Azure Pipelines: CI/CD

Shailendra Chauhan

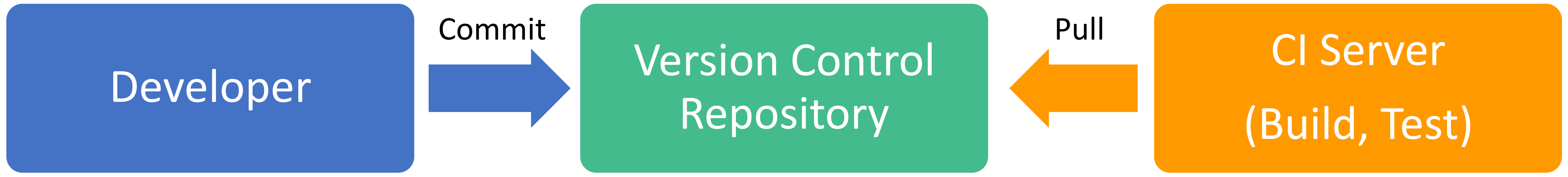
Microsoft MVP, Technical Consultant and Corporate Trainer

DotNetTricks

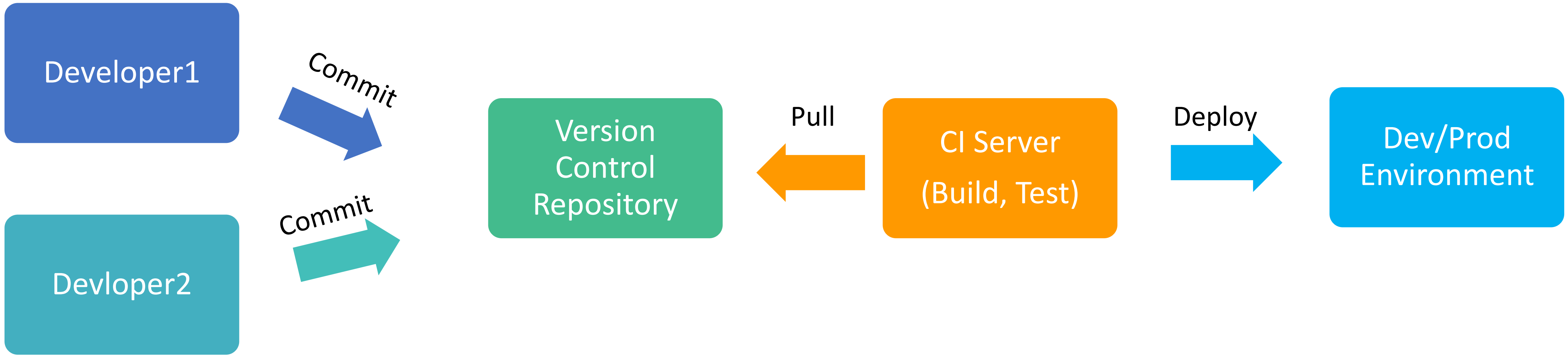
Azure Pipeline

- A cloud service, used to automatically build and test your project code and making it available to users.
- Works with any language or project type.
- Combines continuous integration (CI) and continuous delivery (CD) to constantly and consistently test and build your code and ship it to any target.

Continuous Integration

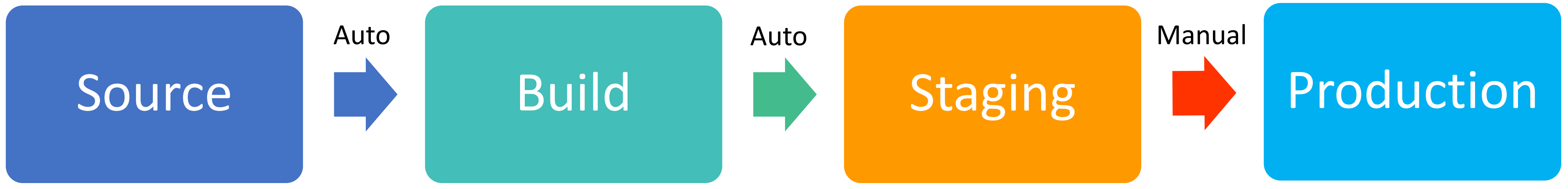


Continuous Delivery (CD)

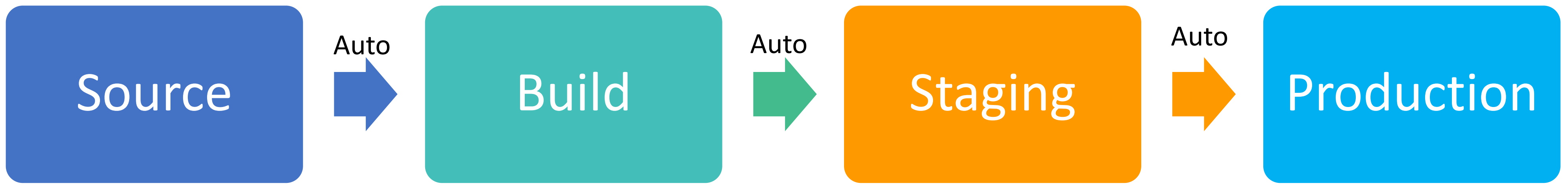


DotNetTricks

Continuous Delivery & Continuous Deployment



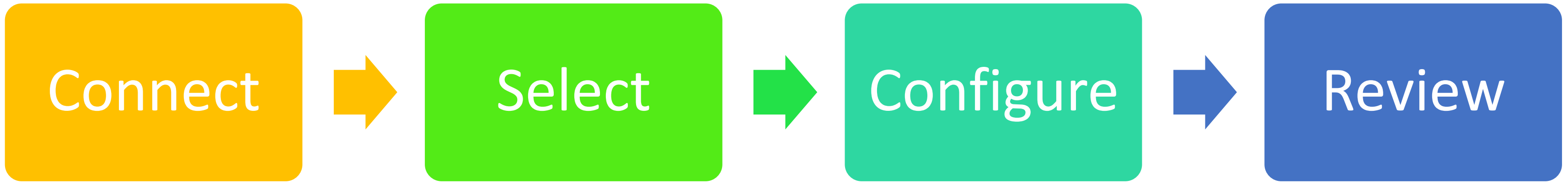
Continuous Delivery



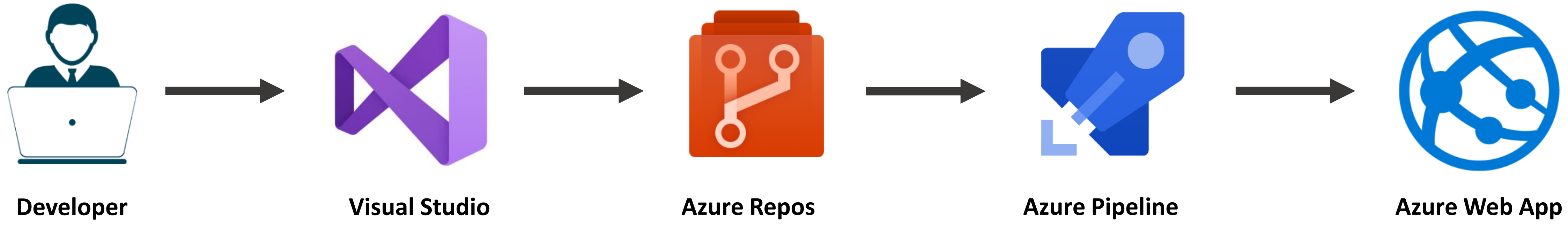
Continuous Deployment

DotNetTricks

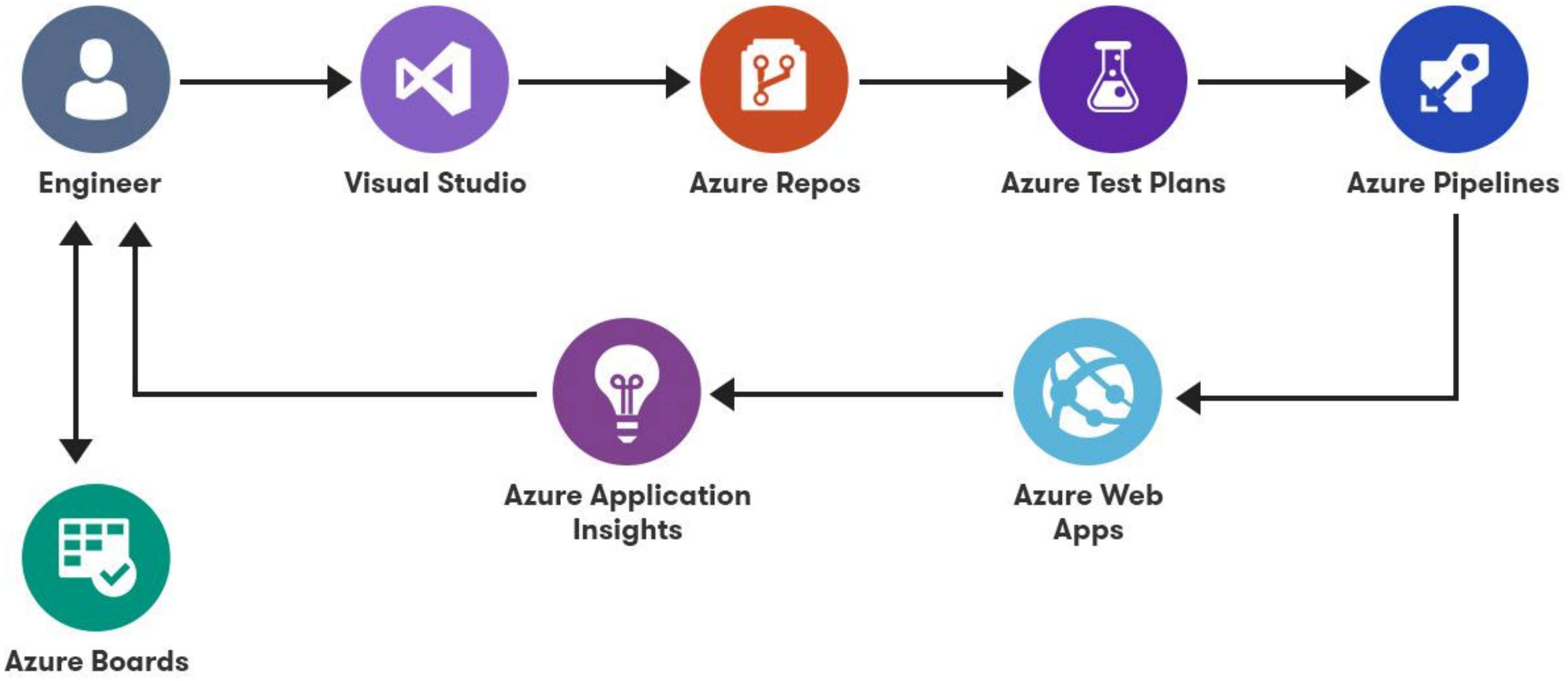
Steps to Configure a Azure Pipeline



Azure Pipeline with Azure Web Apps



Azure Pipeline with Azure Web Apps



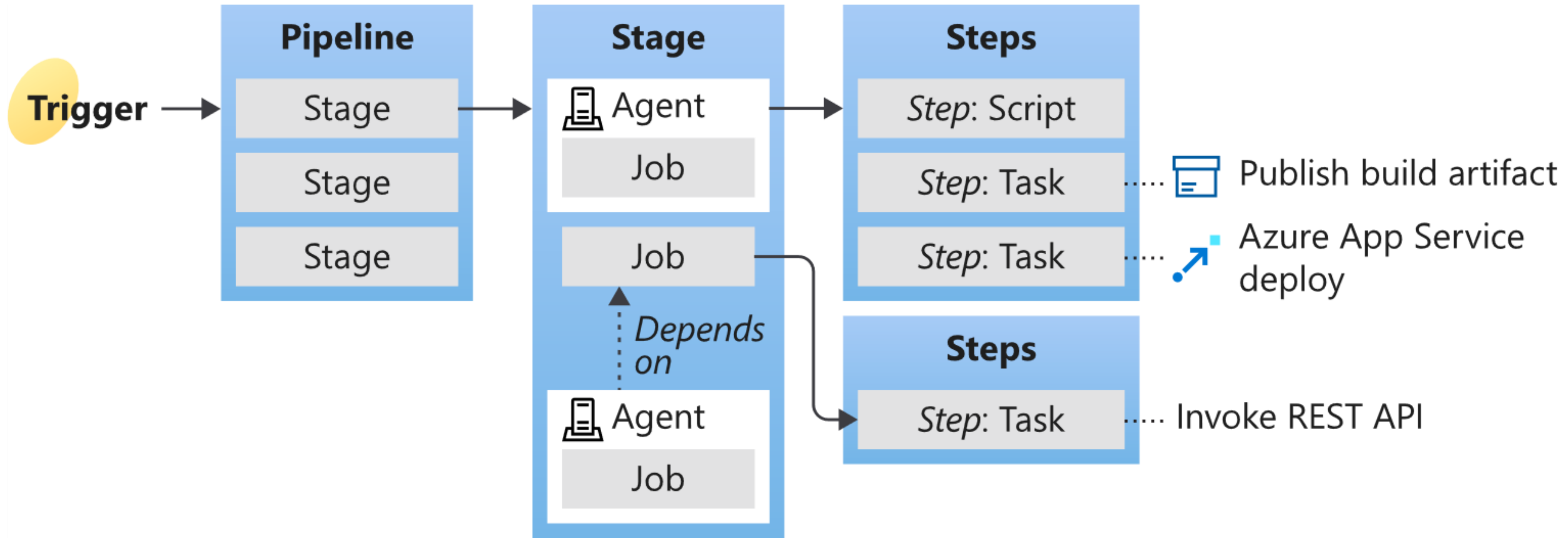
Azure Pipelines Platform Support

.NET Core	Android	Docker	Go
Java	Kubernetes	Linux VM	Node.js
npm	NuGet	PHP	Python
Ruby	UWP	Xamarin	Xcode

DotNetTricks

Azure Pipeline Structure

- A pipeline has one or more stages to describe a CI/CD process.
- Stages are the major divisions in a pipeline.



DotNetTricks

Azure Pipeline YAML Example

```yaml
name: my-pipeline-v1
variables:
 project: xyz
 build_no: 1.0.1

stages:
- stage: Build
  jobs:
  - job: BuildJob
    steps:
    - script: 'echo Building $(project)!'
- stage: Deploy
  jobs:
  - job: Deploy
    steps:
    - script: 'echo Deploying $(build_no)!'
```

DotNetTricks

Azure Pipeline: Stage

- A stage is a collection of related jobs.
- By default, stages run sequentially. Each stage starts only after the preceding stage is complete.
- Use approval checks to manually control when a stage should run.

```
stages:
- stage: string  # name of the stage (A-Z, a-z, 0-9, and underscore)
  displayName: string  # friendly name to display in the UI
  dependsOn: string | [ string ]
  condition: string
  variables:
  jobs: []
```

Azure Pipeline: Job

- A job is a collection of steps run by an agent or on a server.
- A Job can run conditionally and depend on earlier job.

```
jobs:
- job: string  # name of the job
  displayName: string  # name to display in the UI
  dependsOn: string | [ string ]
  condition: string
  strategy:
    parallel: # parallel strategy
  continueOnError: boolean  # defaults to 'false'
  pool: pool
  workspace:
    clean: outputs | resources | all # clean up before the job runs
  variables: # define variable
  steps: [] # define variable
```

DotNetTricks

Azure Pipeline: Steps

- A step is a linear sequence of operations to make a job.
- Each step runs in its own process on an agent and has access to the pipeline workspace.
- Environment variables aren't preserved between steps but file system changes preserved.

```
steps: [ script | bash | pwsh | powershell | checkout | task | templateReference ]
```

Agents

- An agent is a software that runs one job at a time.
- To build your code or deploy your code using Azure Pipelines, you need at least one agent.
- Jobs can be run directly on the host machine of the agent or in a container.
- Two Types of agents : Microsoft-hosted agents, Self-hosted agents

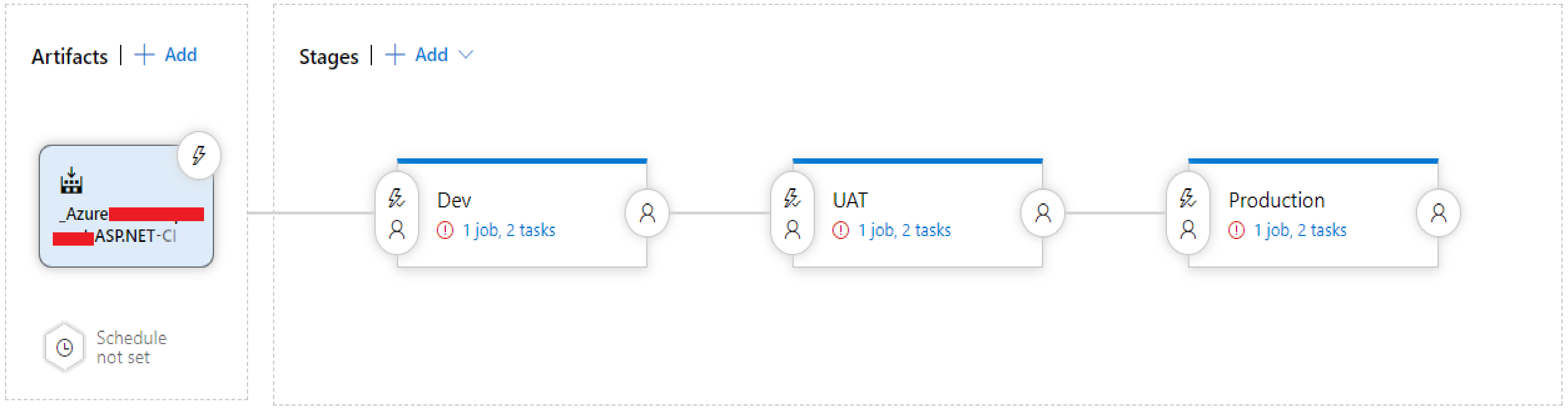
Microsoft-hosted agents

- With Microsoft-hosted agents, maintenance and upgrades are taken care by Microsoft.
- Each time you run a pipeline, you get a fresh virtual machine.
- The virtual machine is discarded after one use.
- Microsoft-hosted agents can run jobs directly on the VM or in a container.

Self-hosted agents

- An agent that you set up and manage to run jobs.
- Self-hosted agents give you more control to install dependent software needed for your builds and deployments.
- A Self-hosted can be installed on Linux, macOS, Windows machines or Docker container.

Release pipelines



DotNetTricks

YAML

- YAML is a human friendly data serialization standard for all programming languages.
- Used for defining configuration or in an application where data is being stored or transmitted.
- YAML is introduced in 2001.
- Extension are .yml or .yaml
- YAML is case sensitive.
- YAML supports spaces instead of tabs.

XML vs. JSON vs. YAML

```xml
<Servers>
    <Server>
        <name>server1</name>
        <location>india</location>
        <status>active</status>
    </Server>
</Servers>
```

```json
{
    "Servers": [
        "Server": {
            "name": "server1",
            "location": "india",
            "status": "active"
        }
    ]
}
```

```yaml
Servers:
  - name: server1
    location: india
    status: active
```

DotNetTricks

YAML in Action

```yaml
integer: 25
string1: name
string2: "name"
string3: 'name'
float: 25.0
boolean: true
```

Basic: Datatype

```yaml
name: server1
location: india
status: active
```

Key/Value

```yaml
Servers:
  - server1
  - server2
  - server3
```

Array/List: Ordered

```yaml
Servers:
    name: server1
    status: active
    location: india
```

Dictionary/Map: Unordered

```yaml
Servers:
  - server1:
      location: india
      status: active
  - server2:
      location: usa
      status: active
```

Array/Dictionary/Key-value

DotNetTricks

```yaml
# ASP.NET Core Azure Build Pipeline YAML

trigger:
- master

pool:
  vmImage: 'ubuntu-latest'
  #vmImage: 'windows-latest'
variables:
  buildConfiguration: 'Release'

steps:
- script: dotnet build --configuration $(buildConfiguration)
  displayName: 'dotnet build $(buildConfiguration)'

- task: DotNetCoreCLI@2
  inputs:
    command: 'publish'
    publishWebProjects: true
    arguments: '--configuration $(buildConfiguration) --output  "$(Build.ArtifactStagingDirectory)"'

- task: PublishBuildArtifacts@1
  inputs:
    PathtoPublish: '$(Build.ArtifactStagingDirectory)'
    ArtifactName: 'drop'
    publishLocation: 'Container'
```

```
# ASP.NET Core Stages
trigger:
- main
pool:
  #vmImage: 'windows-latest'
  vmImage: 'ubuntu-latest'

variables:
  buildConfiguration: 'Release'

stages:
- stage: Dev
  jobs:
    - job:
      steps:
      - script: dotnet build --configuration $(buildConfiguration)
        displayName: 'dotnet build $(buildConfiguration)'
      - task: DotNetCoreCLI@2
        inputs:
          command: 'publish'
          publishWebProjects: true
          arguments: '--configuration $(BuildConfiguration) --output "$(build.artifactstagingdirectory)"'
      - task: PublishBuildArtifacts@1
        inputs:
          PathtoPublish: '$(Build.ArtifactStagingDirectory)'
```

```yaml
# Node.js with Angular Build Pipeline YAML

trigger:
- master

pool:
  vmImage: 'ubuntu-latest'

steps:
- task: NodeTool@0
  inputs:
    versionSpec: '10.x'
  displayName: 'Install Node.js'

- script: |
    npm install -g @angular/cli
    npm install
    ng build --prod
  displayName: 'npm install and build'

- task: PublishBuildArtifacts@1
  inputs:
    PathtoPublish: 'dist/myapp'
    ArtifactName: 'drop'
    publishLocation: 'Container'
```

Azure Pipeline Built-In Variables

- Agent variables - Agent.BuildDirectory, Agent.JobStatus, Agent.Name etc.
- Build variables - Build.ArtifactStagingDirectory, Build.BuildNumber etc.
- Pipeline variables - Pipeline.Workspace
- Deployment job variables - Environment.Name, Environment.Id etc.
- System variables - System.JobName, System.StageName etc.

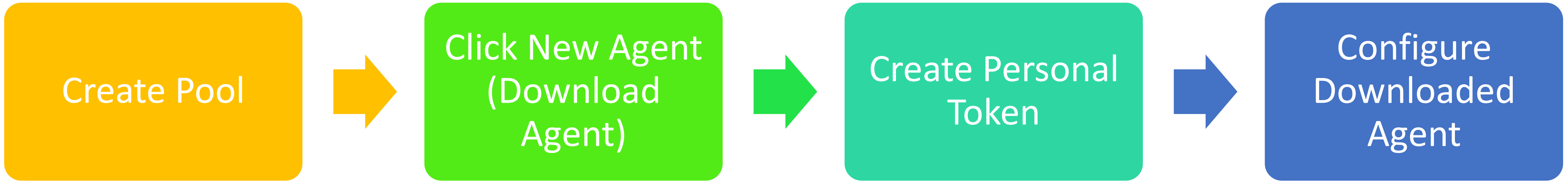
Reference: https://docs.microsoft.com/en-us/azure/devops/pipelines/build/variables

Variable Scopes

- At the root level, to make it available to all jobs in the pipeline.
- At the stage level, to make it available only to a specific stage.
- At the job level, to make it available only to a specific job.

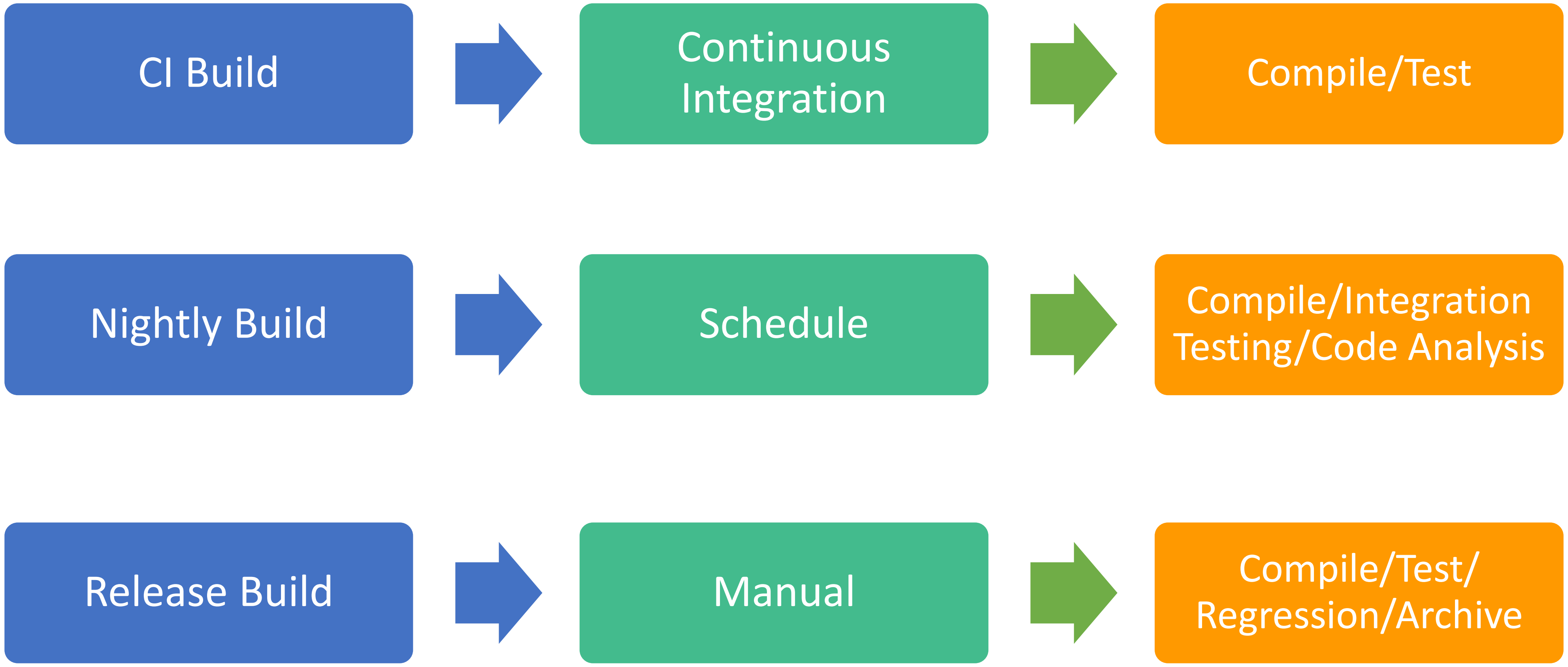
```yaml
variables:
# a regular variable
- name: myvariable
  value: myvalue
# a variable group
- group: myvariablegroup
# a reference to a variable template
- template: myvariabletemplate.yml
```

Steps to Configure a Custom Agent



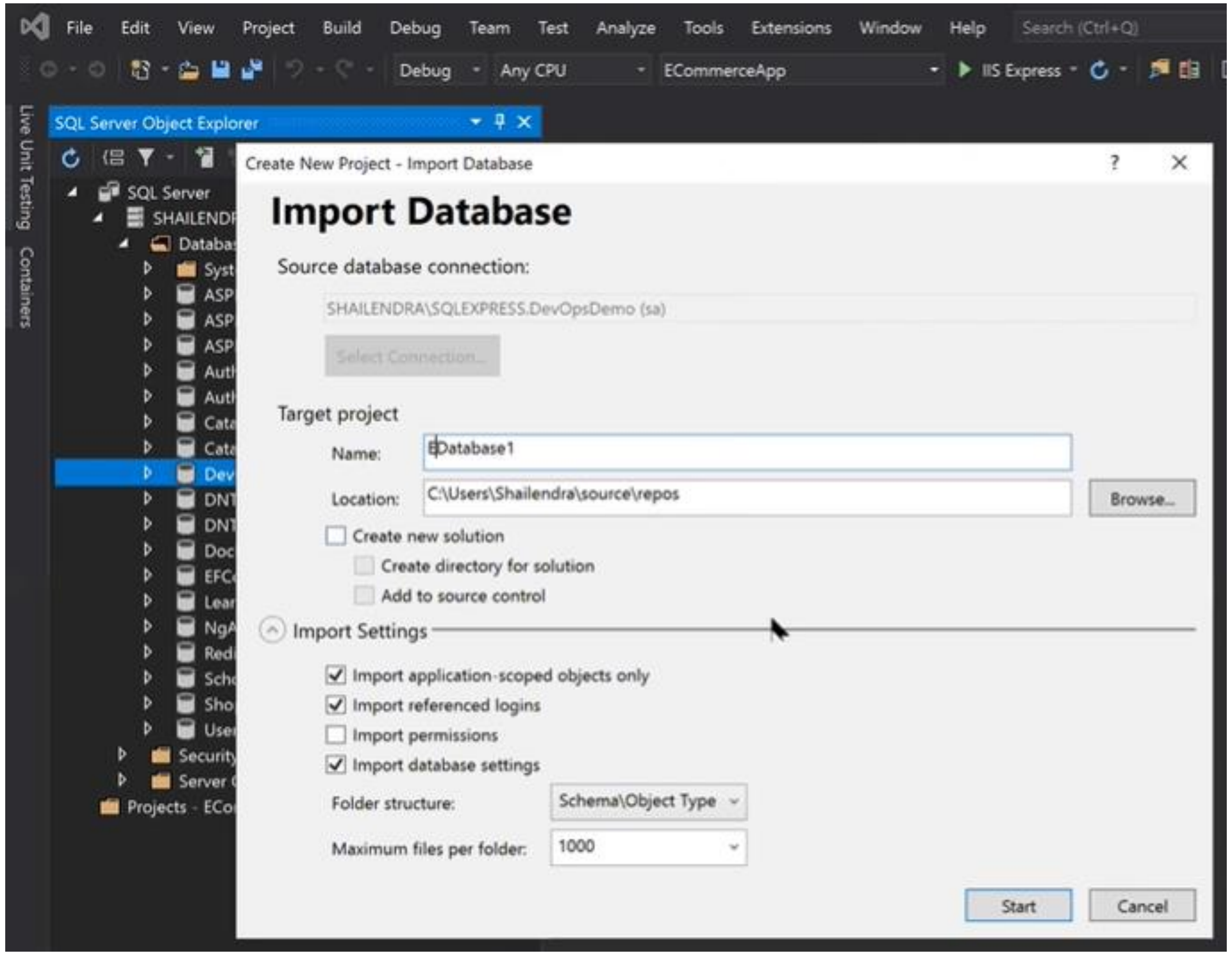
```
> C:\CustomAgent>config
> Enter server Url > https://dev.azure.com/dnttestorg
> Enter personal access token > vcrqdthhpmskije7rygnai2grrzet4ock7e4cmztrpll4crqbtxq
> Enter agent pool > Demo Pool
> Enter agent name > MyDemoVm
> Enter work folder > C:\CustomAgent\workdir

-- If it's offline
> C:/CustomAgent> run
```

Different Types of Builds



DotNetTricks

Database Pipeline: Creating Project



Create New Project - Import Database

Import Database

Source database connection:

SHAILENDRA\SQLEXPRESS.DevOpsDemo (sa)

Select Connection...

Target project

Name:	EDatabase1
Location:	C:\Users\Shailendra\source\repos

Browse...

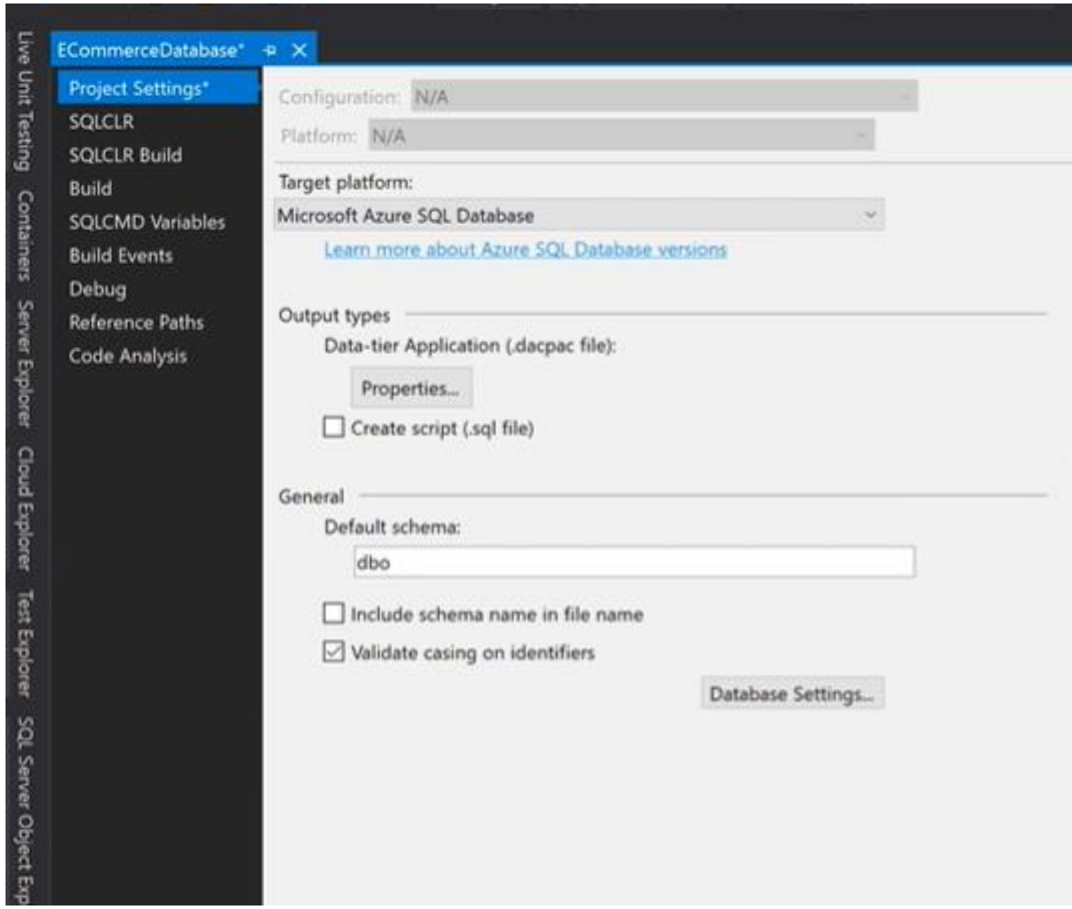
- [ ] Create new solution
- [ ] Create directory for solution
- [ ] Add to source control

Import Settings

- [x] Import application-scoped objects only
- [x] Import referenced logins
- [ ] Import permissions
- [x] Import database settings

Folder structure:	Schema\Object Type
Maximum files per folder:	1000

Start   Cancel



Configuration:	N/A
Platform:	N/A

Target platform:

Microsoft Azure SQL Database

Learn more about Azure SQL Database versions

Output types

Data-tier Application (.dacpac file):

Properties...

- [ ] Create script (.sql file)

General

Default schema:

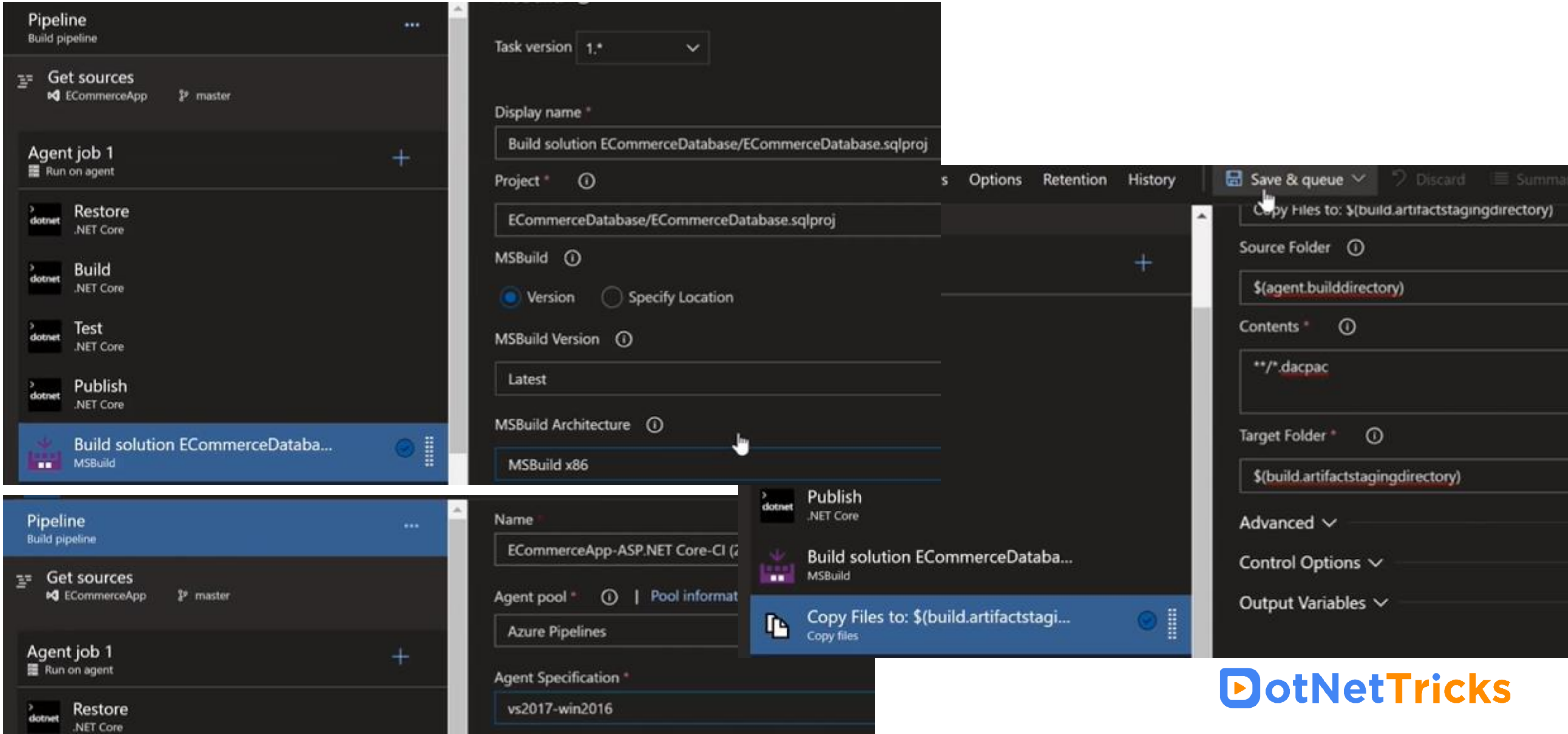
dbo

- [ ] Include schema name in file name
- [x] Validate casing on identifiers

Database Settings...

DotNetTricks

Database Pipeline: Creating Build Pipeline



Pipeline
Build pipeline

Get sources
EcommerceApp   master

Agent job 1
Run on agent

Restore
.NET Core

Build
.NET Core

Test
.NET Core

Publish
.NET Core

Build solution ECommerceDataba...
MSBuild

Task version: 1.*

Display name *: Build solution ECommerceDatabase/ECommerceDatabase.sqlproj

Project *: ECommerceDatabase/ECommerceDatabase.sqlproj

MSBuild: [✓ Version] Specify Location

MSBuild Version: Latest

MSBuild Architecture: MSBuild x86

Copy Files to: $(build.artifactstagingdirectory)

Source Folder: $(agent.builddirectory)

Contents *: **/*.dacpac

Target Folder *: $(build.artifactstagingdirectory)

Publish
.NET Core

Build solution ECommerceDataba...
MSBuild

Copy Files to: $(build.artifactstagi...
Copy files

Pipeline
Build pipeline

Get sources
EcommerceApp   master

Agent job 1
Run on agent

Restore
.NET Core

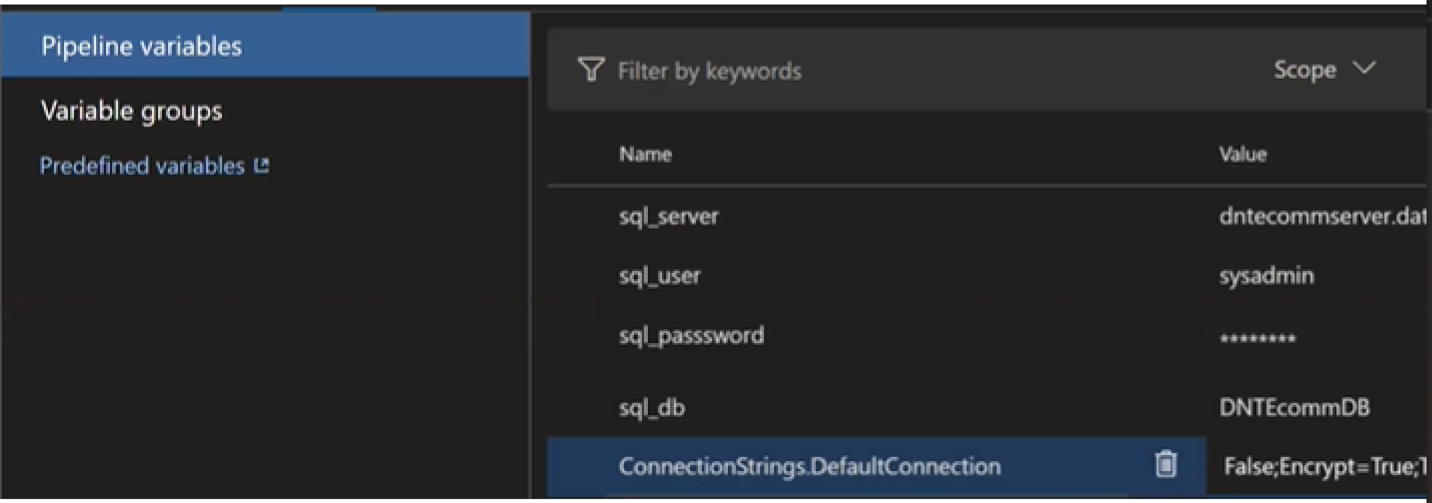
Name *: ECommerceApp-ASP.NET Core-CI (

Agent pool *: Azure Pipelines | Pool informat

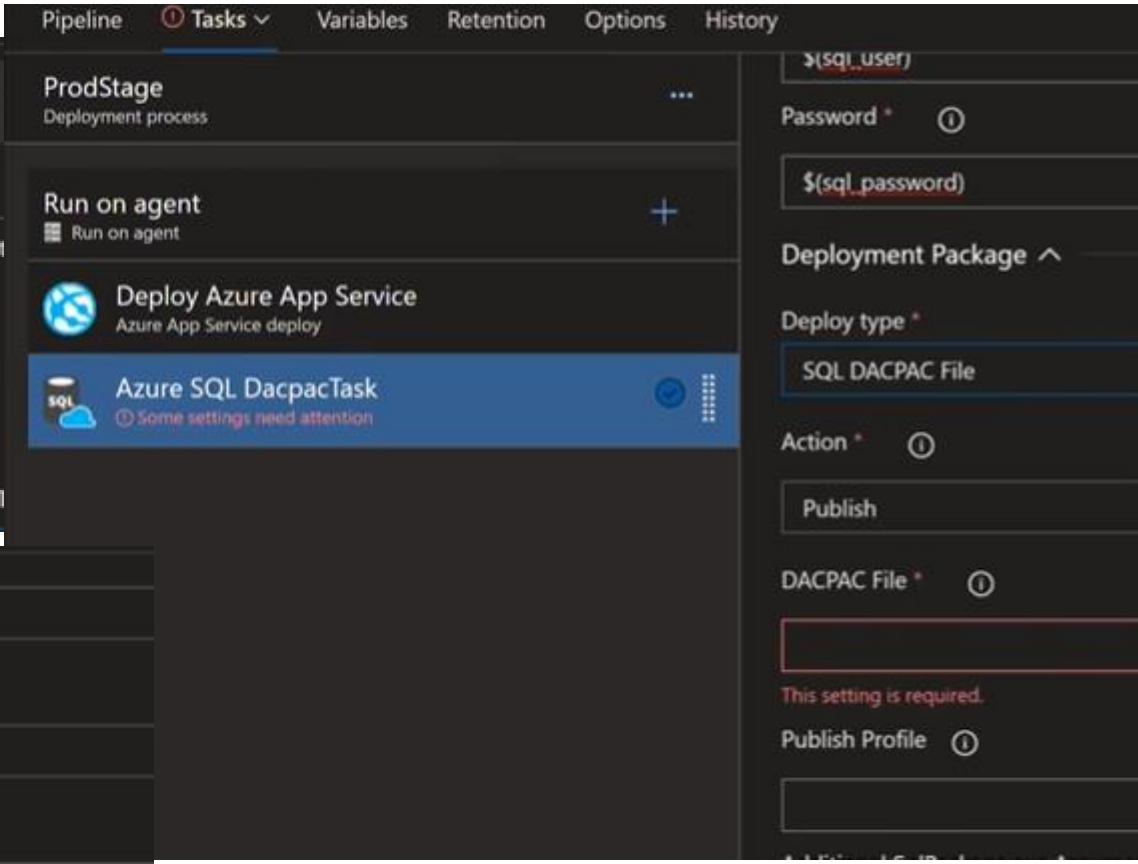
Agent Specification *: vs2017-win2016

DotNetTricks

Database Pipeline: Creating Release Pipeline



Name	Value
sql_server	dntecommserver.dat
sql_user	sysadmin
sql_passsword	********
sql_db	DNTEcommDB
ConnectionStrings.DefaultConnection	False;Encrypt=True;T



$(sql_user)

Password * $(sql_password)

Deployment Package

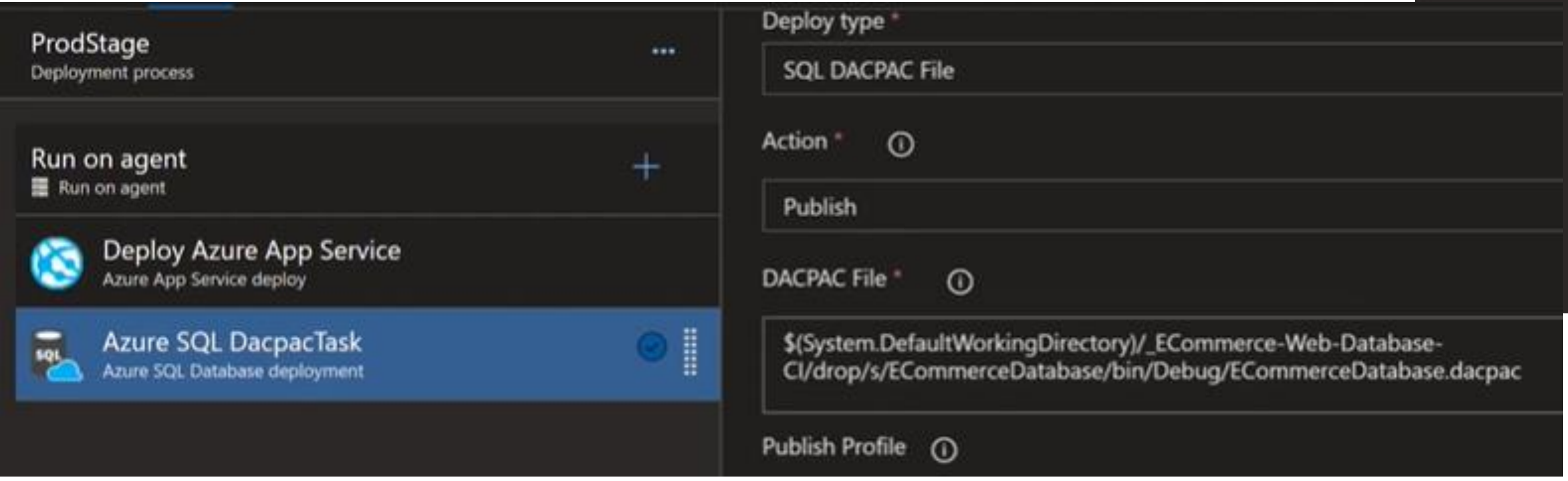
Deploy type * SQL DACPAC File

Action * Publish

DACPAC File *

This setting is required.

Publish Profile



Deploy type * SQL DACPAC File

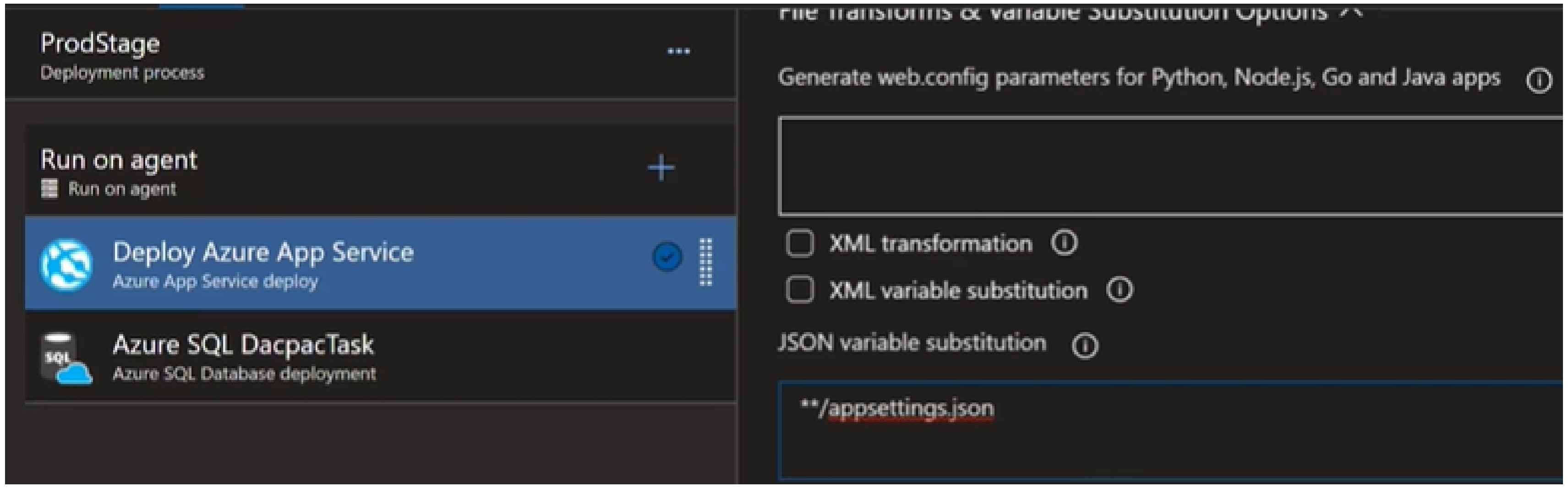
Action * Publish

DACPAC File * $(System.DefaultWorkingDirectory)/_ECommerce-Web-Database-CI/drop/s/ECommerceDatabase/bin/Debug/ECommerceDatabase.dacpac

Publish Profile

DotNetTricks

Database Pipeline: Replacing DB Connection


File Transforms & Variable Substitution Options

ProdStage
Deployment process

Run on agent
Run on agent

Deploy Azure App Service
Azure App Service deploy

Azure SQL DacpacTask
Azure SQL Database deployment

Generate web.config parameters for Python, Node.js, Go and Java apps

- [ ] XML transformation
- [ ] XML variable substitution

JSON variable substitution

```
**/appsettings.json
```

DotNetTricks