nic Cloud Connect

David Pazdera

A practical guide to Test-Driven Development of infrastructure code

nic Cloud Connect

Questions

- What is Test-Driven Development?
- What benefits does it bring aka *What's in it for me*?
- Can TDD be really used for Infrastructure as Code practice?
- Can I re-use existing skills or learn a ton of new stuff?
- Who's this guy speaking?

About me

- cloud architect @ Devoteam M Cloud
- ex ’blue badge’
- meetups, conferences, ACP, communities (ALZ, Azure Arc, Bicep, Terraform in Azure)
- sports & outdoor enthusiast

- GitHub | LinkedIn | Sessionize | SpeakerDeck | X : pazdedav handle
- Blog: https://pazdedav.blog

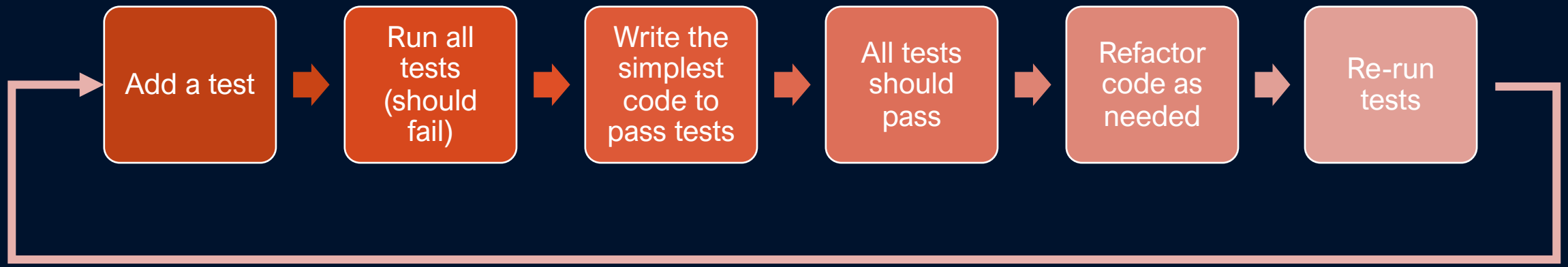
Scenario

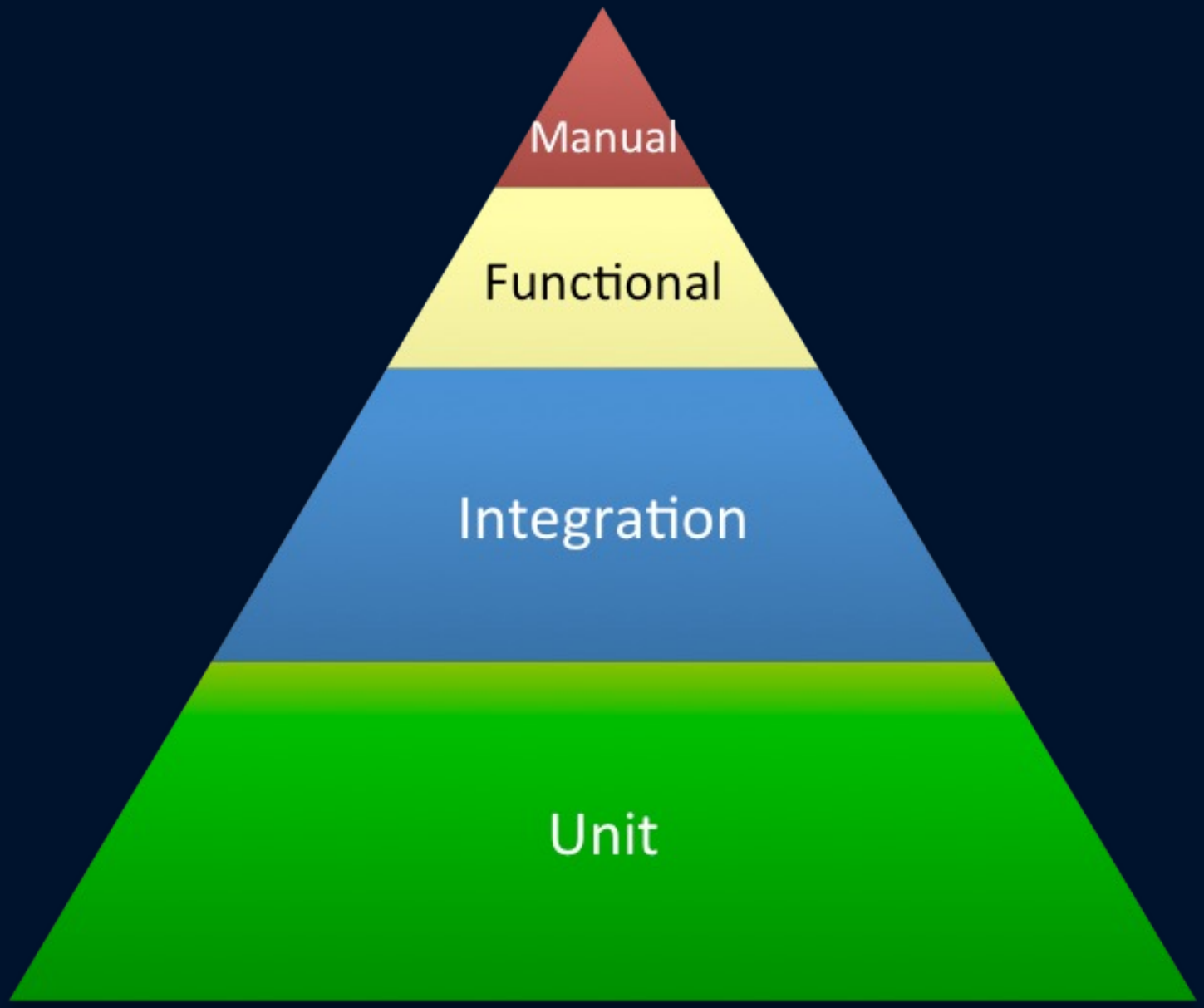
- DevOps engineer in an organization, responsible for Bicep configuration for a project.
- Current technology stack and tooling:

- **Goal:** improve infra code quality using TDD practice and introduce new tools
- **Preferences:**
  - cross-platform
  - free or freemium

Test-driven development (TDD)

Software development process relying on software requirements being converted to test cases before software is fully developed and tracking all software development by repeatedly testing the software against all test cases. *This is as opposed to software being developed first and test cases created later.*




Manual
Functional
Integration
Unit

nic Cloud Connect

We want our infra code to …

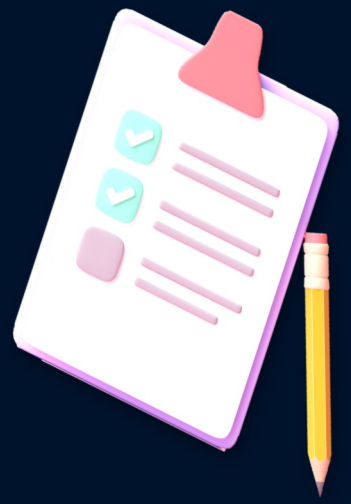
be valid (syntax and coding standards)

✅ follow security best practices

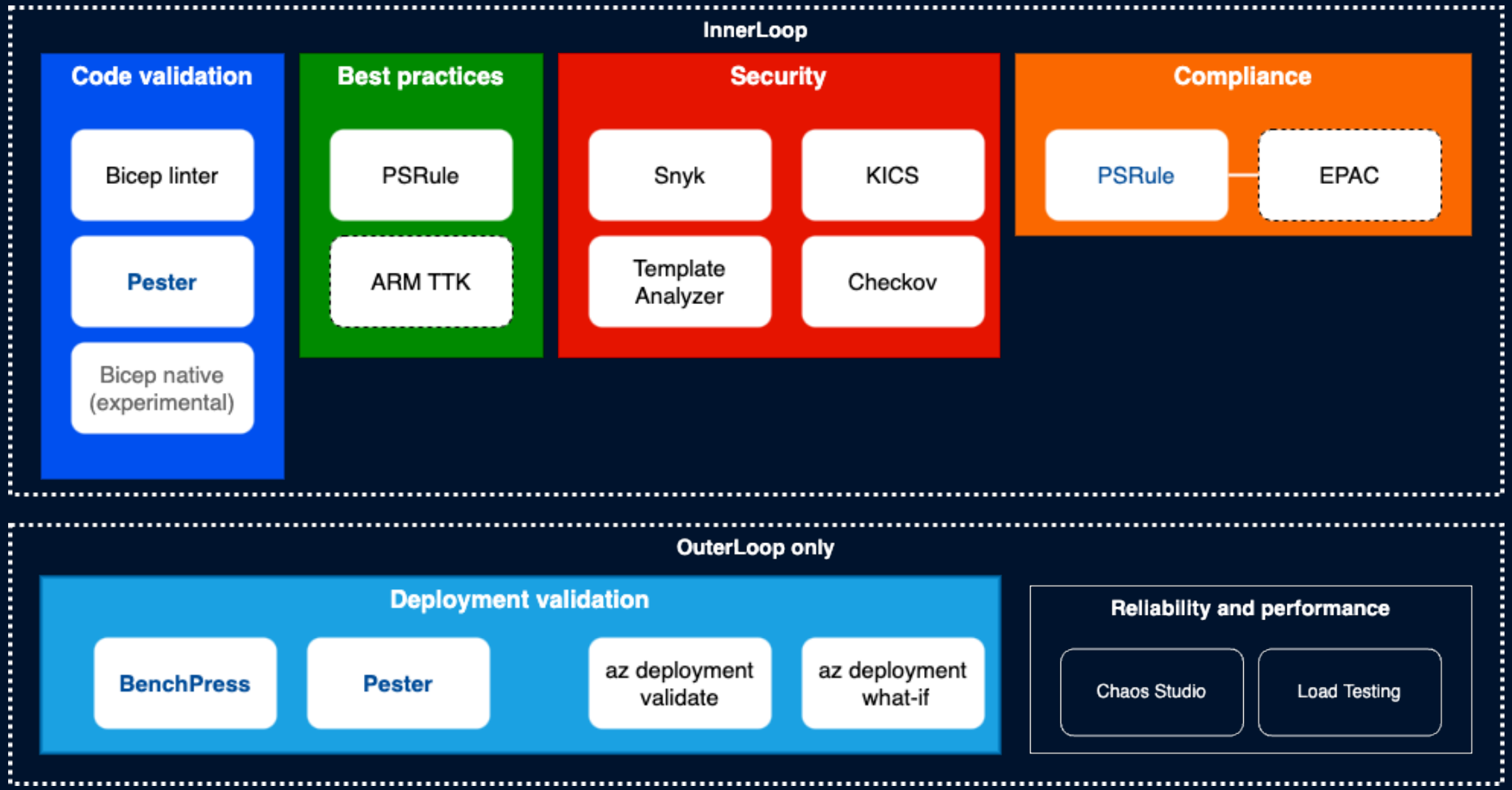
✅ be compliant with target environment's policies

✅ follow cloud provider's best practices like WAF

✅ provision required resources (functional requirements)



Tools and services map



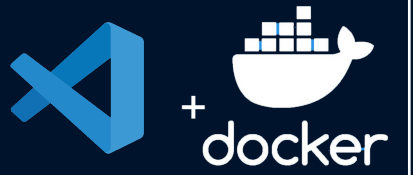
Overview of tools

Tool	Need to write own tests?	Built-in rules	Custom rules
Bicep linter	No	Yes	No
Bicep testing framework	Yes	No	Yes
Pester	Yes	No	Yes
PSRule for Azure	No	Yes	Yes
ARM-TTK	No	Yes	No
KICS, Snyk	No	Yes	No
PSRule for Azure + EPAC	Generate rule collection from existing Azure Policies		
BenchPress	Yes	No	Yes

Coding environment

Local dev environment

All tools installed locally



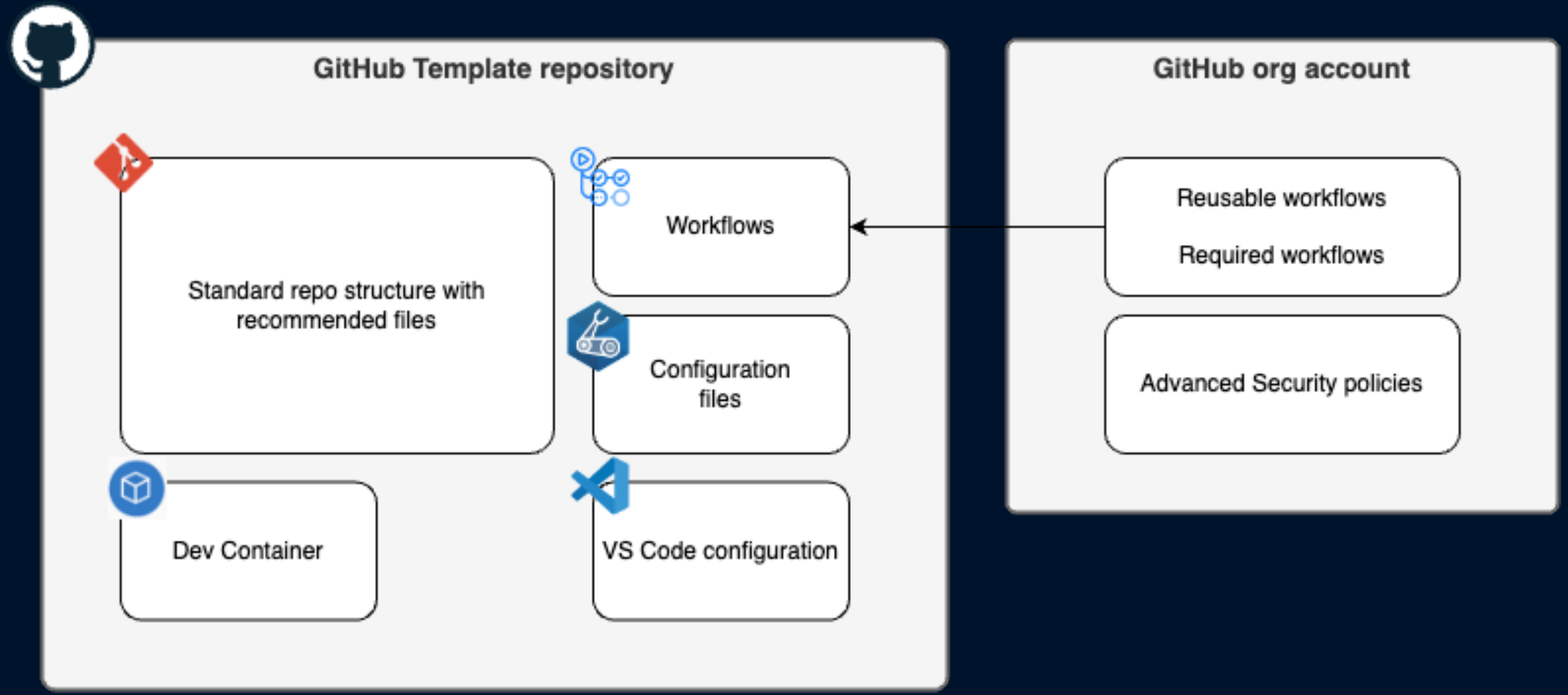
All tools in a Dev Container

Remote dev environment

GitHub Codespaces

Microsoft DevBox

Coding “Landing Zone”

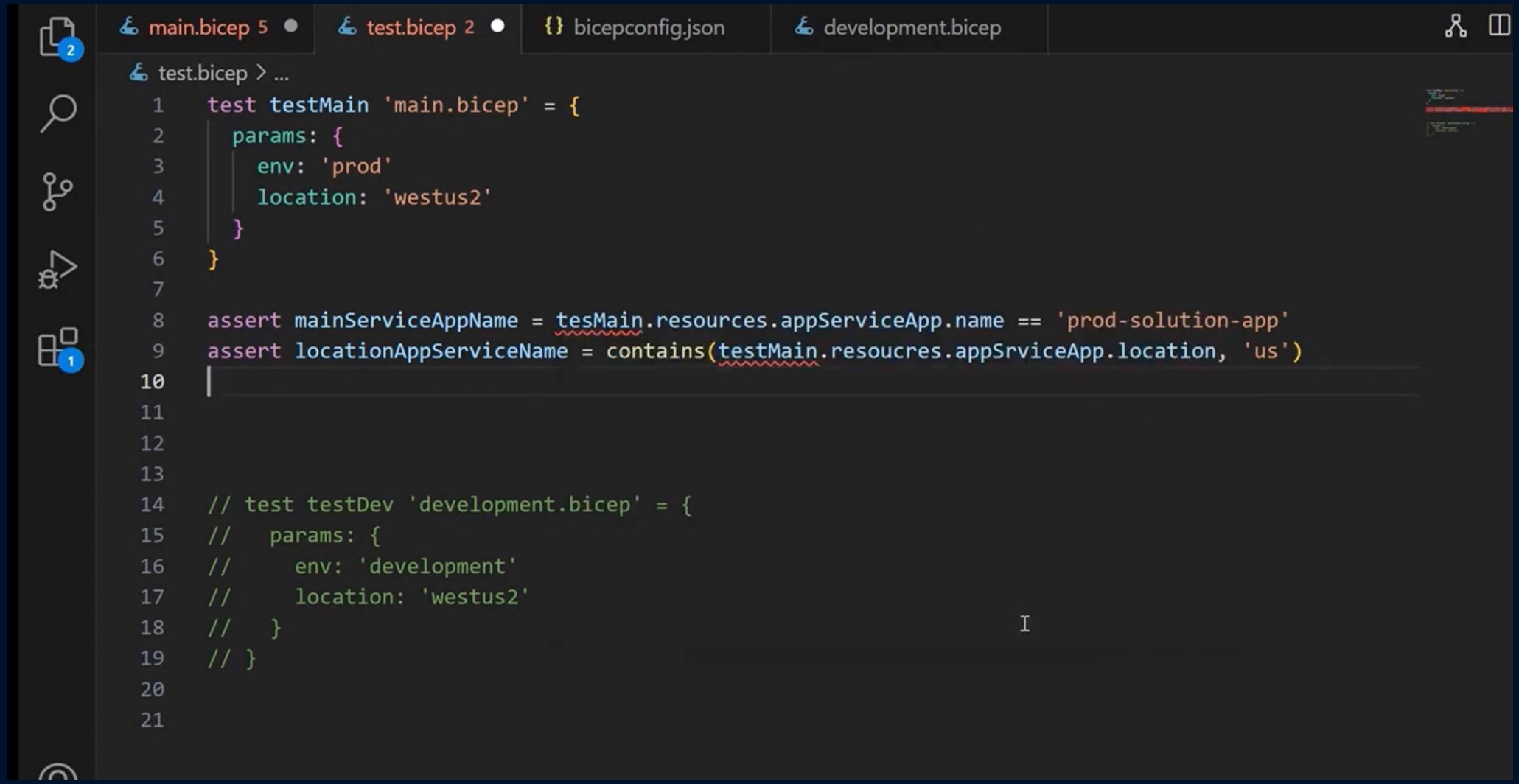


Code validation

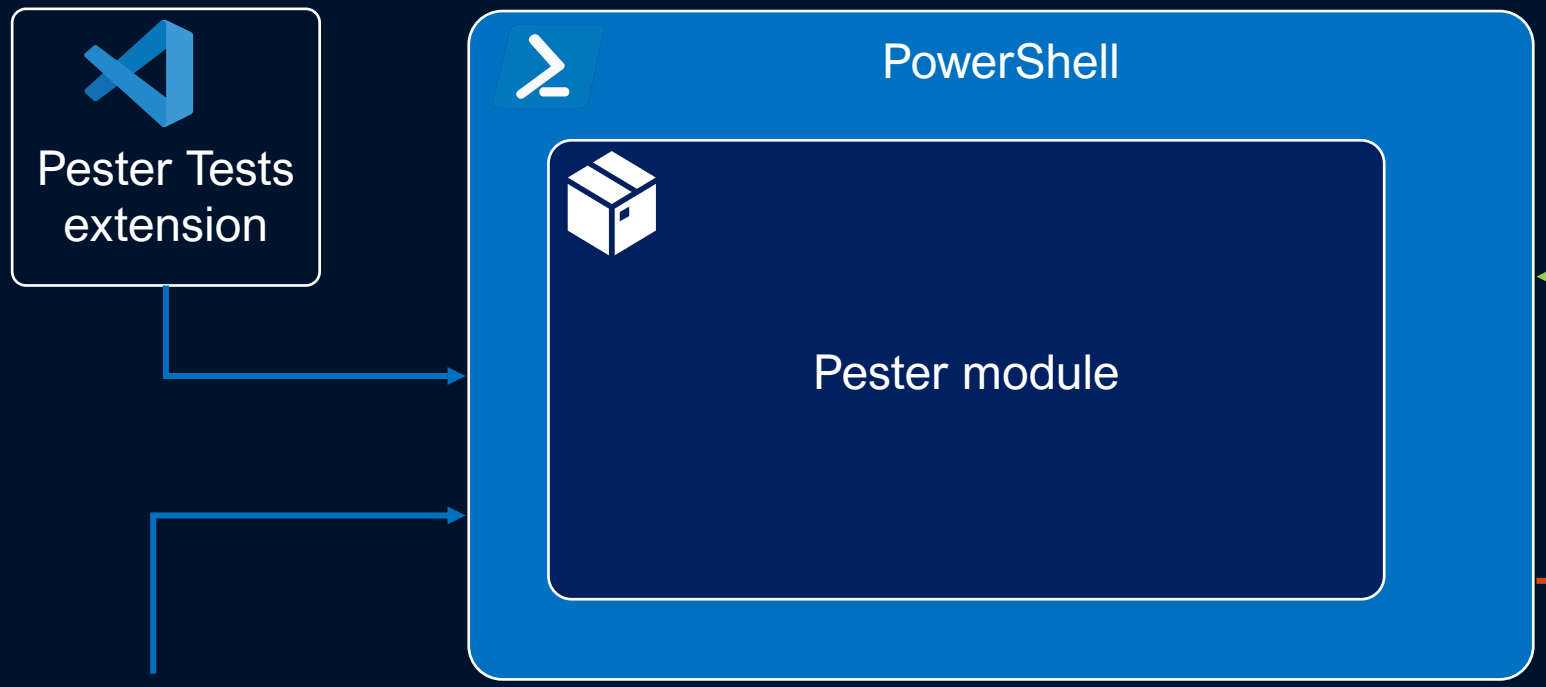
Bicep linter

VS Code extension	code --install-extension ms-azuretools.vscode-bicep
Local installation	brew install azure-cli az bicep upgrade
Local execution	bicep build demo.bicep
Configuration	bicepconfig.json
GH Action	run: az bicep build

Bicep Native Testing Framework



Pester

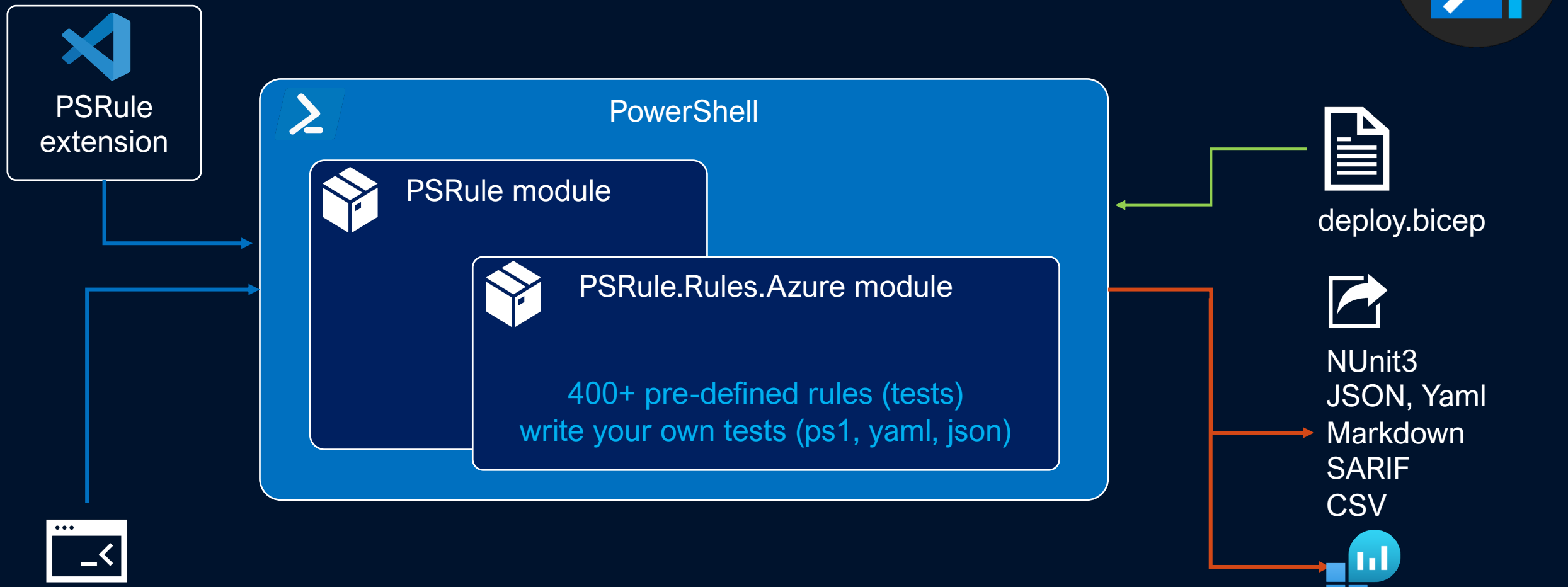

Pester Tests extension
PowerShell
Pester module
deploy.Tests.ps1
deploy.json
NUnit
CI Format
(ADO, GH)
New-PesterContainer
Invoke-Pester -Container

Pester

VS Code extension	code --install-extension pspester.pester-test
Local installation	Install-Module Pester -Force
Local execution	New-PesterContainer | Invoke-Pester -Container
Configuration	
GH Action	run: New-PesterContainer | Invoke-Pester -Container

Best practices validation

PSRule for Azure



PSRule for Azure

VS Code extension	code --install-extension bewhite.psrule-vscode
Local installation	Install-Module -Name 'PSRule' -Repository PSGallery Install-Module -Name 'PSRule.Rules.Azure' -Repository PSGallery
Local execution	Assert-PSRule -InputPath path-to-main.tests.bicep –Module 'PSRule.Rules.Azure'
Configuration	ps-rule.yaml
GH Action	microsoft/ps-rule@v2.9.0, with: modules: 'PSRule.Rules.Azure'

Security testing

Security testing with Snyk

snyk iac test {path-to-arm-template.json} [--report]

Security testing with Snyk

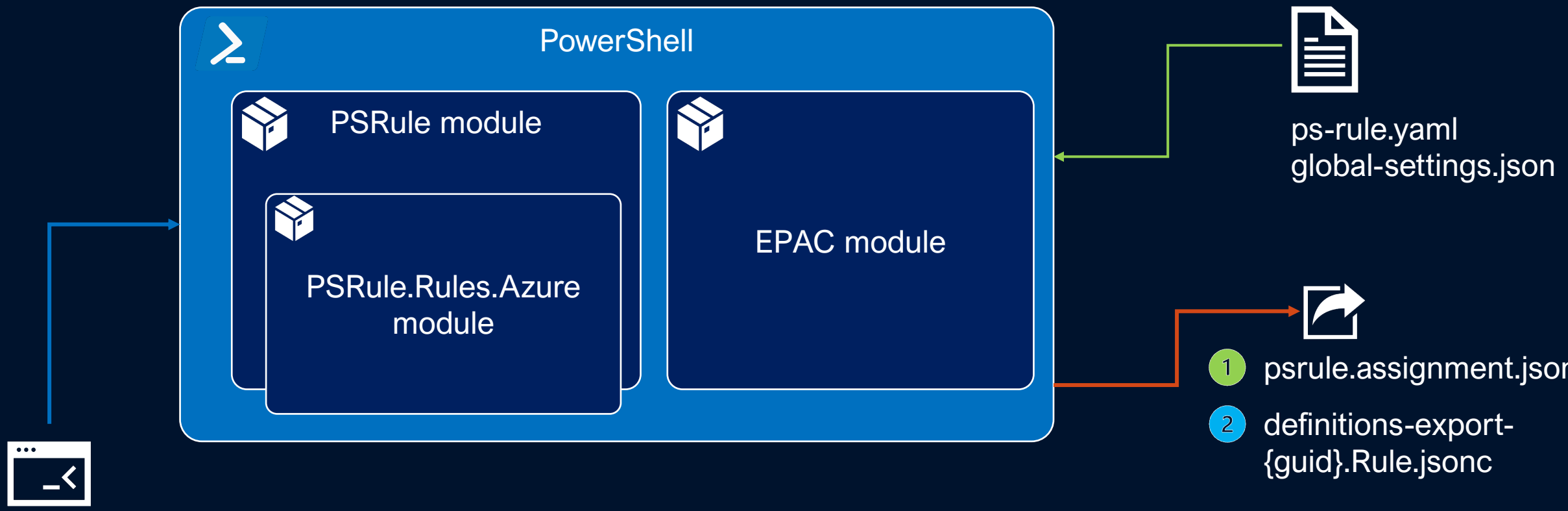
VS Code extension	code --install-extension snyk-security.snyk-vulnerability-scanner
Local installation	brew tap snyk/tap brew install snyk
Local execution	snyk auth snyk iac test {file_name}.json
Configuration	
GH Action	snyk/actions/iac@master

Security testing with KICS

VS Code extension	code --install-extension checkmarx.ast-results
Local installation	Docker
Local execution	docker run -t -v {path_to_host_folder_to_scan}:/path checkmarx/kics:latest scan -p /path -o "/path/"
Configuration	
GH Action	Checkmarx/kics-github-action@v1.7.0

Compliance validation

Compliance with PSRule + EPAC

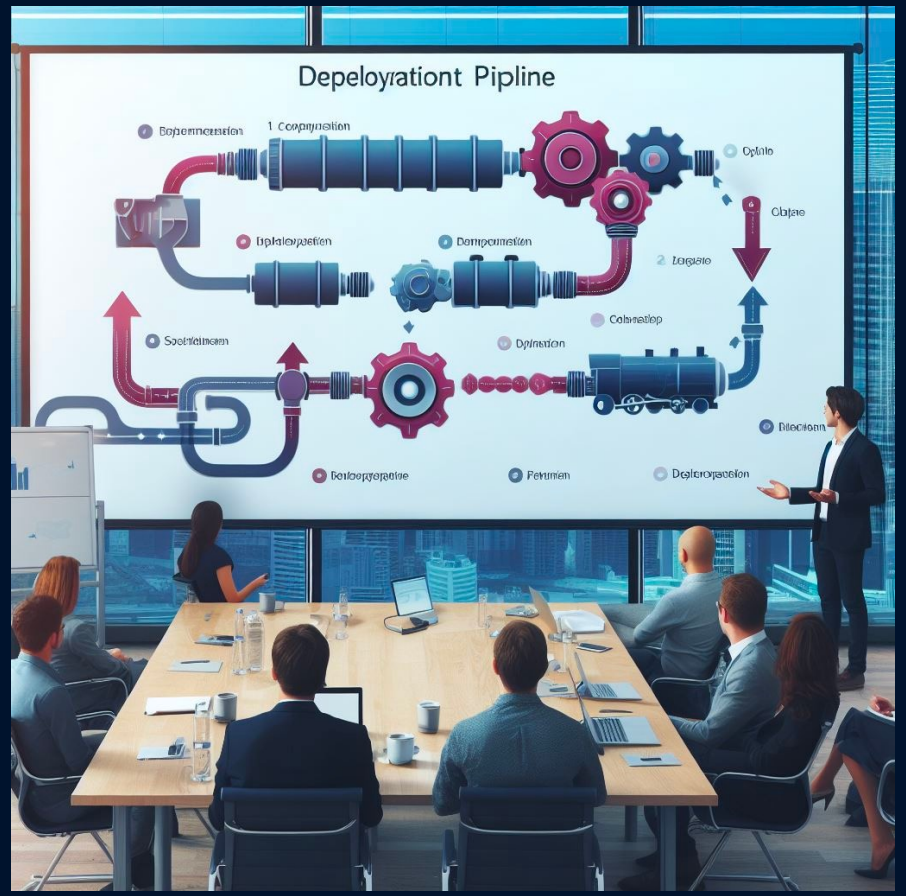


1. Export-AzPolicyResources -DefinitionsRootFolder .\ -Mode psrule -OutputFolder .\
2. Export-AzPolicyAssignmentRuleData –AssignmentFile .\psrule.assignment.json –OutputPath .\
3. Assert-PSRule –InputPath .\ -Module “PSRule.Rules.Azure” –Format File

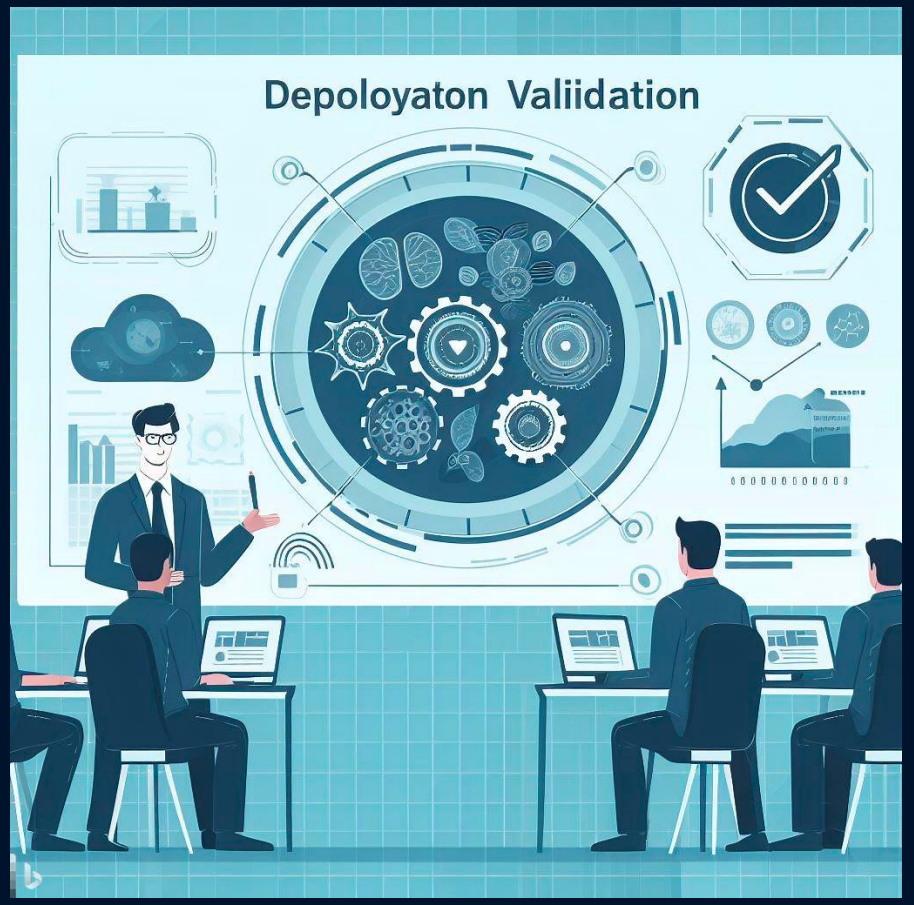
Compliance with PSRule + EPAC

VS Code extension	code --install-extension bewhite.psrule-vscode
Local installation	Install-Module -Name 'PSRule' -Repository PSGallery Install-Module -Name 'PSRule.Rules.Azure' -Repository PSGallery Install-Module -Name 'EnterprisePolicyAsCode' -Repository PSGallery
Local execution	Assert-PSRule -InputPath path-to-main.tests.bicep
Configuration	psrule.yaml
GH Action	microsoft/ps-rule@v2.9.0, with: modules: 'PSRule.Rules.Azure'

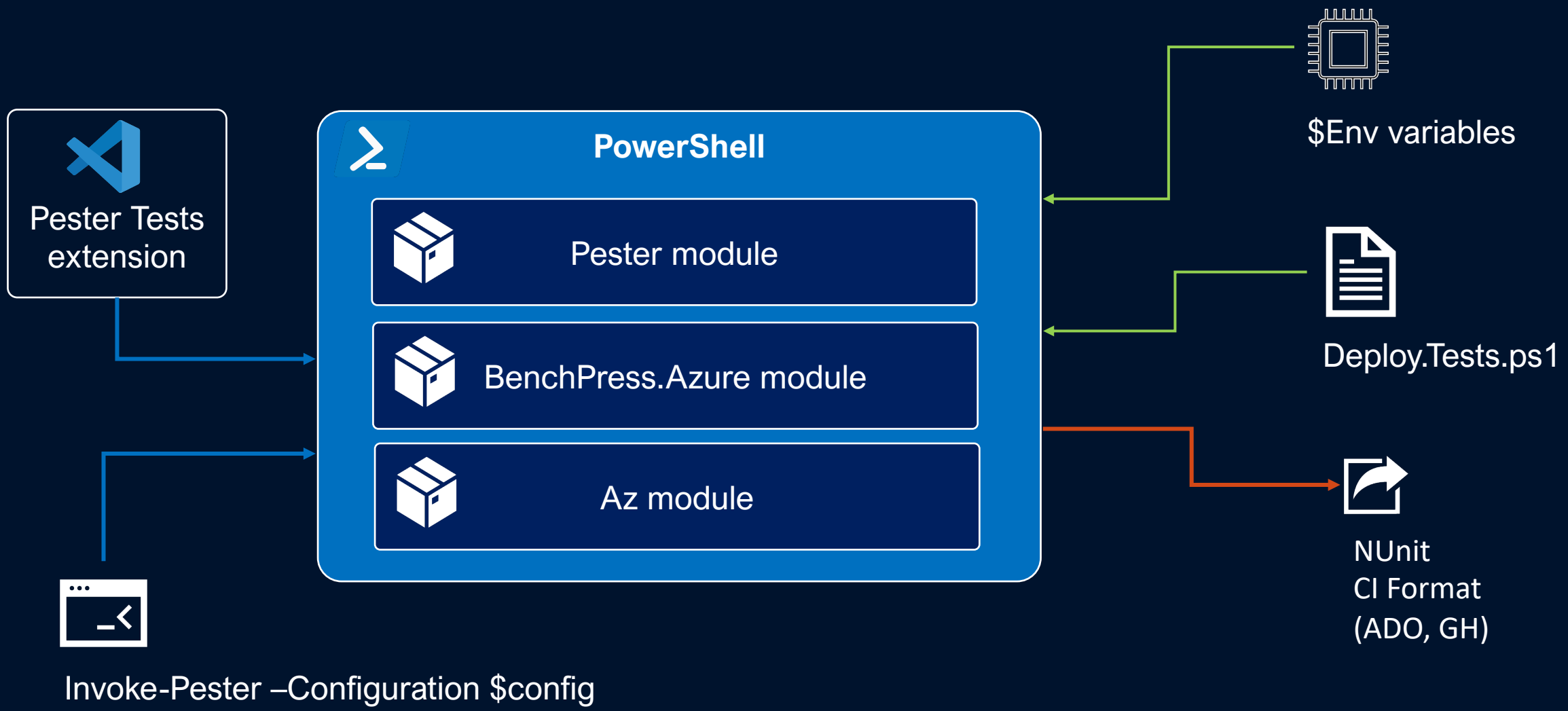
OuterLoop



Deployment validation



BenchPress

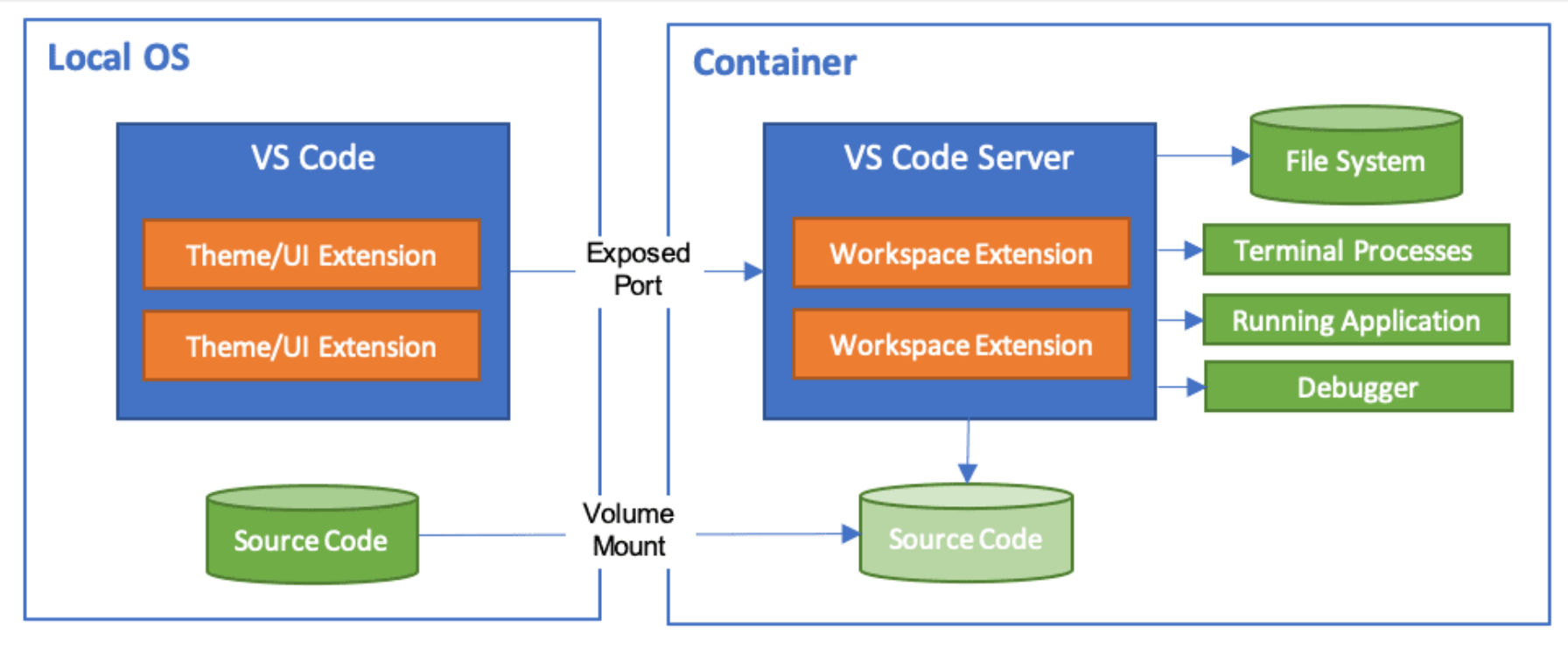

Pester Tests extension
PowerShell
Pester module
BenchPress.Azure module
Az module
$Env variables
Deploy.Tests.ps1
NUnit
CI Format
(ADO, GH)
Invoke-Pester –Configuration $config

BenchPress

VS Code extension	code --install-extension pspester.pester-test
Local installation	Install-Module Pester -Force Install-Module Az -Force Install-Module -Name ‘BenchPress.Azure' -Repository PSGallery
Local execution	Invoke-Pester -Path .\File.Tests.ps1
Configuration	*Pester Configuration object*
GH Action	azure/powershell@v1

nic Cloud Connect

Wrapping things up…



Honorable mentions

- **Template Analyzer** – https://github.com/Azure/template-analyzer
  - Template scanner for security misconfiguration and best practices
  - Microsoft Security DevOps (Preview)
    - CLI and GitHub action
    - support for SARIF, integration with GHAS
    - uses Template Analyzer in the background

Tools

Name	Language
AntiMalware	code, artifacts
Bandit	python
BinSkim	binary - Windows, ELF
ESlint	JavaScript
Template Analyzer	Infrastructure-as-code (IaC), ARM templates, Bicep files
Terrascan	Infrastructure-as-code (IaC), Terraform (HCL2), Kubernetes (JSON/YAML), Helm v3, Kustomize, Dockerfiles, Cloudformation
Trivy	container images, file systems, and git repositories

nic Cloud Connect

Questions

What is Test-Driven Development?

What benefits does it bring aka What's in it for me?

Can TDD be really used for Infrastructure as Code practice?

Can I re-use existing skills or learn a ton of new stuff?

Who's this guy speaking?

nic Cloud Connect

Code repository

https://github.com/pazdedav/nic-2023-project