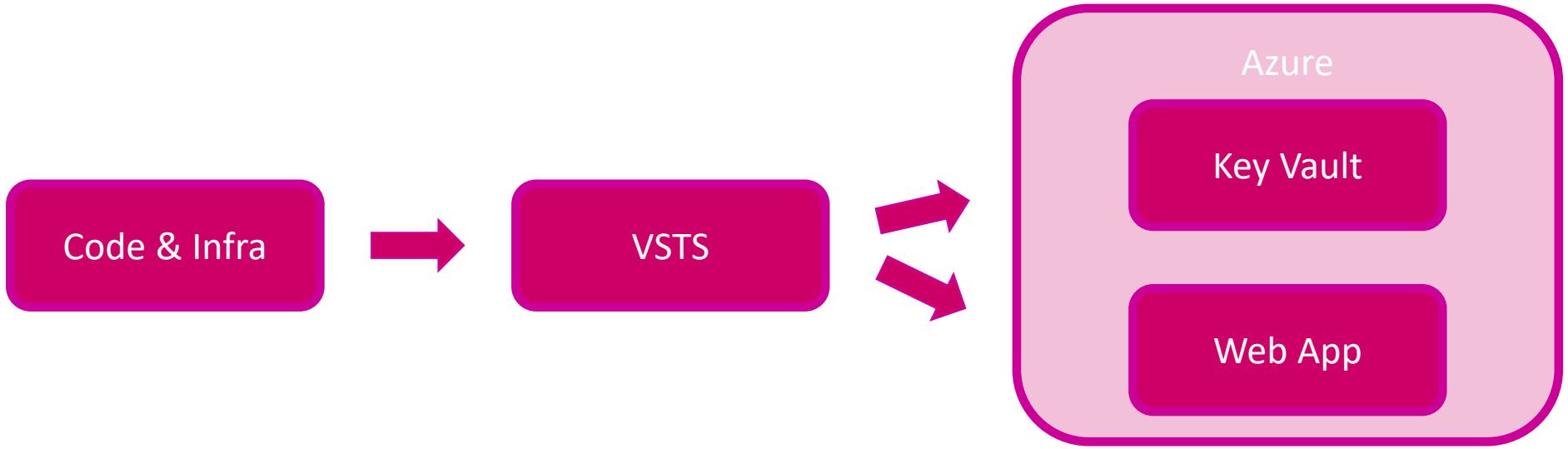
USING ARM TEMPLATES

The approach:

Use ARM Templates

A. Echo secrets directly from resource into Web App

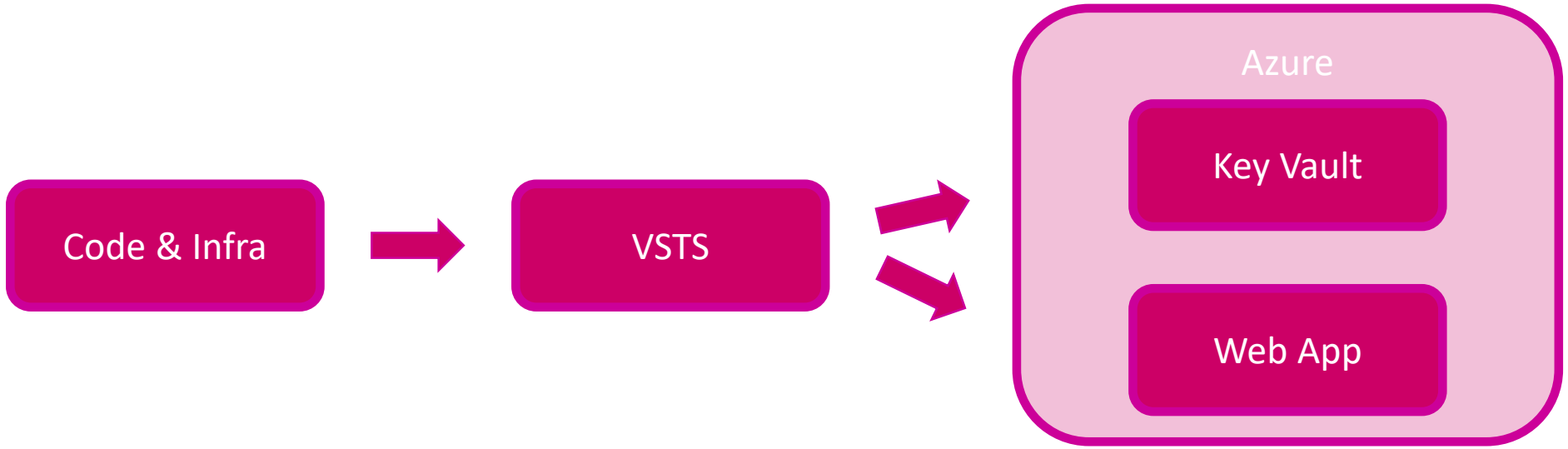
B. Echo secrets from Key Vault into Web App



USING ARM TEMPLATES

Scenario:

- As a team you do CD of your code
- You also do CD of your infrastructure



DEMO **TIME!**
USING ARM TEMPLATES

USING ARM TEMPLATES

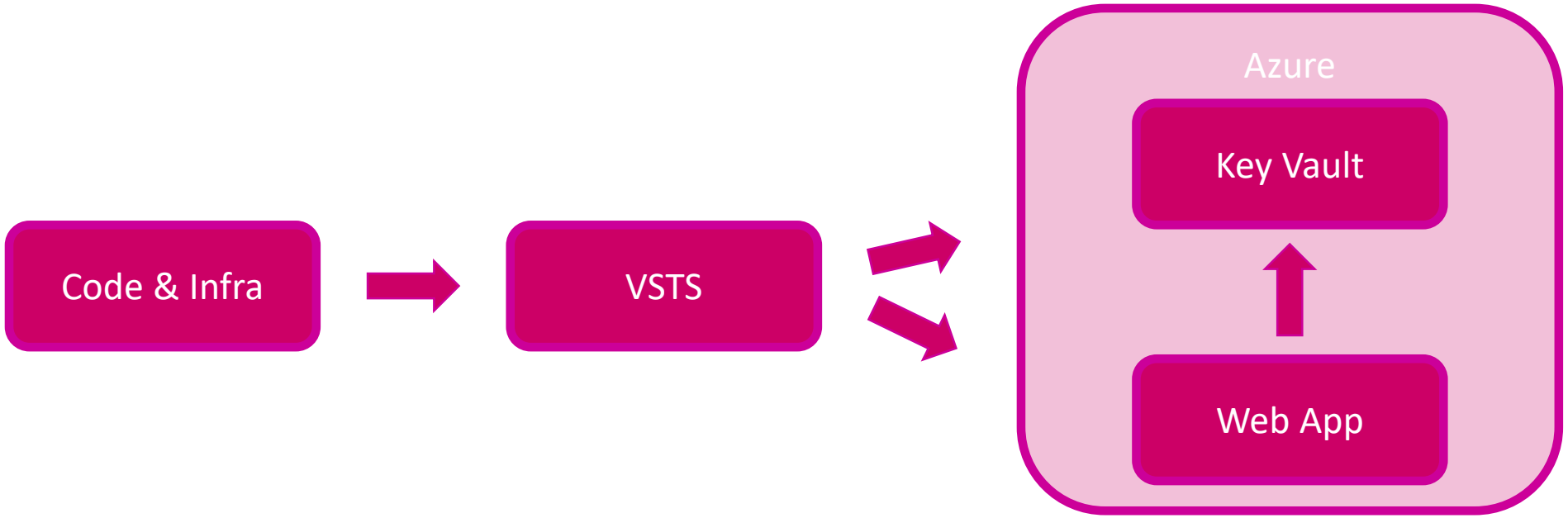
Pros

- No manual copying or sharing of secrets
- No more manual duplication of Azure keys

Cons

- Secrets visible in portal
- Still cannot roll secrets easily

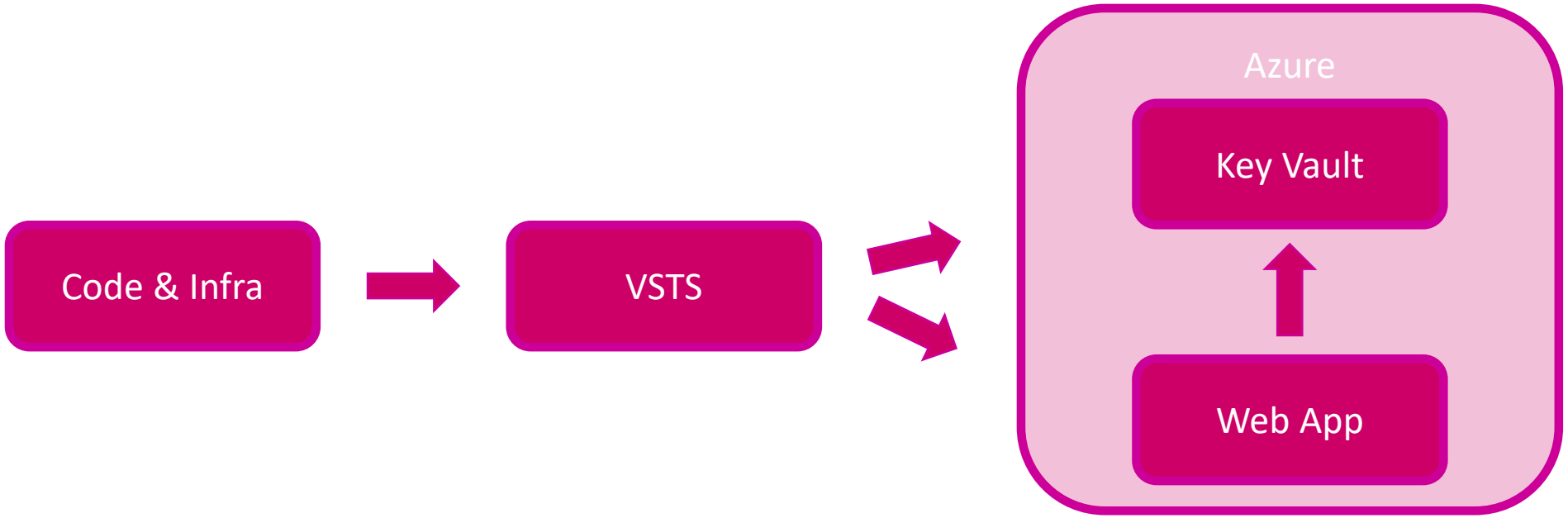
Approach 3
DIRECTLY FROM KEY VAULT



DIRECTLY ACCESS KEY VAULT

The approach:

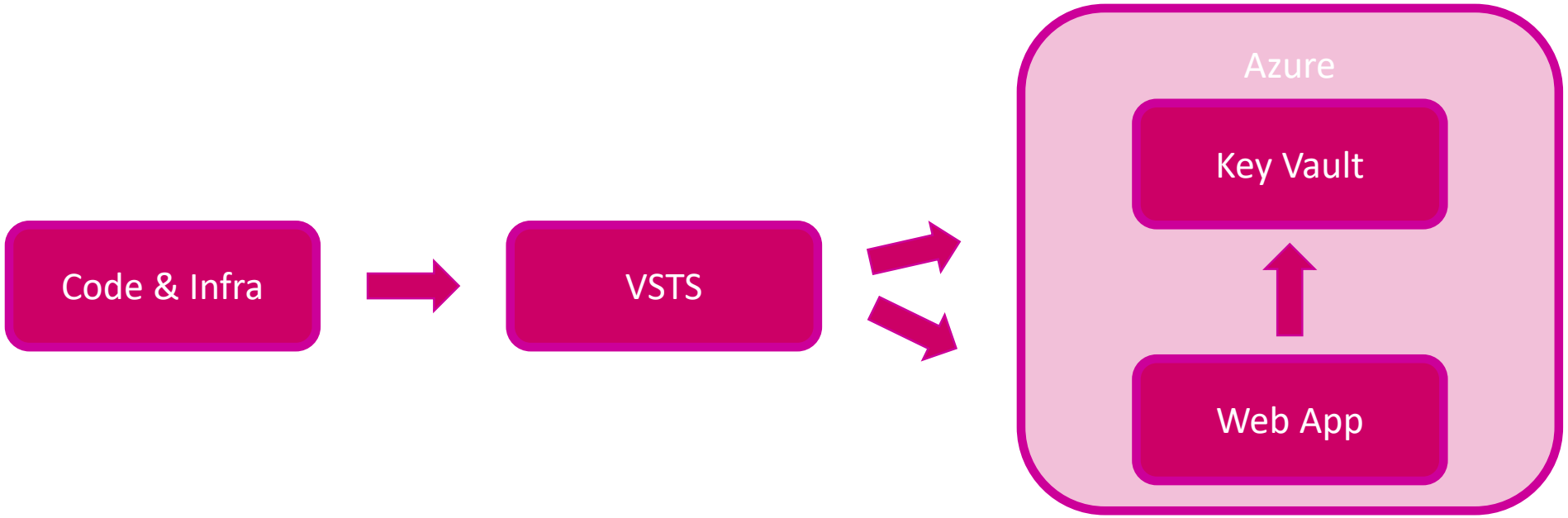
1. Give your application a (runtime) AAD identity
2. Grant your application access to KeyVault



DIRECTLY ACCESS KEY VAULT

Scenario:

1. You do CD of both infrastructure and code
2. You are allowed to provision to the AD and use non GA stuff



HOWTO: Local Development

1. Grant your developer account access to (another) Key Vault
   - Best alternative
   - Requires your machine to be in the same AD domain

2. Only use Key Vault in Azure (locally use Web.config)
   - You have to write code to do this (though pretty straightforward)

3. Manually create a development identity and use that
   - However… do not check secrets into source control


DEMO TIME!
DIRECTLY ACCESS KEY VAULT
SPEED:

DIRECTLY ACCESS KEY VAULT

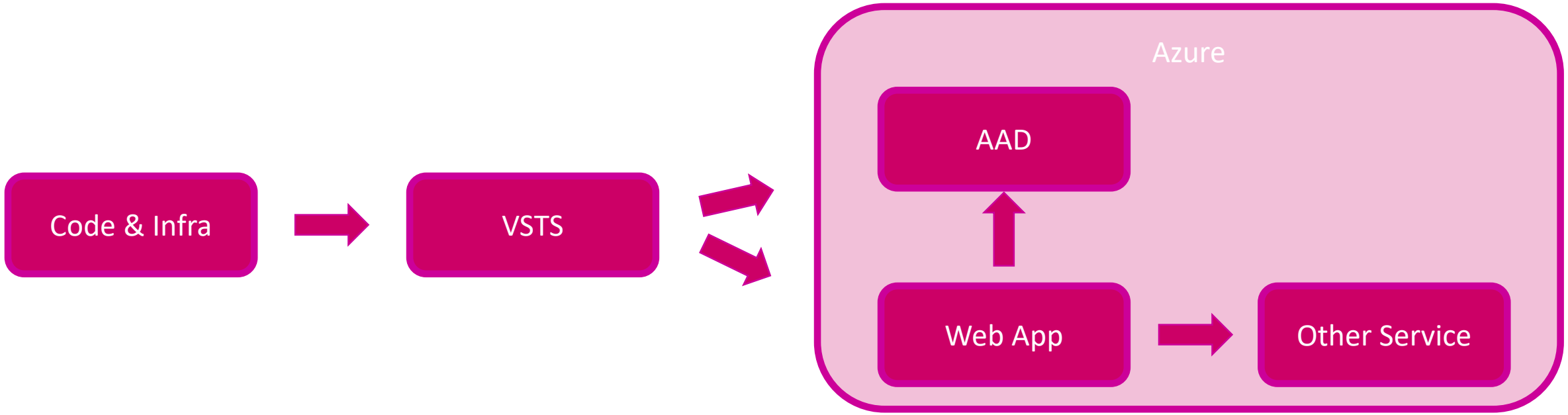
Pros

- No manual copying or sharing of secrets
- No more duplication of Azure keys
- Secrets no longer visible in portal
- Changed secrets are automatically picked up

Cons

- Not GA yet
- Only available on Azure Web Apps

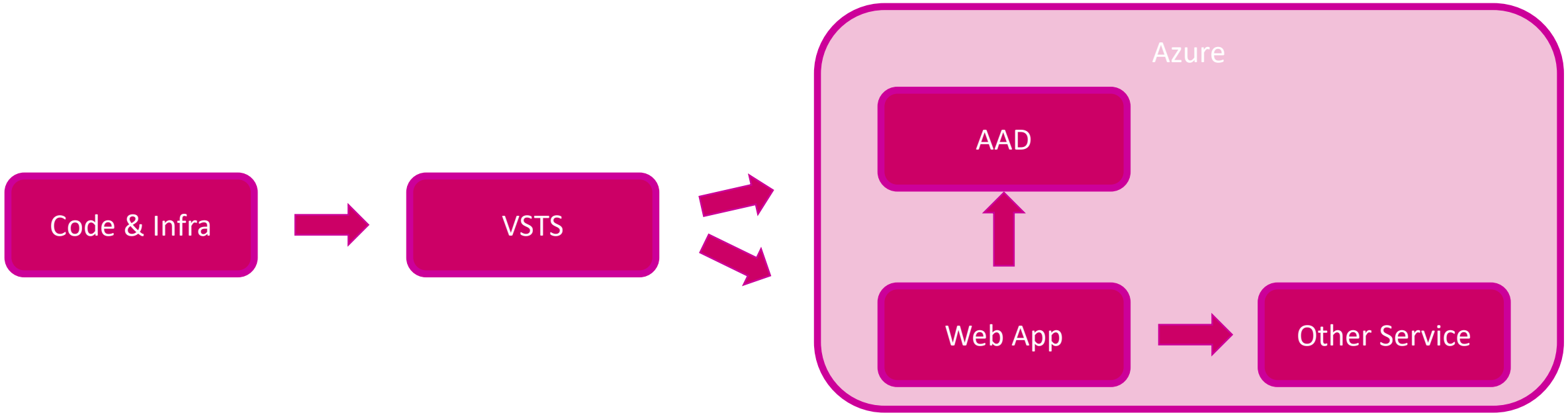
Approach 4
DIRECTLY ACCESS SERVICE



DIRECTLY ACCESS SERVICE

The approach:

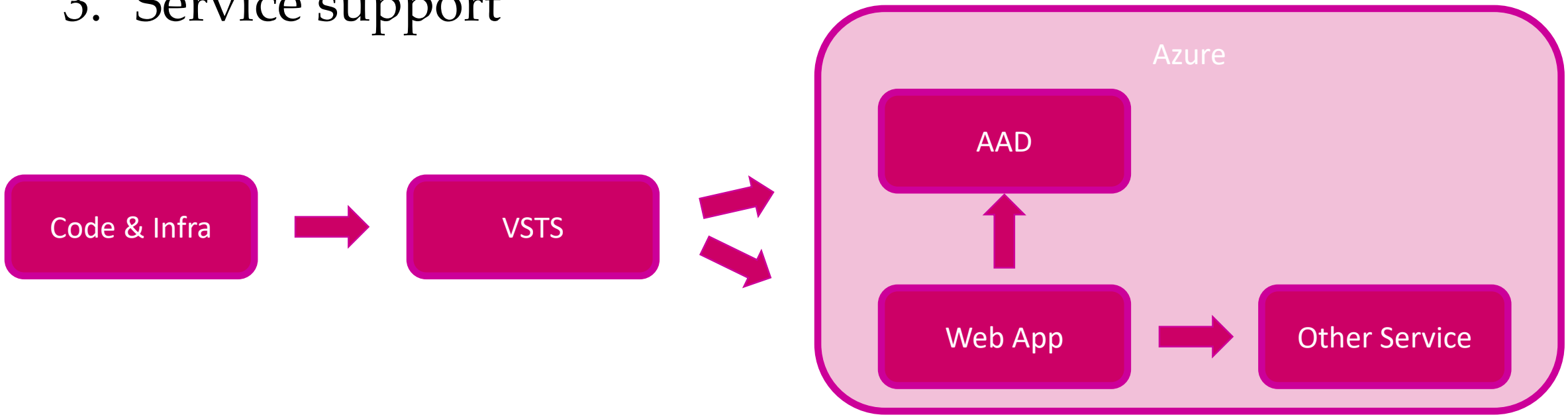
1. Give your application a (runtime) AAD identity
2. Directly access services using that identity



DIRECTLY ACCESS SERVICE

Scenario:

1. You do CD of both infrastructure and code
2. You are allowed to provision to the AD
3. Service support




DEMO TIME!
DIRECTLY ACCESS SERVICE
SPEED:

DIRECTLY ACCESS SERVICE

Pros

- No more secrets

Cons

- MSI is only available on Azure Web Apps
- MSI is not GA yet
- AAD based access only on:
  - Management APIs
  - Key Vault
  - SQL DB
  - Data Lake Store

RECAP
WHAT TO USE WHEN

WHAT TO USE WHEN?

Let team operations deploy	NEVER EVER EVUHRR
Use your release orchestrator	When you deploy only code
Keyvault and ARM templates	When you also deploy infra
Application identity / KeyVault	When available and allowed
Application identity / Oauth resource	When available and allowed

SECRETS FOR LOCAL DEVELOPMENT USING CONFIG BUILDERS

Requires:

- (.NET Framework ≥ 4.7.2 &&.NET Framework 4.7.2 Development Tools) || ASP.NET Core 2.0

Using configuration builders

1. Add Connected Service
2. Four hours of debugging and Googling
3. Update references to configuration builders
4. Update configbuilder configuration
    1. Remove vaultUri
    2. Add connection string

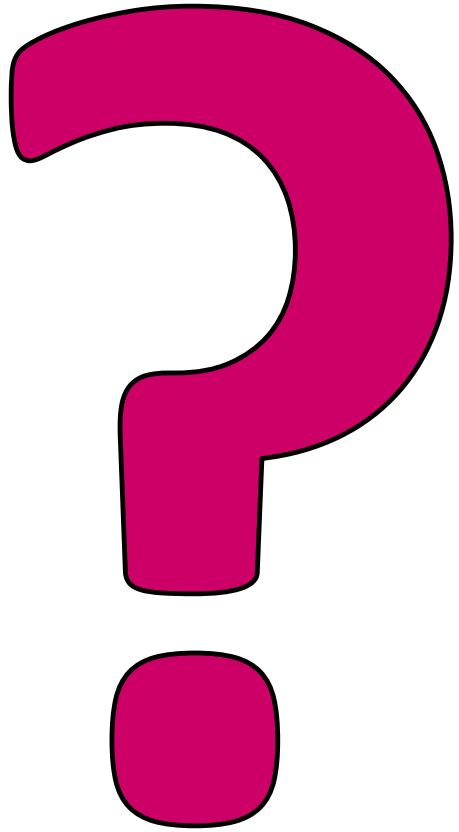
Result: Still secrets in your web.config.

Reason: Still relies on your developer identity being in the AAD

Conclusion: Scenario 3 all over again

RunAs=Developer;

DeveloperTool=VisualStudio



SO **EMBRACE SECRET MANAGEMENT**
AND BE THE BEST **DEVOPS**
YOU CAN BE!

DO TRY THIS **AT HOME!**

HENRY BEEN
Independent Devops & Azure Architect

E: consultancy@henrybeen.nl
T: @henry_been
L: linkedin.com/in/henrybeen
W: henrybeen.nl