AZ-400 AZURE DEVOPS

Saleh Elnaggar

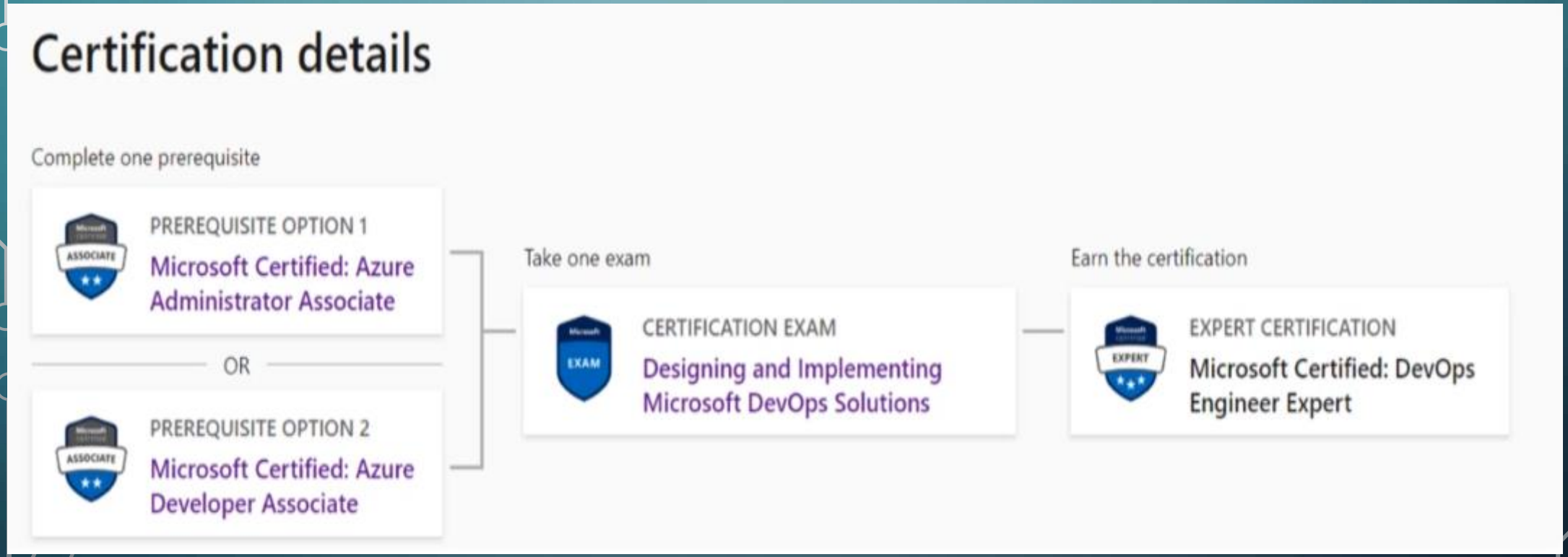
Azure DevOps

linkedin.com/in/saleh-elnaggar
salehelnaggar.live
saleh.elnaggar@gmail.com

AZ-400 AZURE DEVOPS

- COURSE STRUCTURED
- WHAT IS DEVOPS?
- SOURCE CONTROL AND VERSION CONTROL
- CONTINUOUS INTEGRATION
- CONTINUOUS DELIVERY
- IMPLEMENTING INFRASTRUCTURE

About the certificate



Pre-requisite “requirement”

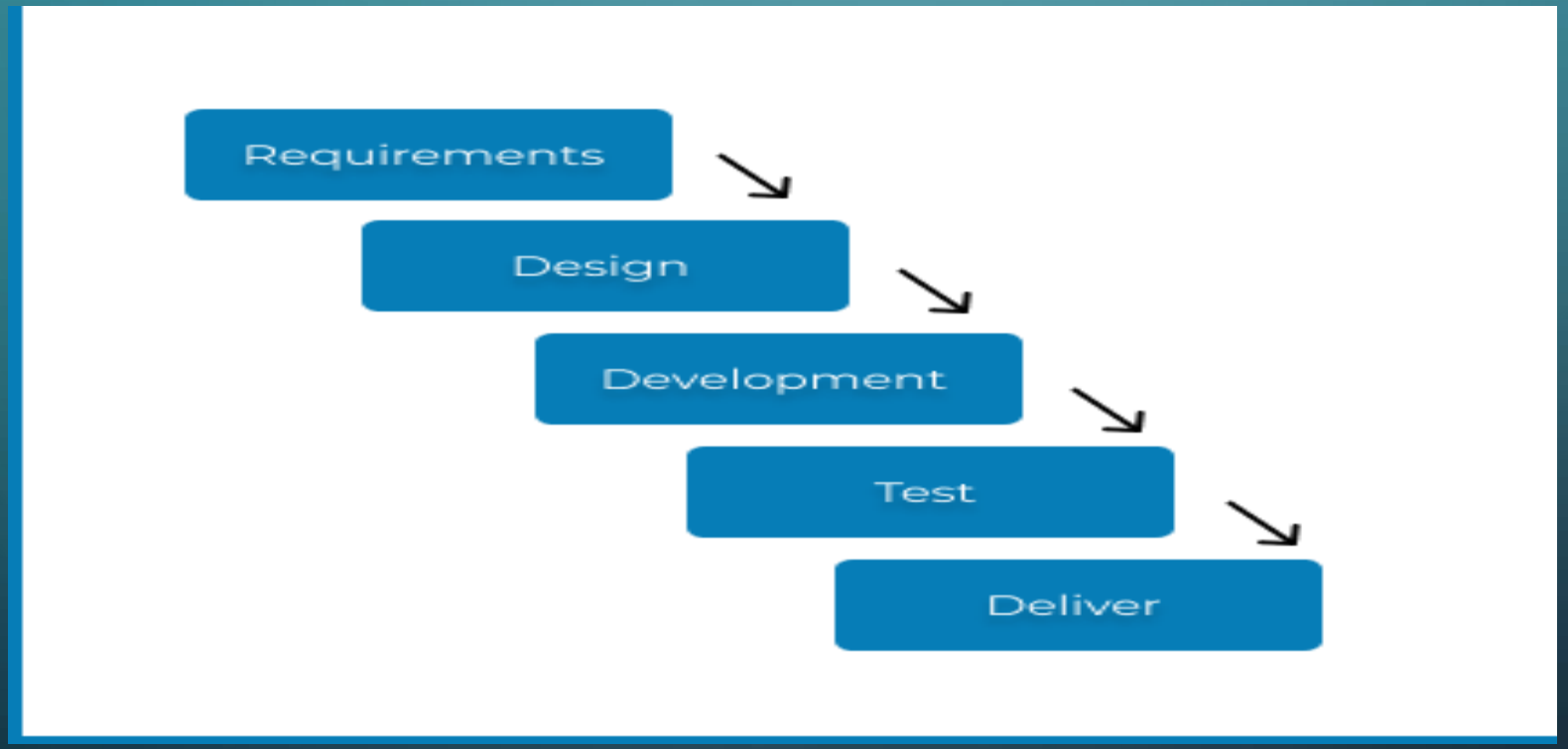
- Familiar with Azure common services “Azure vm, vmss, azure wep apps”
- Simple knowledge on any development framework
- Knowledge on how applications are deployed

What do you need?

- An Azure account with Azure subscription
- An Azure DevOps account – easy to create

Before DevOps concept

- Traditional Project lifecycle

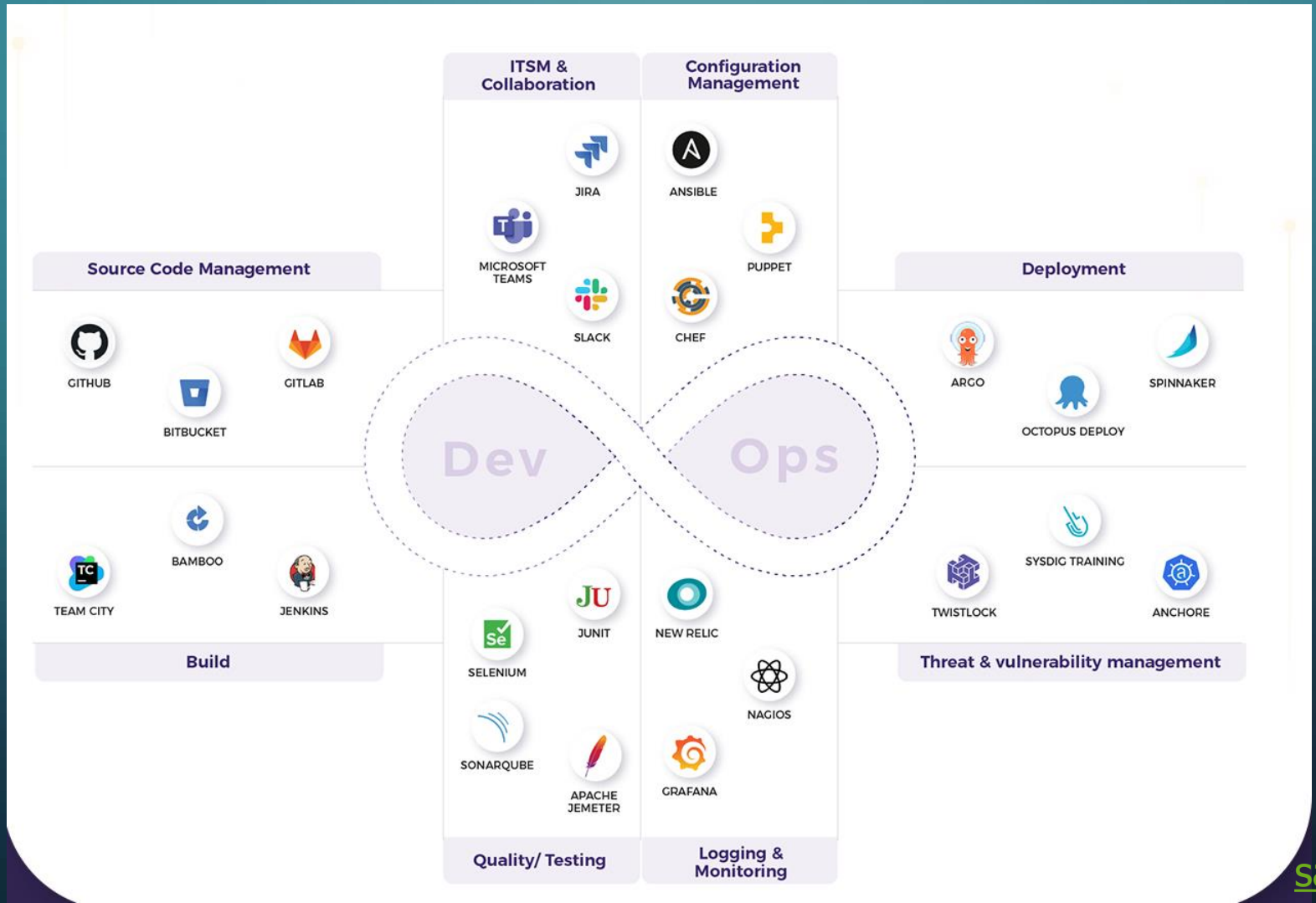


Why DevOps?

- Agile Project lifecycle

Why DevOps?

- DevOps tools



Azure DevOps tools

Azure Boards

Plan, track, and discuss work across teams, deliver value to your users faster.

Azure Repos

Unlimited cloud-hosted private Git repos. Collaborative pull requests, advanced file management, and more.

Azure Pipelines

CI/CD that works with any language, platform, and cloud. Connect to GitHub or any Git provider and deploy continuously to any cloud.

Azure Test Plans

The test management and exploratory testing toolkit that lets you ship with confidence.

Azure Artifacts

Create, host, and share packages. Easily add artifacts to CI/CD pipelines.

Azure DevOps services pricing

Basic Plan

First 5 users free,
then $6 per user per month

Start free

- **Azure Pipelines:** Includes the free offer from INDIVIDUAL SERVICES
- **Azure Boards:** Work item tracking and Kanban boards
- **Azure Repos:** Unlimited private Git repos
- **Azure Artifacts:** 2 GiB free per organization

First 5 users free

Basic + Test Plans

$52 per user
per month

30 day free trial

- Includes all Basic plan features
- Test planning, tracking & execution
- Browser-based tests with annotation
- Rich-client test execution
- User acceptance testing
- Centralized reporting

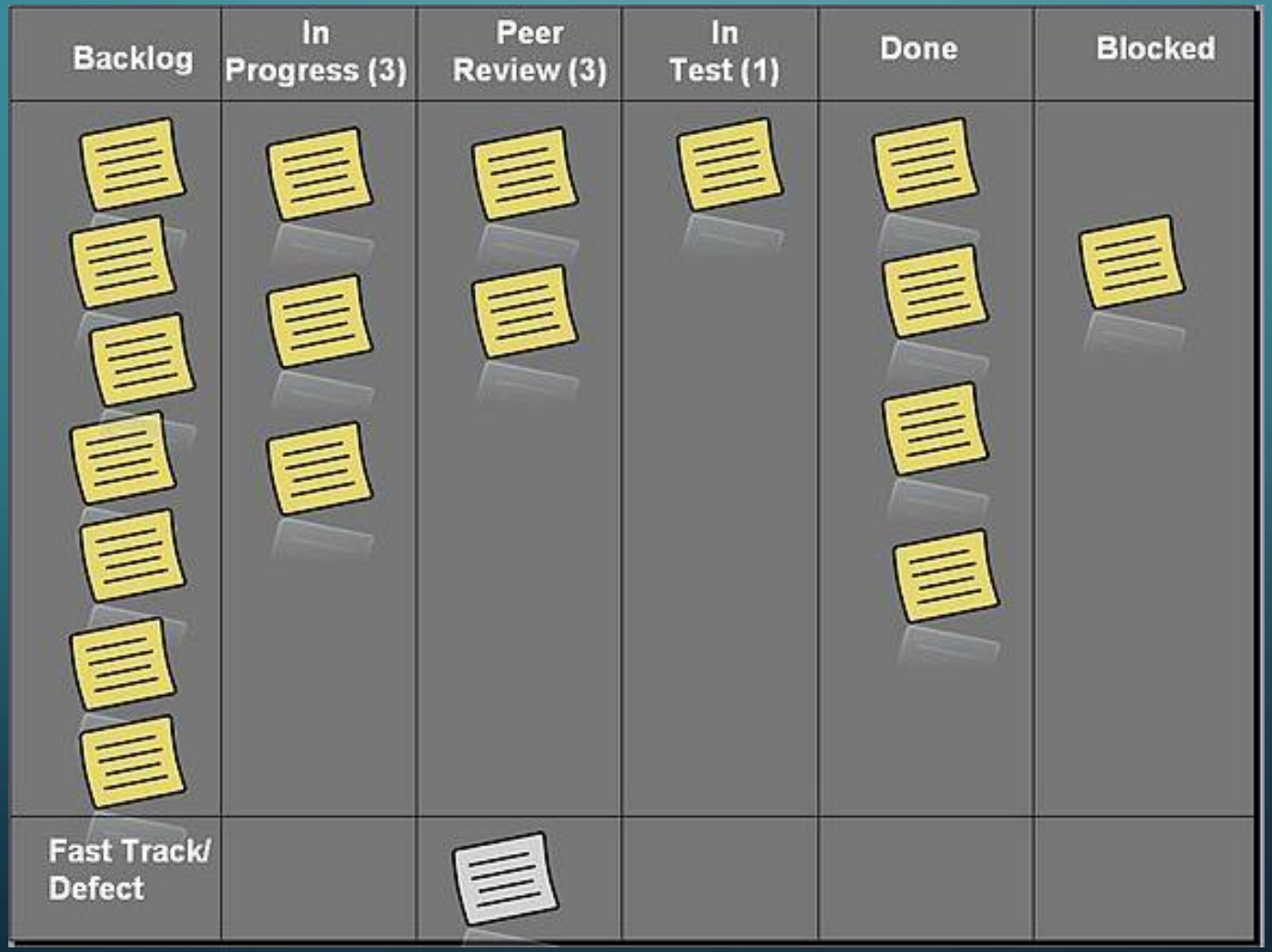
more details

salehelnaggar.live

Azure Boards

Azure Boards

- Sprint

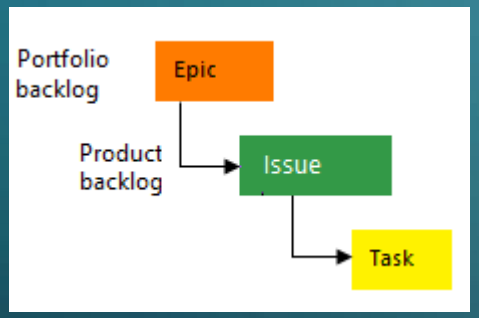


Azure Boards

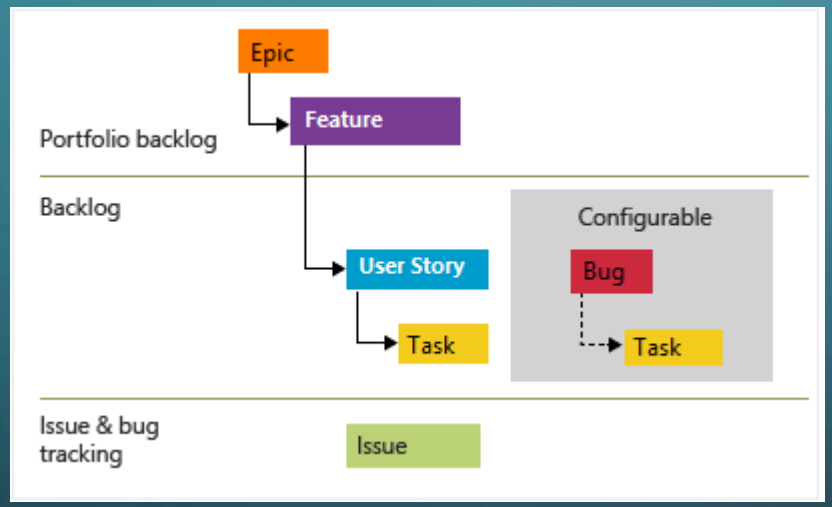
Demo lab

Azure Boards – Project types

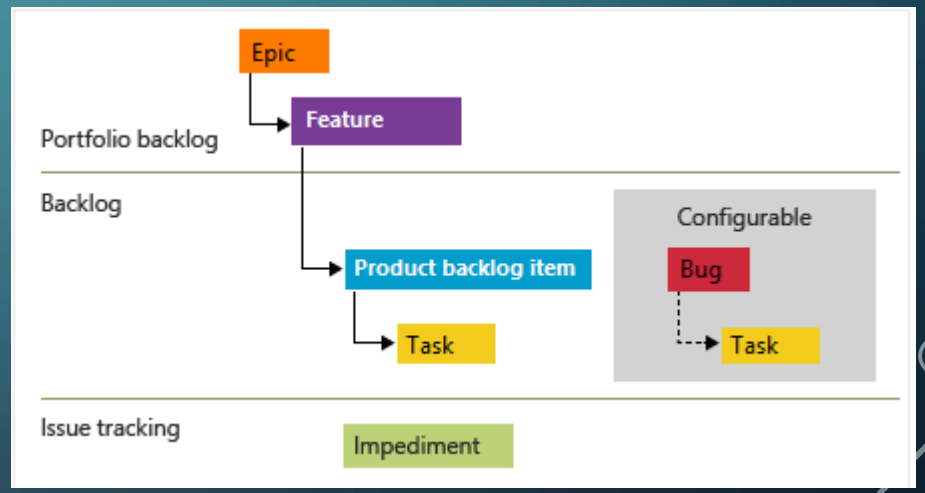
Basic



Agile



Scrum



more details

Azure Boards – use sprint

Demo lab

Azure Boards – integration with slack

Demo lab

Azure Boards – Azure AD integration

Demo lab

Azure Boards – add users to project

Demo lab

Azure Boards – different charts

Burndown
Displays burndown across multiple teams and multiple sprints. Create a release burndown or bug burndown.

Focus on the remaining work within the specified period of time

Burnup
Displays burnup across multiple teams and multiple sprints. Create a release burnup or bug burnup.

Focuses on the completed work

Chart for Work Items
Visualize work items like bugs, user stories, and features using shared work item queries.

Are we on track to complete the set of work by the end date

Cumulative Flow Diagram (CFD)
Visualize the flow of work and identify bottlenecks in the software development process.

This helps to see the items as they move through the different states

Azure Boards – different charts

Cycle Time
Visualize and analyze your team's cycle time using a control chart.

Measures the time taken for the team to complete work items once they have begun actively working on them

Lead Time
Visualize and analyze your team's lead time using a control chart.

Measures the total time elapsed from the creation of work items to theire completion

Source code tool – version control

- What is Git?
- Azure repos

Source code tool – version control

- Version control categories:
  - Centralized system
    - Subversion control
    - Team foundation
  - Decentralized system
    - git

Source code tool – version control

- Version control categories:
  - Centralized system
    - Subversion control
    - Team foundation

Source code tool – version control

- Decentralized system
  - git

Git

Demo lab

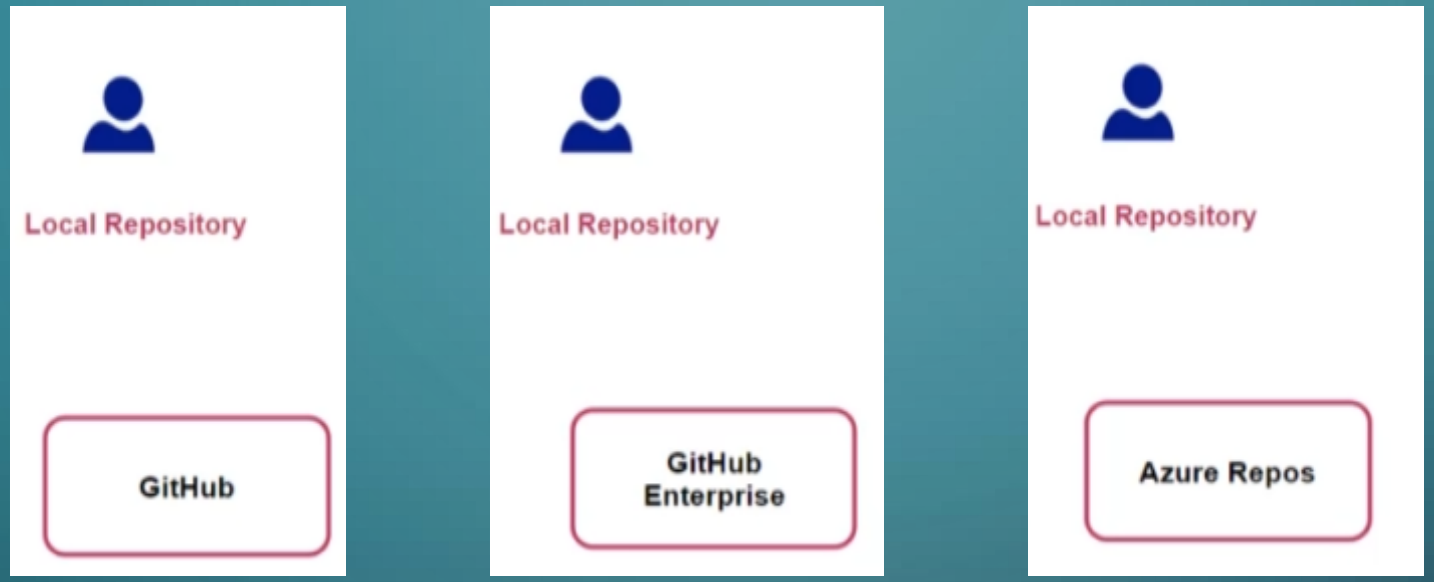
1. Install git.
2. Initialize an empty repository.
3. Playing with git locally.

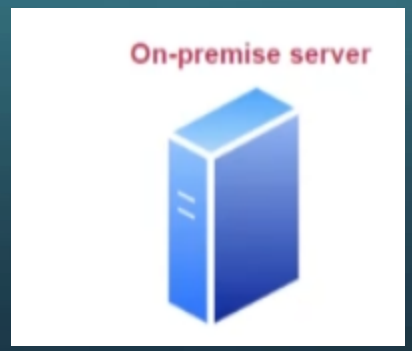
Git

Demo lab

1. Making changes to your files
2. Go back to previous commit.

Central git repository





Using GitHub

Demo lab

1. Create new repo
2. Add remote repo to local repo.

Using GitHub

Demo lab

1. Make changes on repo locally
2. See the different pointers.
3. Check it in the remote repo.

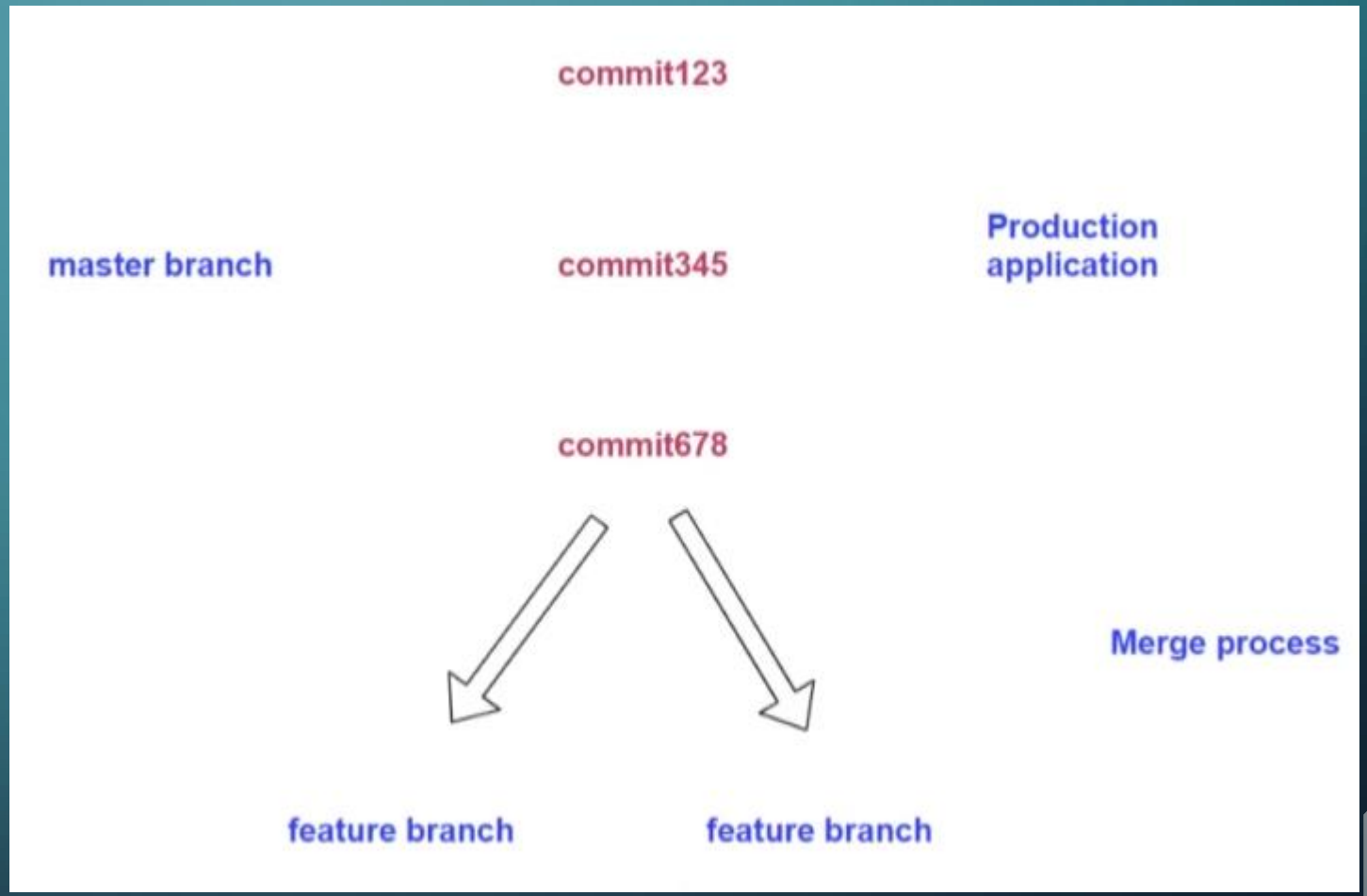
Azure Repos

Demo lab

1. Check the default repo.
2. Create new repo.
3. Add Azure repo to local repo.
4. Make changes locally and push it.

Understanding branches

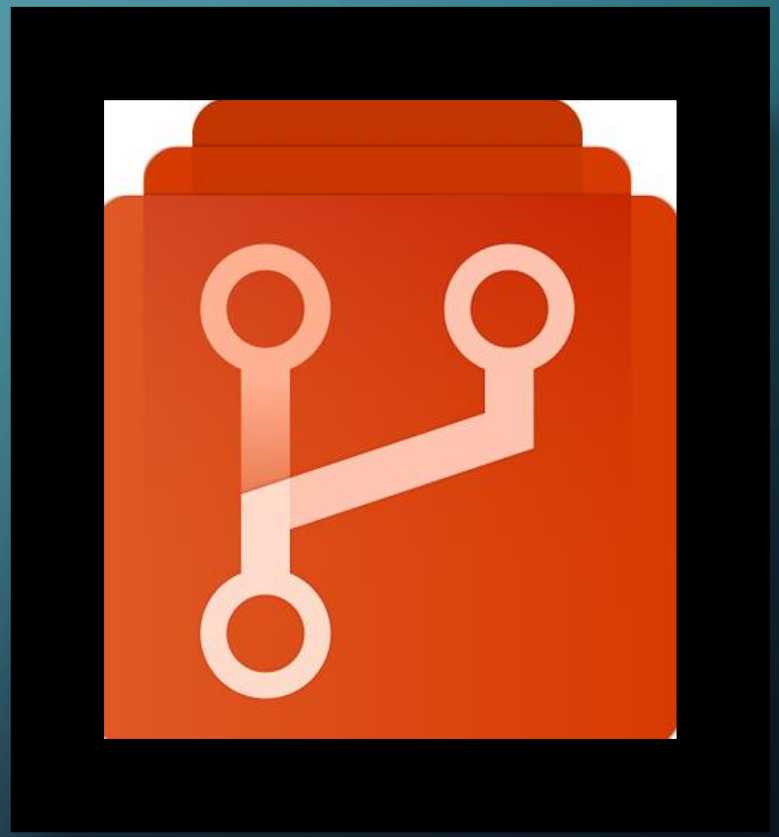
- Good practices:
  - Create many short feature branches
  - Delete once they are not required.



Branches

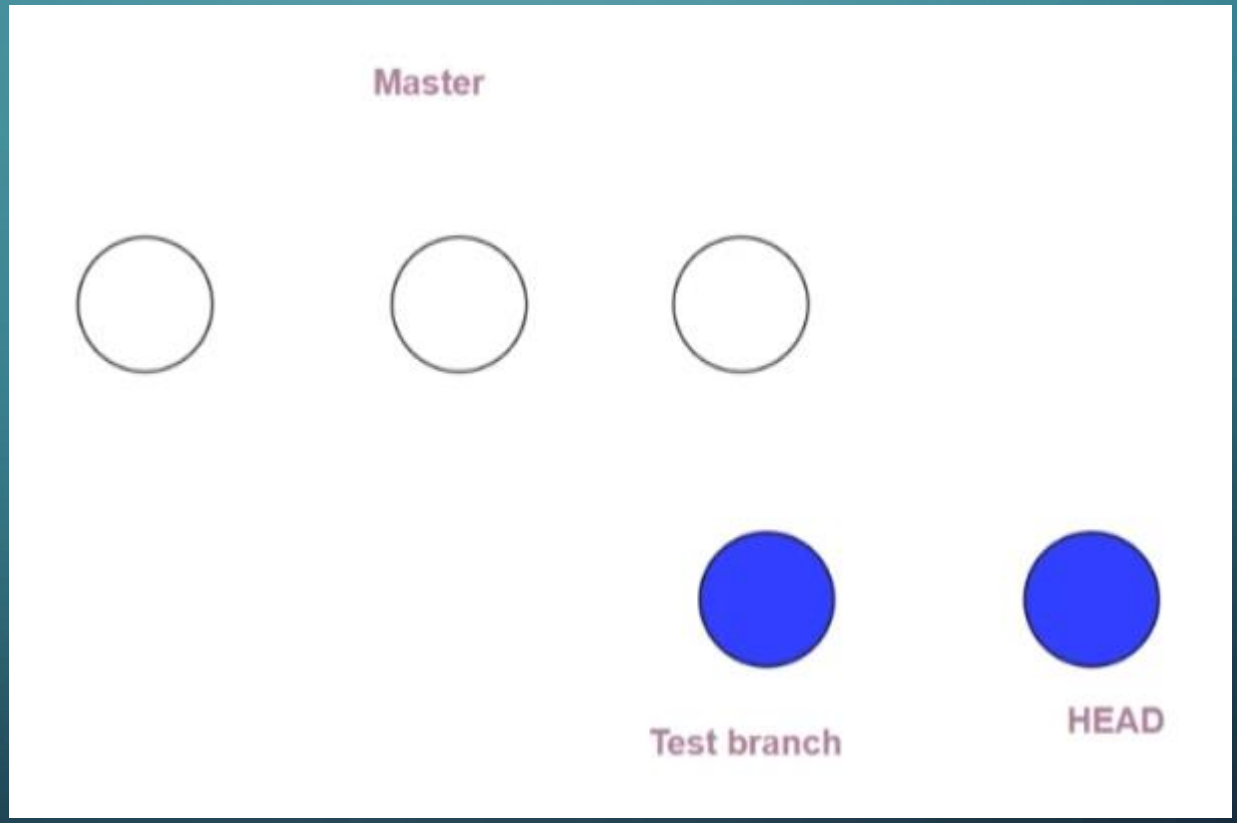
Demo lab

1. Show all branches.
2. Create new branch.
3. Work with the new branch.



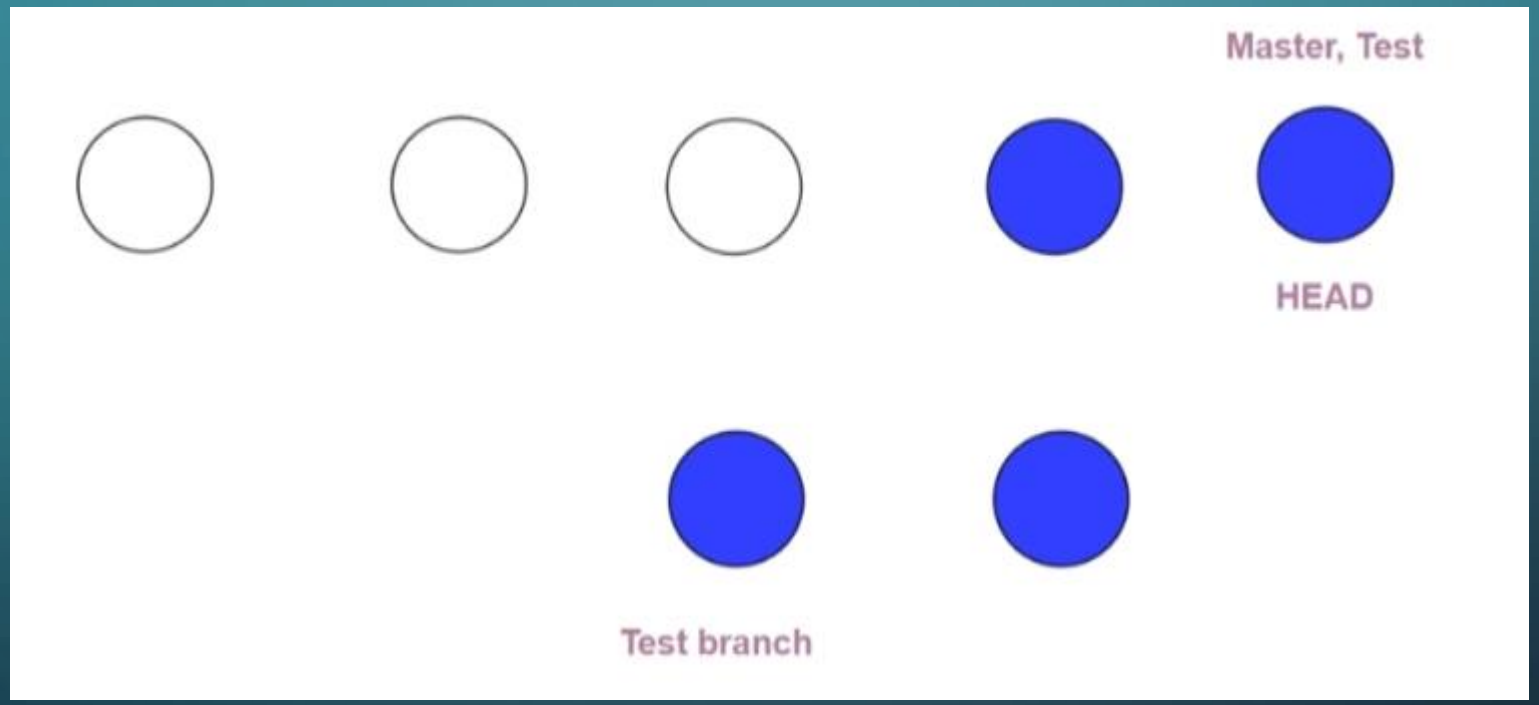
Merges in git

Implicit Merge "Fast forward merge"



Merges in git

Implicit Merge "Fast forward merge"



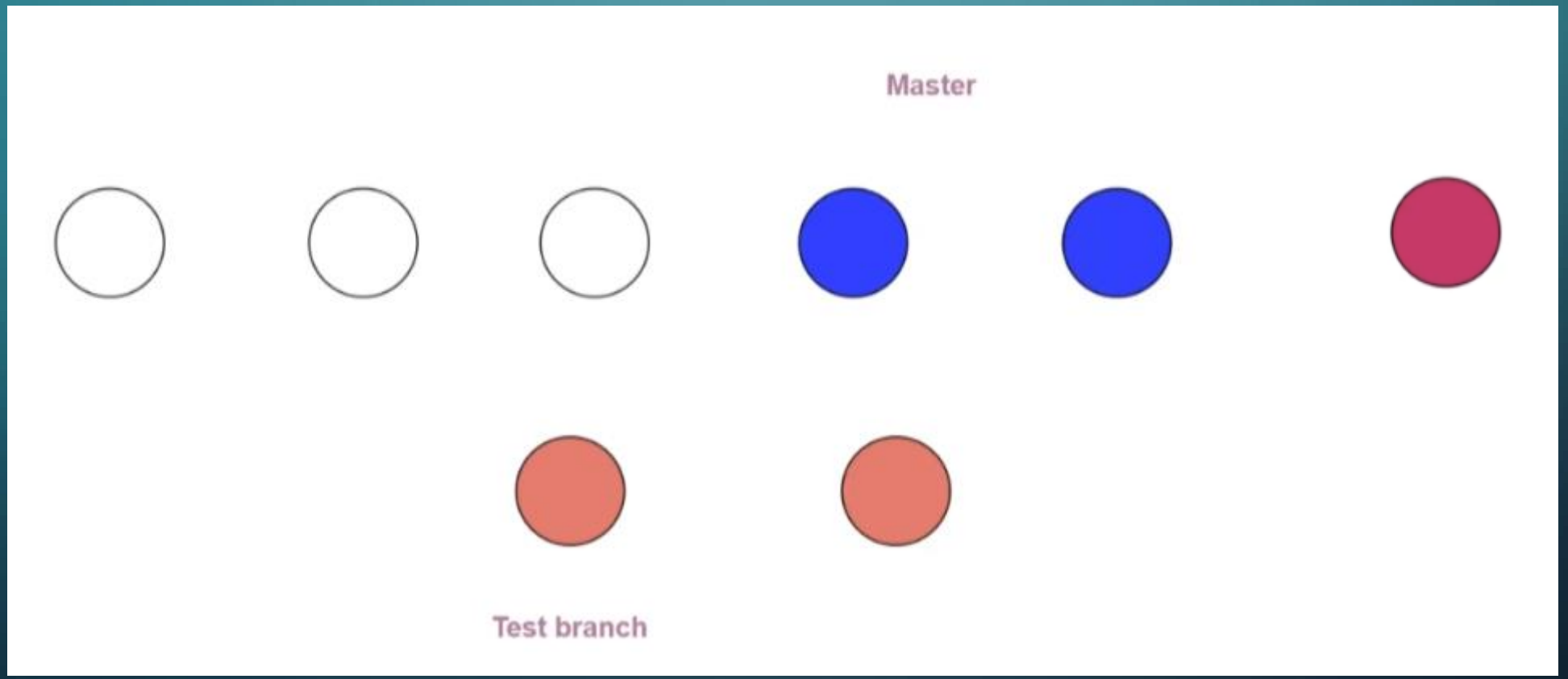
Fast forward merge

Demo lab

1. Do fast forward merge
2. Check the pointer

Merges in git

Recursive merge "3-way merge"



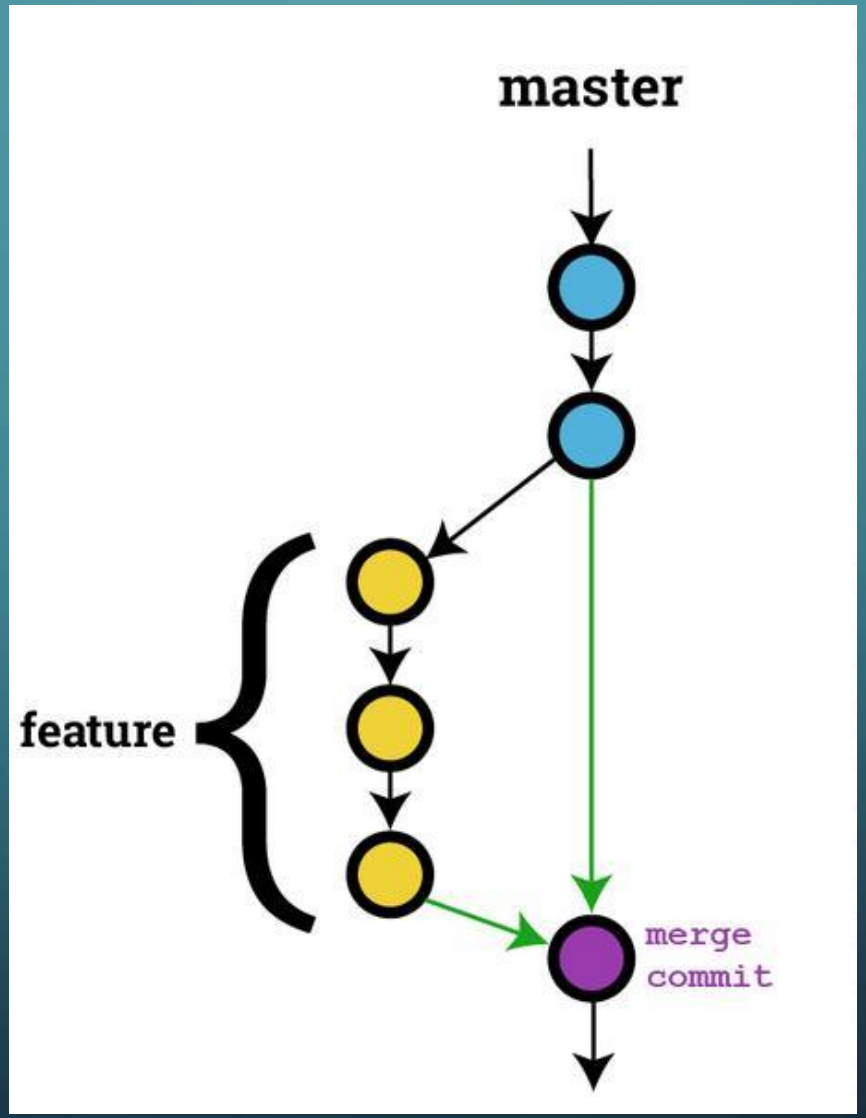
Recursive merge

Demo lab

1. Make changes in the main.
2. Add new file to feature branches.
3. Try to do merge “recursive merge”.

Merges in git

Squash merge



Squash merge

Demo lab

1. Make more than two commit in the feature.
2. Try to do merge “squash merge”.

Conflict in the merge

Demo lab

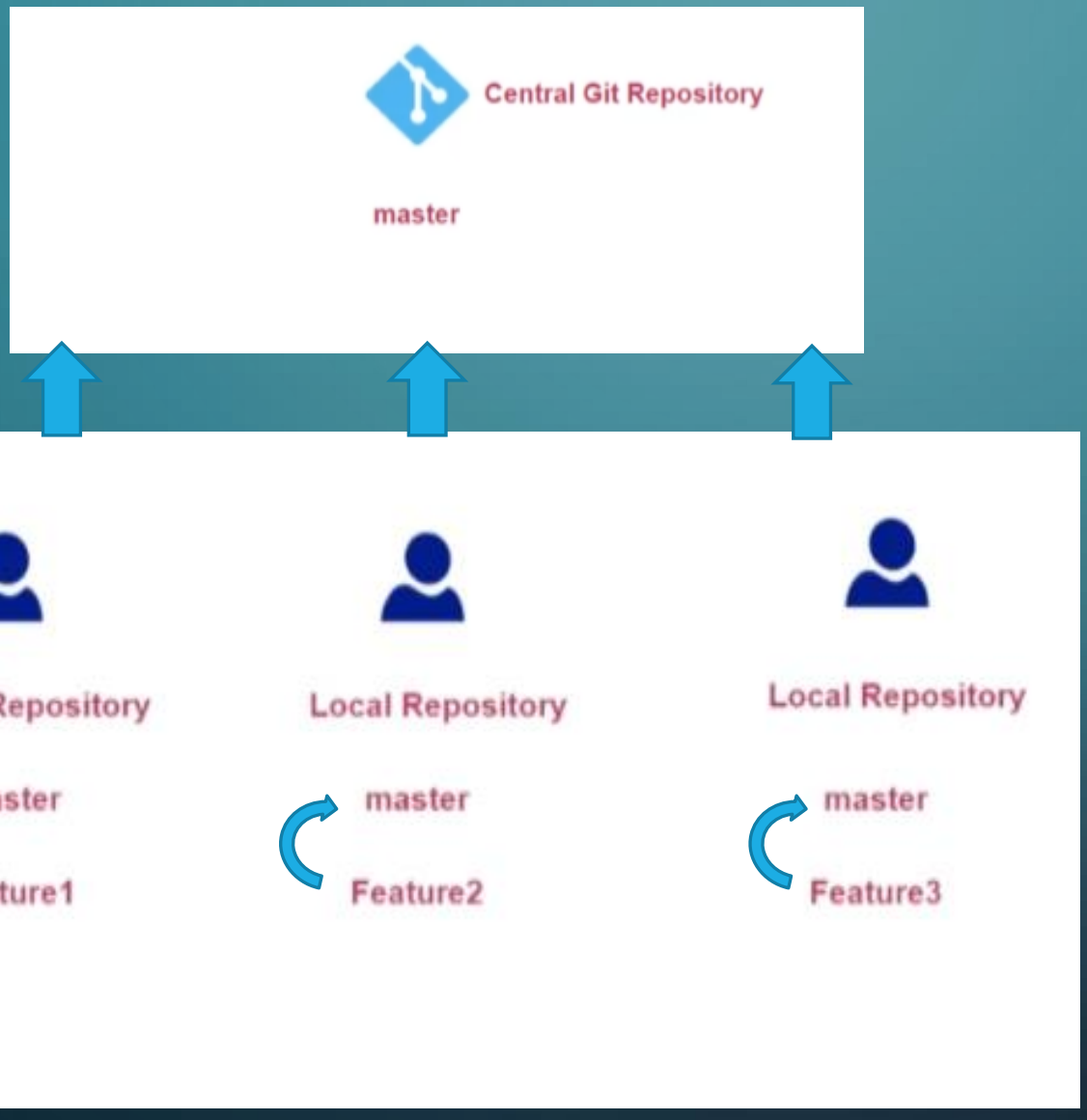
1. Make changes in the main.
2. Make changes in the feature.
3. Try to make merge.
4. Solve the conflict.
5. How to avoid that in the real live?

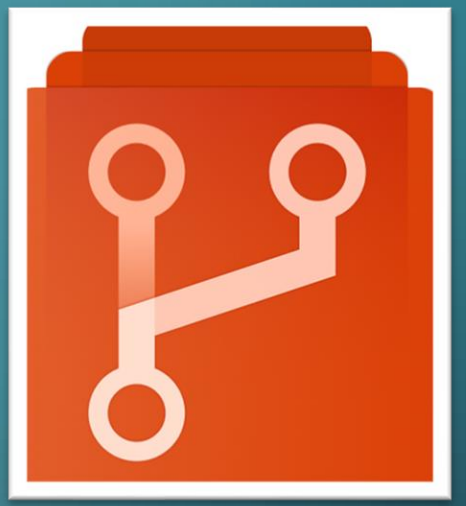
Pushing branches

Demo lab

1. Check the repo branches
2. Use the git push command “push the main”
3. Check again your branches.
4. Push the new branch also.
    1. git push --all origin
    2. git push --u origin feature

Pull requests





Pull request

Demo lab

1. Enable any of branch policies
2. Make changes in the new branch
3. Merge to main locally
4. Try to push to main branch
5. Go to pull requests
    1. Create new pull request
    2. Approve and complete reviewing the changes

Pulling changes from the repo

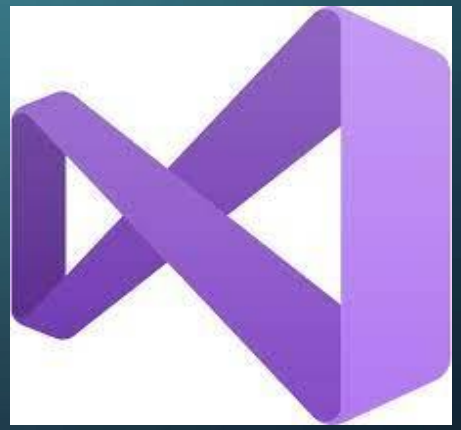
Demo lab

1. Make change in the remote repo.
2. If the developer in the local git try to push after some modification in the same file!
3. His push will rejected.
4. Need to make pull first from the remote repo.

GitHub with Visual Studio

Demo lab

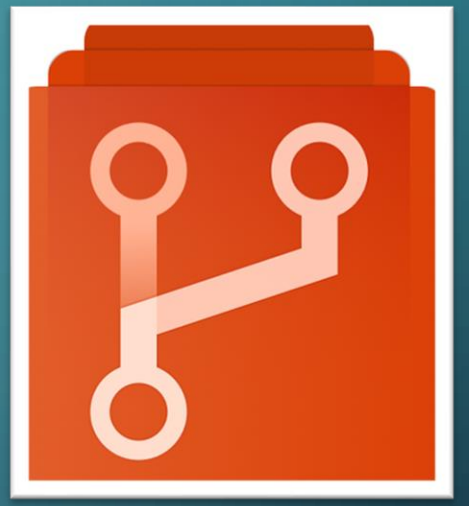
1. Create new .NET core project.
2. Make sure that the source control in your VS is git.
3. Create new GitHub repo.
4. Make changes on the code and commit and push it.
5. Check the changes on GitHub.

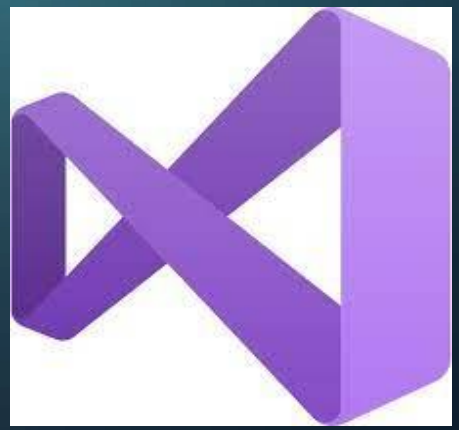


Azure repos with Visual Studio

Demo lab

1. Create new .NET core project.
2. Make sure that the source control in your VS is git
3. Add Azure repo to your project.
4. Push the code to Azure repo.
5. Make changes and check it remotely.





Git – .gitignore file

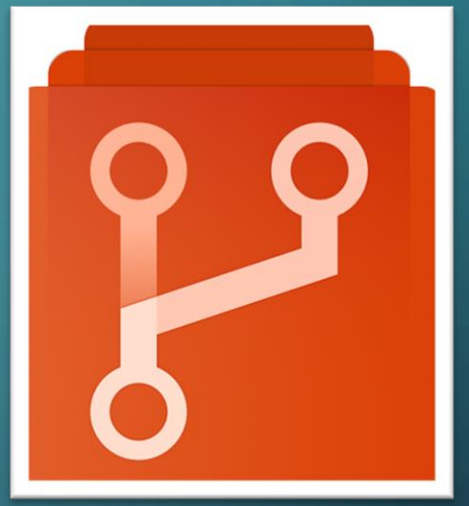
Demo lab

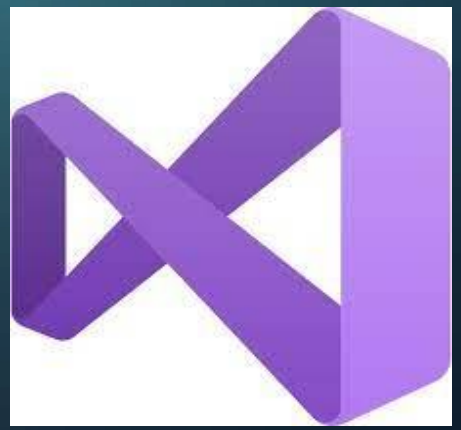
1. Make a new folder to try gitignore.
2. Create more than 3 files
3. Add which files you need to .gitignore
    1. Manually
    2. echo command “echo yourFile >> .gitignore”

Team foundation version control with Visual Studio

Demo lab

1. Create new repo with TFVC
2. Manage connection to browse your repos
3. Choose your TFVC repo and map & Get
4. Add your project to source control.
5. Check in your project to the TFVC repo.
6. Check out for edit and check in again.

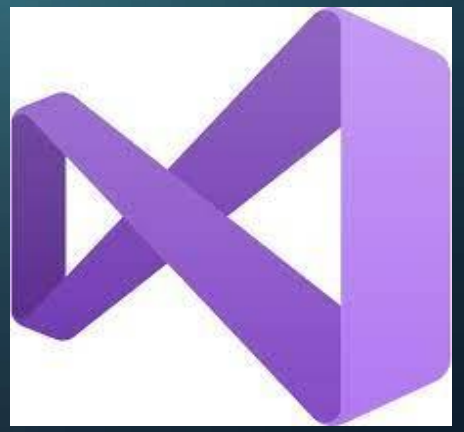




Team foundation version control with Visual Studio

Demo lab

1. Create new repo with TFVC
2. Manage connection to browse your repos
3. Choose your TFVC repo and map & Get
4. Add your project to source control.
5. Check in your project to the TFVC repo.
6. Check out for edit and check in again.

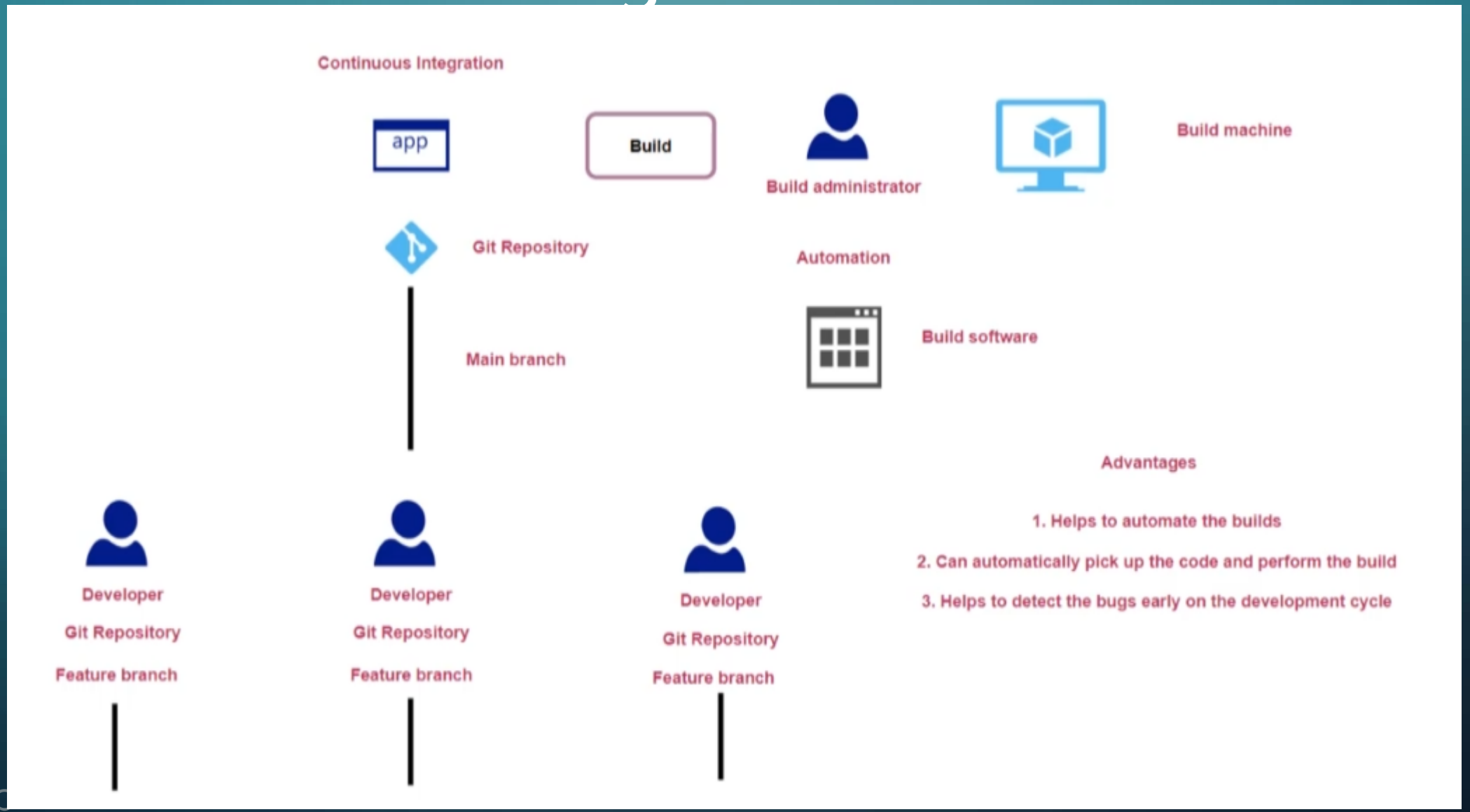


Integration GitHub with Azure Boards

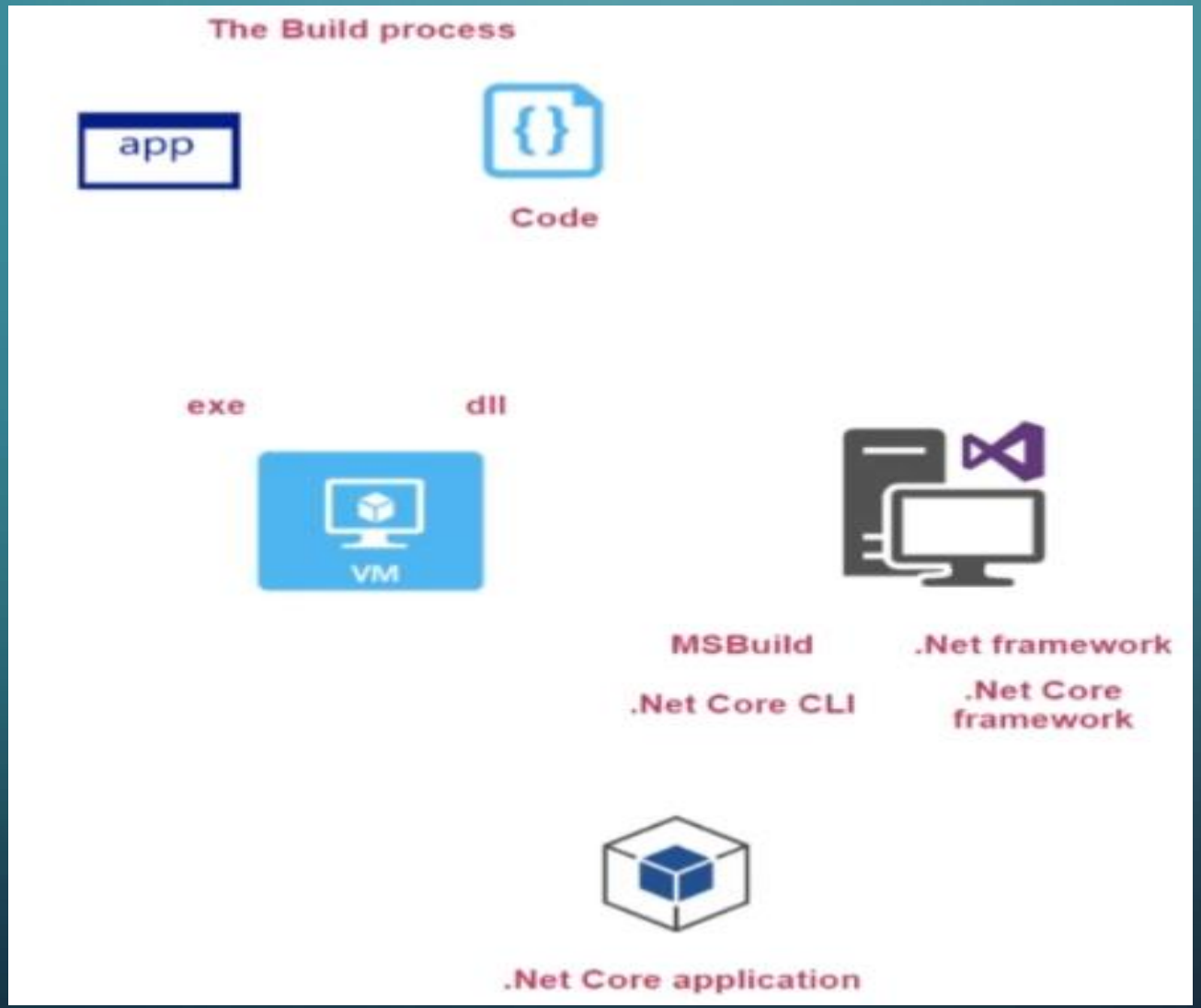
Demo lab

1. Make GitHub connection
2. Using the specific keyword which integrate with Azure Boards “Fixed AB#taskNumber”

Continuous integration



Build

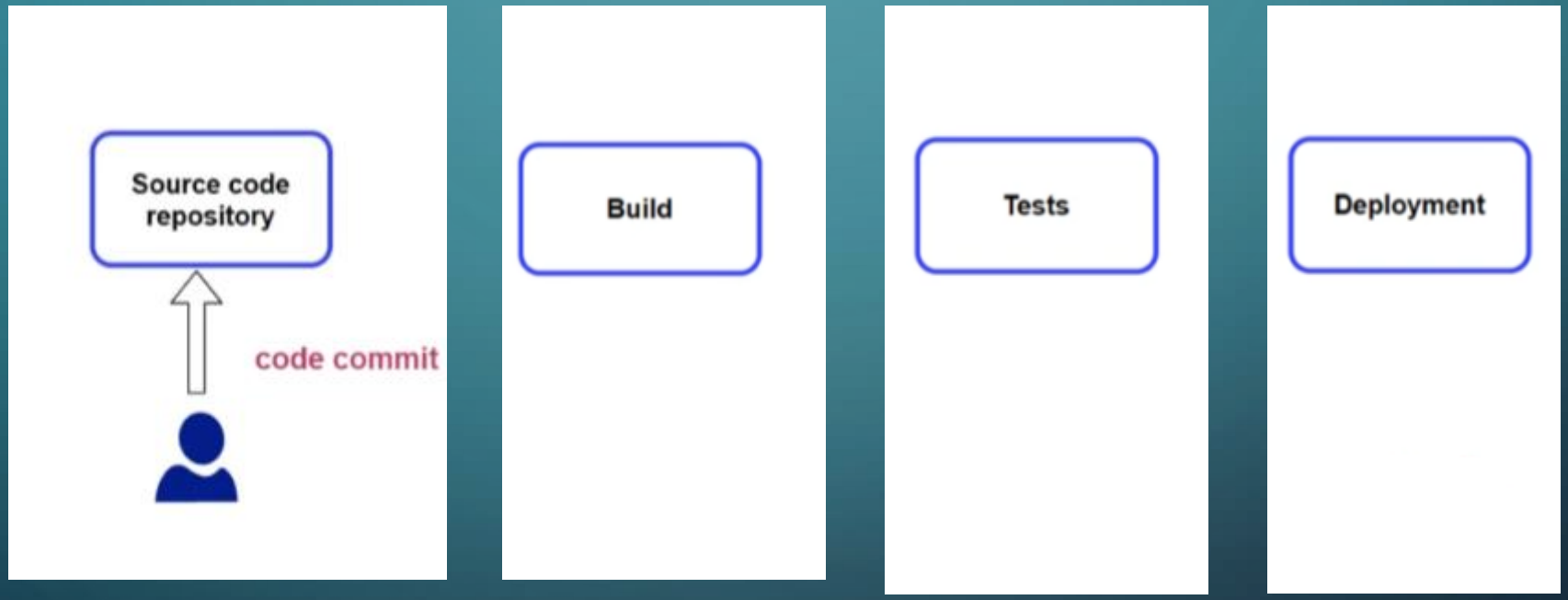

The Build process
app
{}
Code
exe
dll
VM
MSBuild
.Net framework
.Net Core CLI
.Net Core framework
.Net Core application

Build

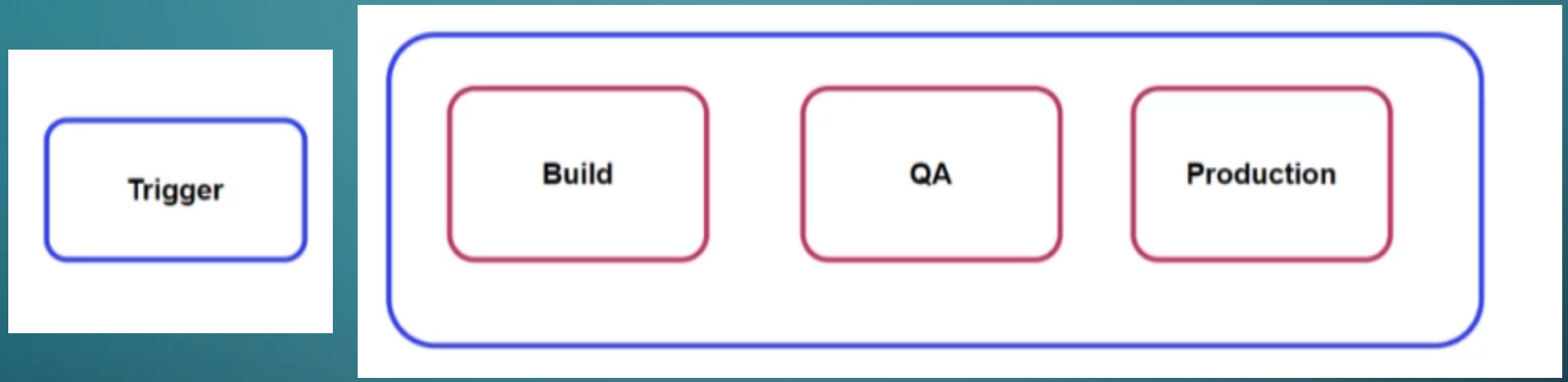
Demo lab

1. Check build files
2. Use Visual Studio for build
3. Run the app
4. Check again the build directory

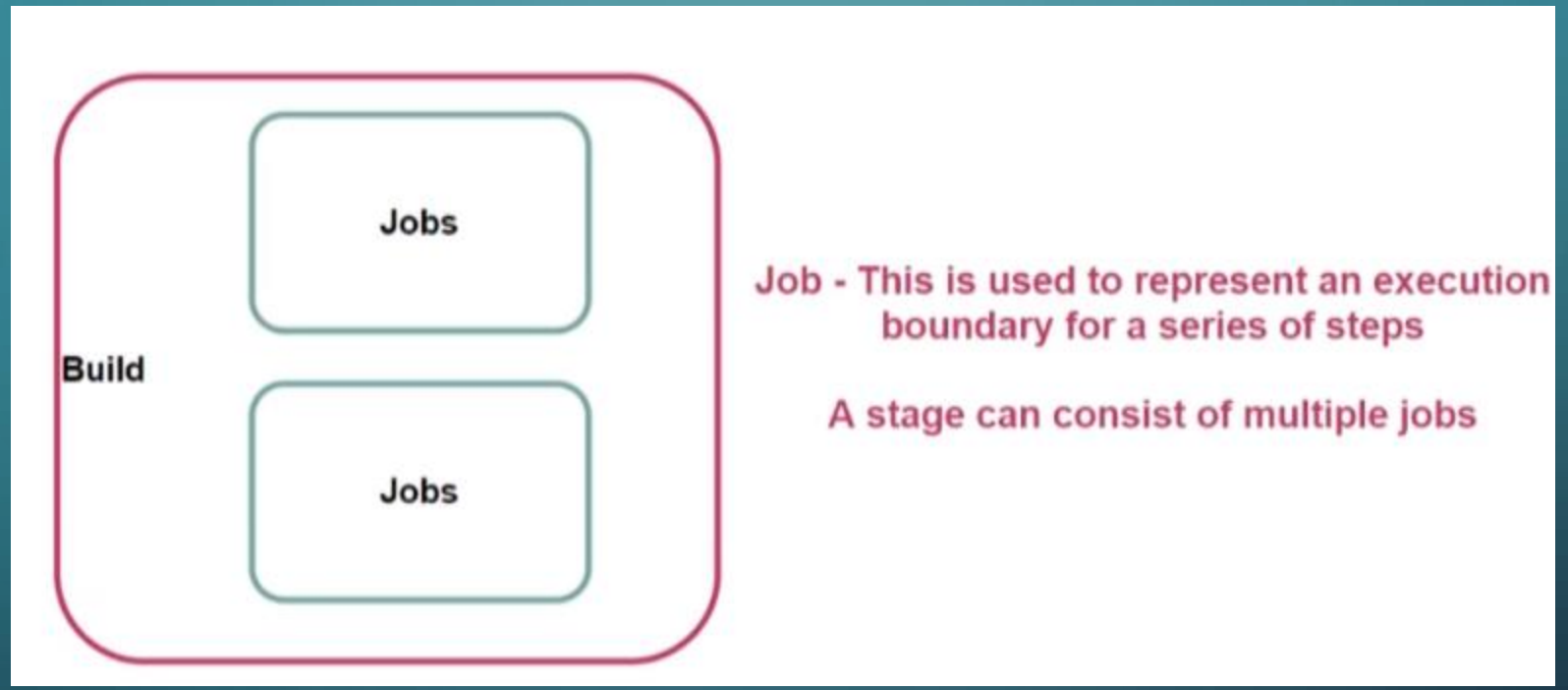
Continuous Integration/Continuous Delivery CI/CD



Azure Pipelines



Azure Pipelines - stage



Azure Pipelines - Build

Azure Pipelines

Demo lab

1. Create simple pipeline
2. Check the trigger
3. Edit the pipeline and use multiple jobs

Azure Pipelines – build .NET Core application

Demo lab

1. Create new repo for app code
2. Push my code to the repo
3. Create new Azure pipeline
4. Check the trigger

More details

Azure Pipelines – GitHub

Demo lab

1. Create new repo for app code
2. Push my code to the repo
3. Create new Azure pipeline and choose to pick your code form GitHub

Microsoft hosted agent

Demo lab

1. Microsoft hosted agent details
    1. Managed service by Azure pipelines
    2. Check various images and it's labels

Self hosted agent

Demo lab

1. Create Azure VM
2. Install git, NuGet, Visual Studio and .NET Core for the build
3. Azure pipelines agent – download agent and run it
4. Use the self agent to build the .NET Core app

Quick note

- After creating your first pipeline, you have the yaml file located in your repo, and this yaml file didn't exist in your local repo, so if you made any changes on your local machine and try to push it in the repo, you will have an issue because you didn't have the latest version from your remote repo.

Jenkins
another Continues Integration tools

Demo lab

1. Create Azure VM
2. Install Java development kit
3. Install Jenkins
4. Log on as a service

Jenkins
Build .NET Core app

Demo lab

1. Install build tools “git, NuGet, visual studio, .NET Core SKD 3.1”
2. Install MSBuild in the Jenkins server
3. Set global tool configuration
    1. Git location
    2. MSBuild
4. Copy the home directory path from configure system

Jenkins
Build .NET Core app

Demo lab

1. Add new item to build the app
2. Choose git as a source control
3. Add build step “windows batch command”
4. Add build a visual studio project

Jenkins
Build .NET Core app

Demo lab

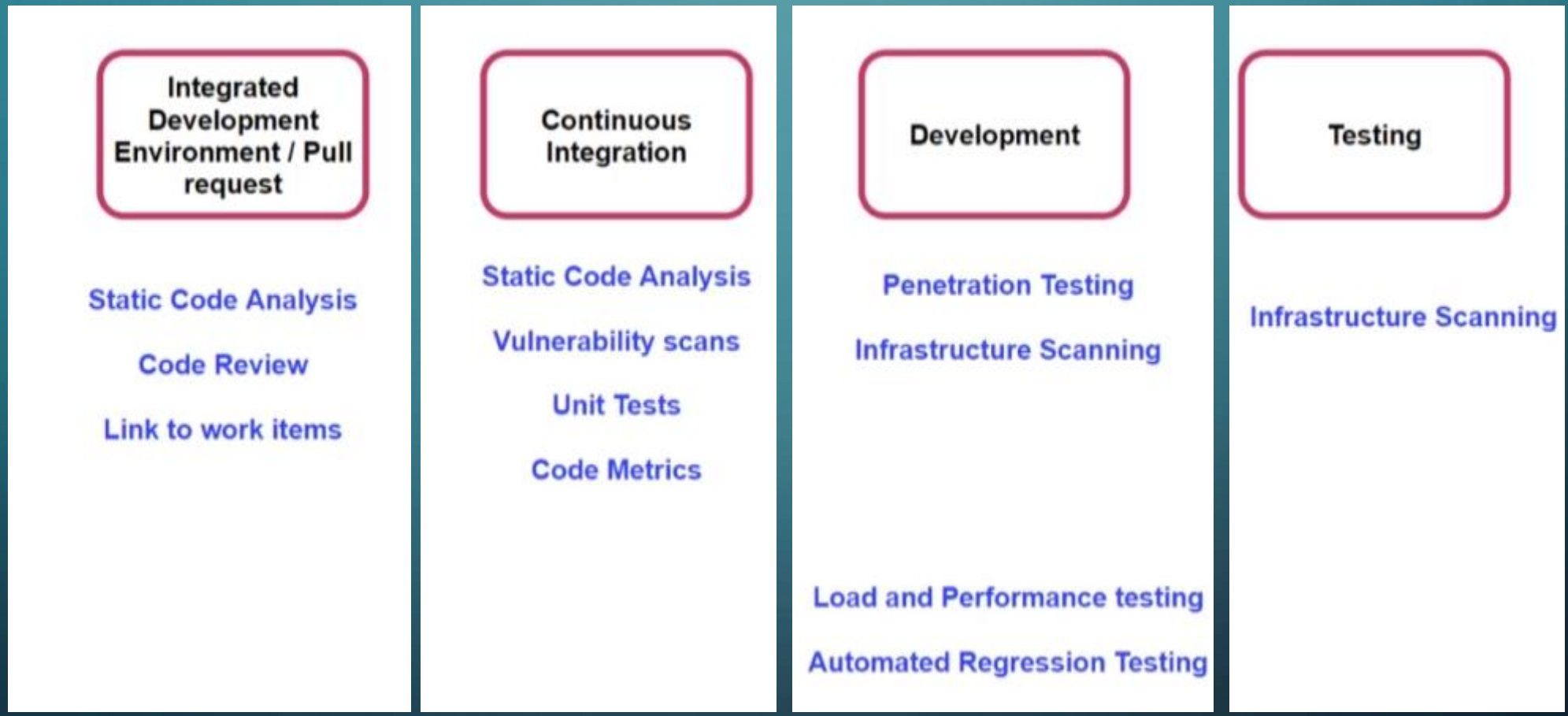
1. Deploy the same build but using Azure repos

Jenkins CI

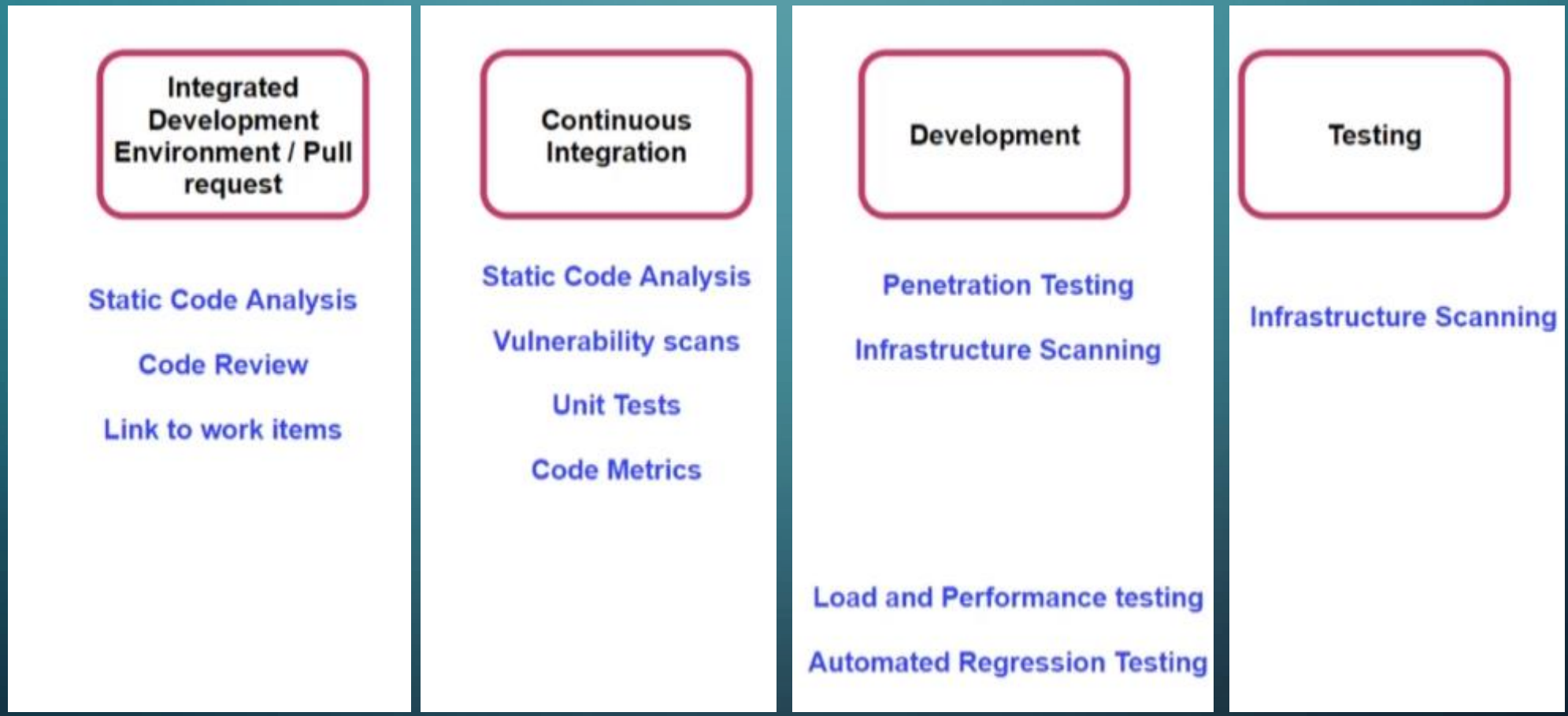
Demo lab

1. Make continuous Integration with Jenkins
    1. Add security group rule to open port 8080
    2. Open port 8080 in windows firewall
    3. Create a Jenkins token (people/configure)
    4. Add Jenkins token to the project service hooks
    5. Make a simple change to the code to check

Security in the CI/CD pipeline



Security in the CI/CD pipeline



Static Code Analysis

Demo lab

1. Analysis code by your VS
2. Add packages for analysis
    1. https://api.nuget.org/v3/index.json
3. Try to make a build by the pipeline

Using WhiteSource Bolt

Demo lab

1. Add whiteSource in the organization extensions
2. Check your pipeline section you will have WhiteSource in place
3. Open any previous pipeline which have .NET Core code
4. Add whiteSource task to your pipeline

Unit testing

Demo lab

1. Run the Unit test on VS.
2. Make a new repo and push your code.
3. Create new pipeline and add test unit task.

Code Coverage

Demo lab

1. We'll use the same last code in unit test.
2. In the unit test project add NuGet package "coverage.msbuild"
3. Create new repo and push your code
4. Create new build pipeline

sonarQube

Demo lab

1. You can use this tool by download it in vm and implement sonarQube or You can use it in the cloud “sonarCloud”
2. Login with your Azure devops account
3. Create new organization and project
4. Create new .NET Core project and new repo
5. Add sonarQube to organization extensions
6. Add new project service connection to connect with this instance
7. Build your pipeline
    1. Before the build task you can add Prepare analysis configuration task
    2. After the build you can add Run code Analysis and publish the result tasks

Technical debt

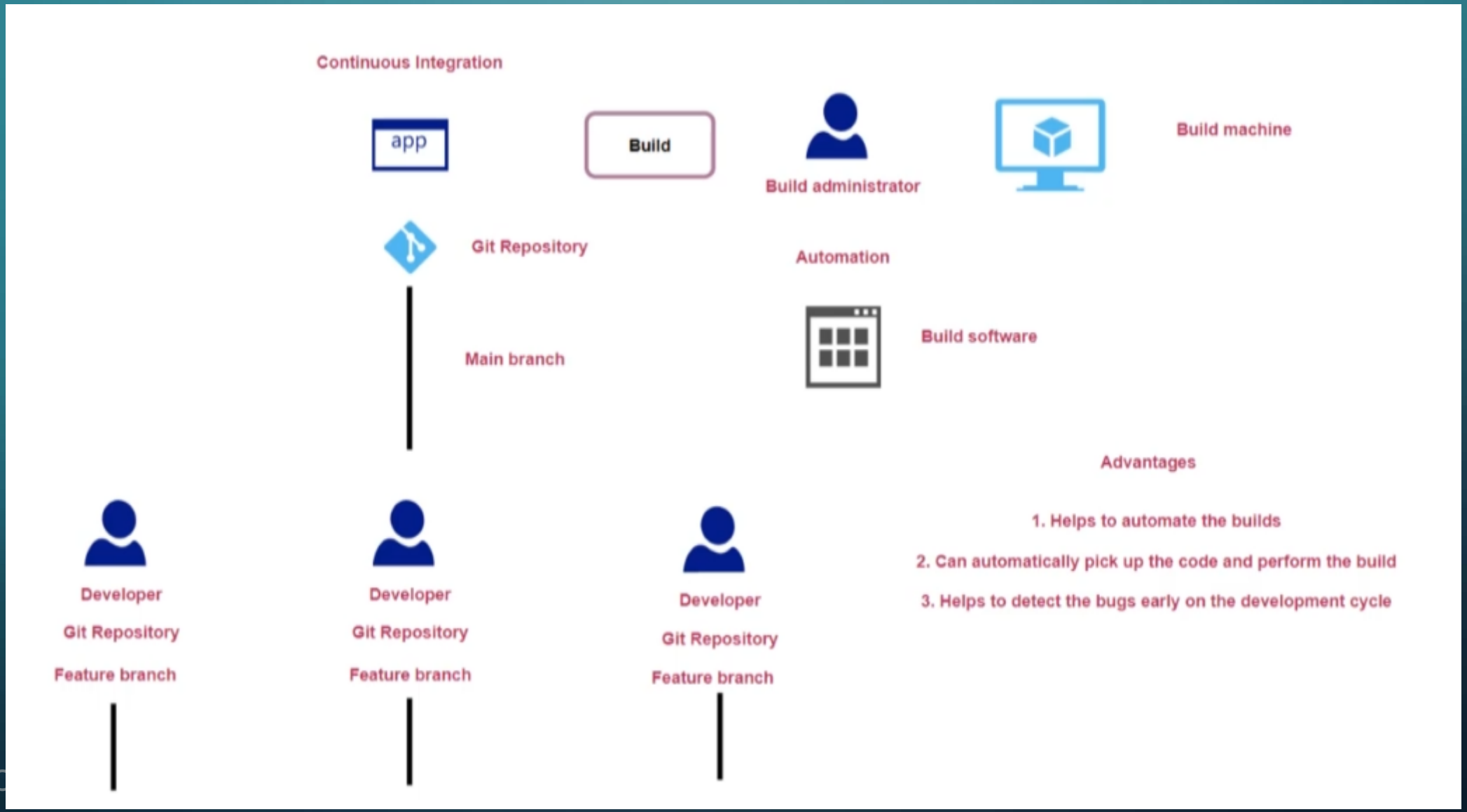
1. Cost of rework caused by choosing easy solution instead of using better approach that would take longer
2. This could be anything that could slow or hinder the entire development process
3. As the technical debt increases, it can become more difficult to make changes to code
4. sonarCloud can calculate technical debt

Classic editor

Demo lab

1. Create new pipeline but choose classic editor to create your pipeline
2. Build our .NET Core by the classic editor

Benefits of CI

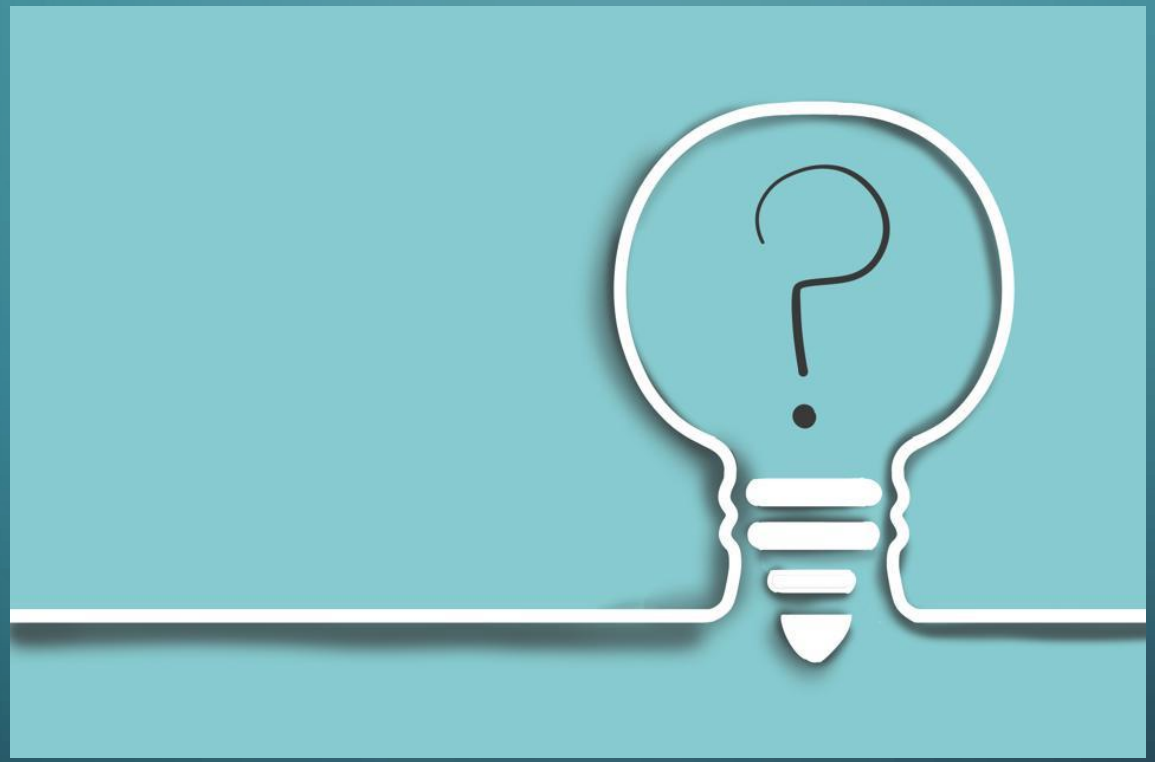


Parallel jobs

Demo lab

1. Check parallel jobs option in organization settings
2. If you need to purchase parallel job you should add your Azure subscription to do that.

Questions



Infrastructure – Azure Web App

Demo lab

- Create Azure Web App
- Publish a web app from visual studio

Infrastructure – Azure VM

Demo lab

- Create Azure VM
- Add IIS role and ensure that the management service installed also If you need to publish directly from VS
- Open the IIS manager to enable management service remote connection
- Install packages and framework for your app
  - .NET Core 3.1
  - Web deploy 3.6
- Configure DNS for the VM
- Publish a web app from visual studio

Infrastructure – Custom script extensions

Demo lab

- Create Azure VM
- Add rule to NSG, open port 80
- Create storage account in the same location of the vm
- Upload your custom script to the storage account container
- Add your custom script to extensions section in your vm

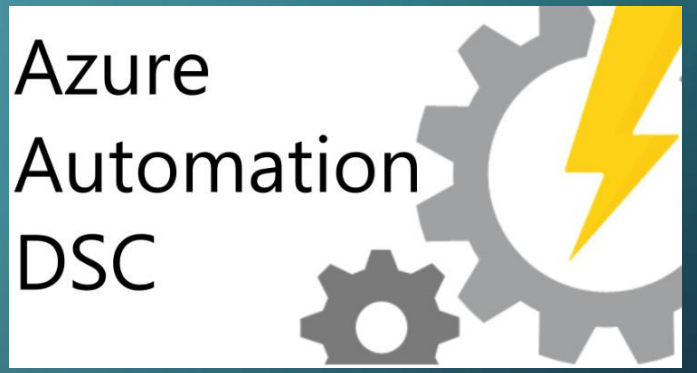
Infrastructure – PS Desire State Configuration

Demo lab

- Create Azure VM
- Add rule to NSG, open port 80
- Create storage account in the same location of the vm
- Upload your configuration file “zip files” & Data configuration file “.psd1” to your storage account container
- Add your DSC files to extensions section in your vm

Infrastructure – PS Desire State Configuration - Maintain

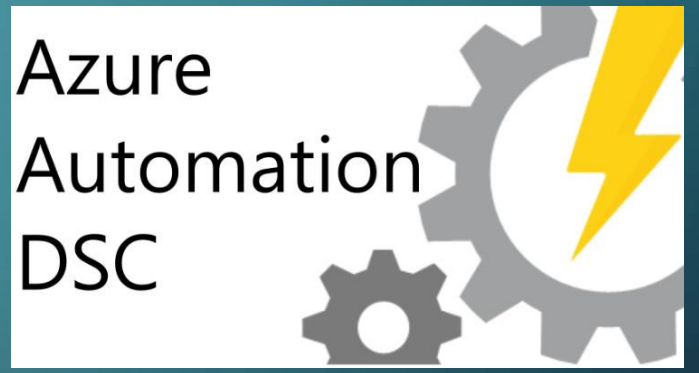
Demo lab



- Create Azure Automation Account
- Go to state configuration (DSC)
- Configurations/Add→ Upload your .ps1 from local machine
- Open your uploaded file and Compile it
- Nodes/Add → Choose the machine, choose node configuration, modify Configuration mode
- Check and remove IIS role

Infrastructure – Automation Account on other machines

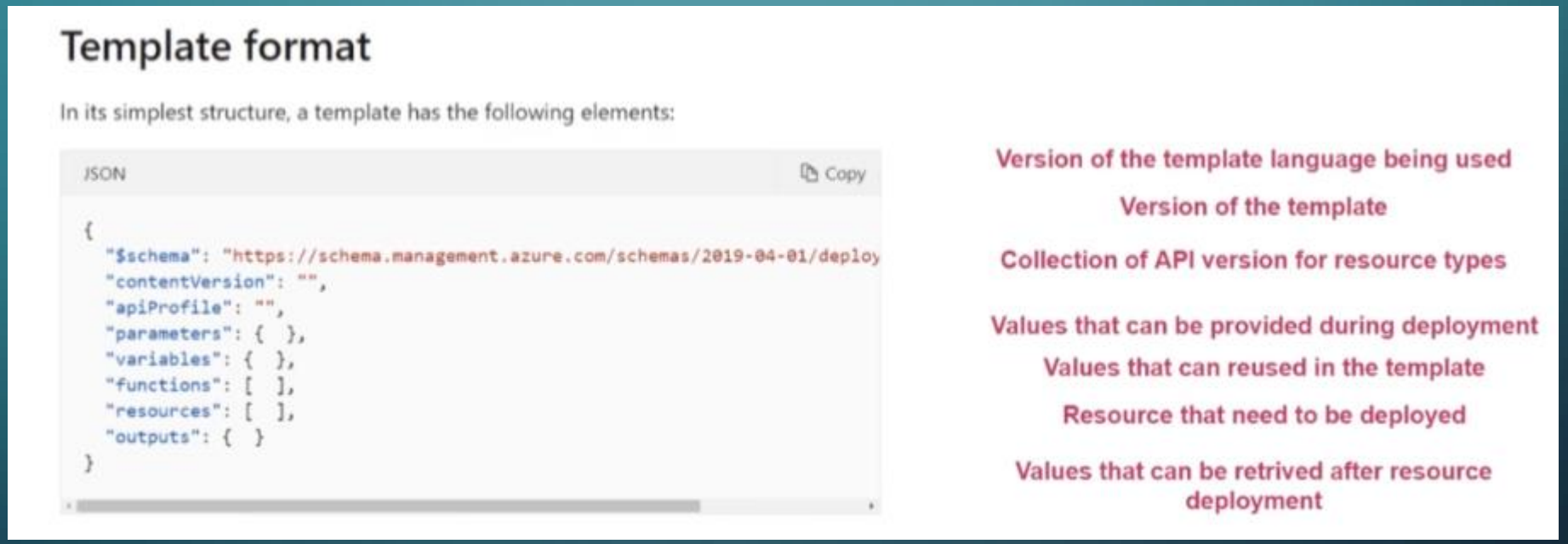
Demo lab



- Create new vm
- Go to state configuration (DSC)
- Configurations/Add→ Upload your .ps1 from local machine
- Open your uploaded file and Compile it
- Edit config file (RegistrationUrl , Registrationkey, configFileName, ComputerName)
- Run the config file to create DSCMetaConfigs
- Set-DscLocalConfigurationManager -Path ./DscMetaConfigs

Infrastructure – Review topics

- Azure Resource Manager template



Infrastructure – Review topics

ARM template – Storage account

Demo lab

Infrastructure – Review topics

ARM template – VM

Demo lab

Infrastructure – Review topics

ARM template – VM with custom script

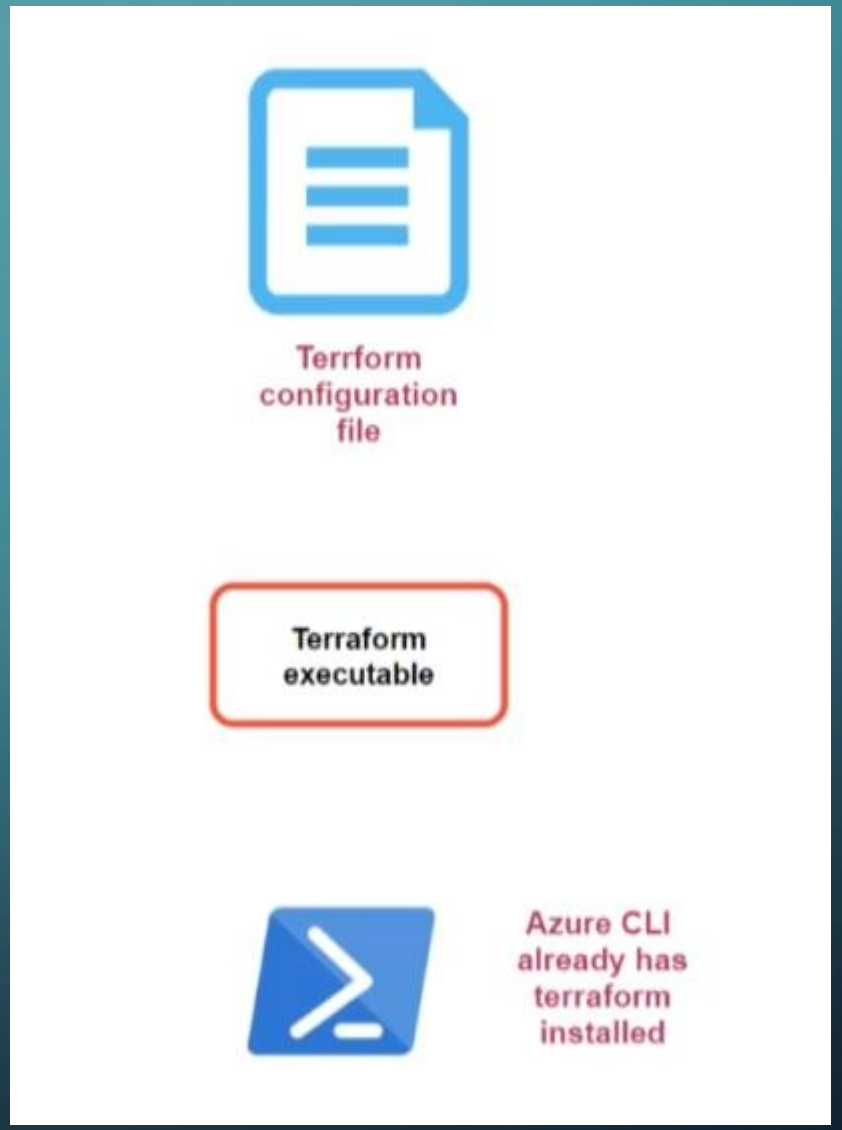
Demo lab

Infrastructure – IAC

- **These are a few popular choices:**
  - Chef
  - Puppet
  - Red Hat Ansible Automation Platform
  - Saltstack
  - Terraform
  - AWS CloudFormation
  - Azure ARM

Infrastructure – IAC

Terraform



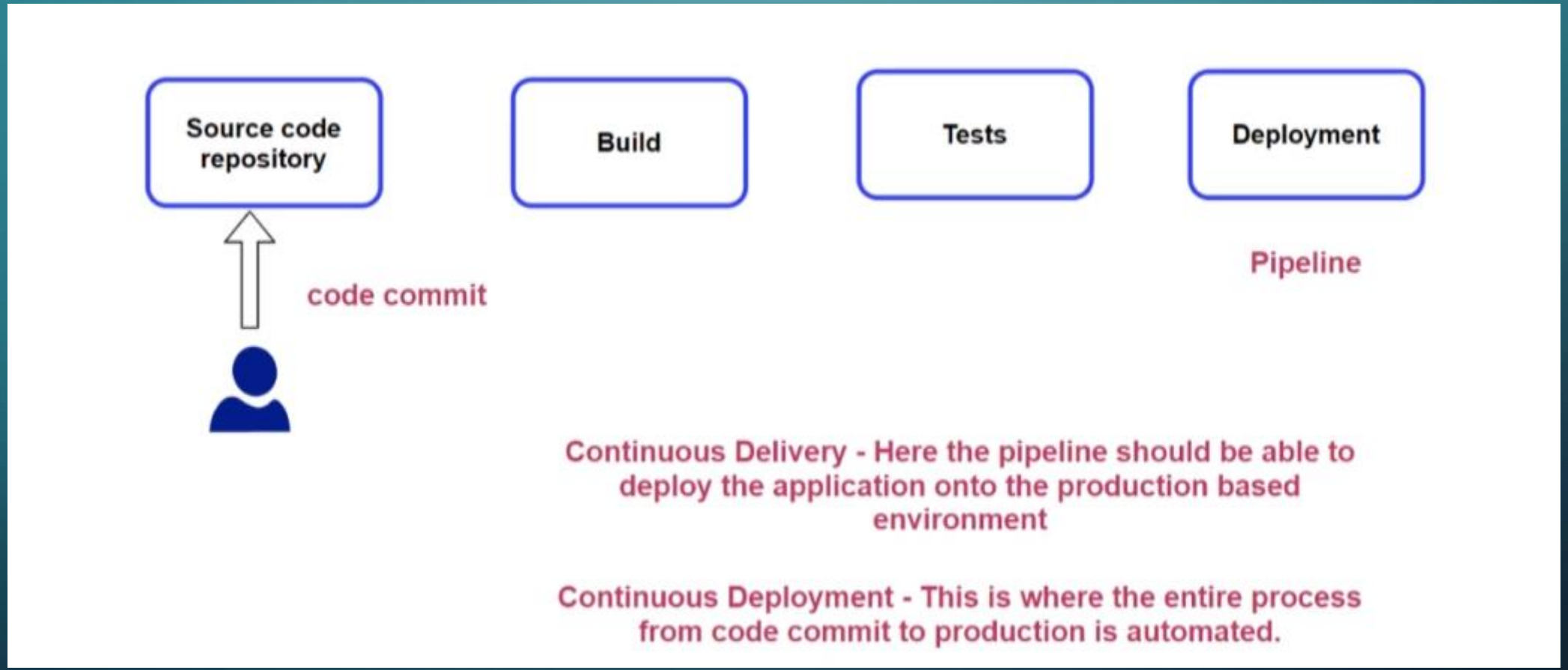
Terraform – storage account

Demo lab

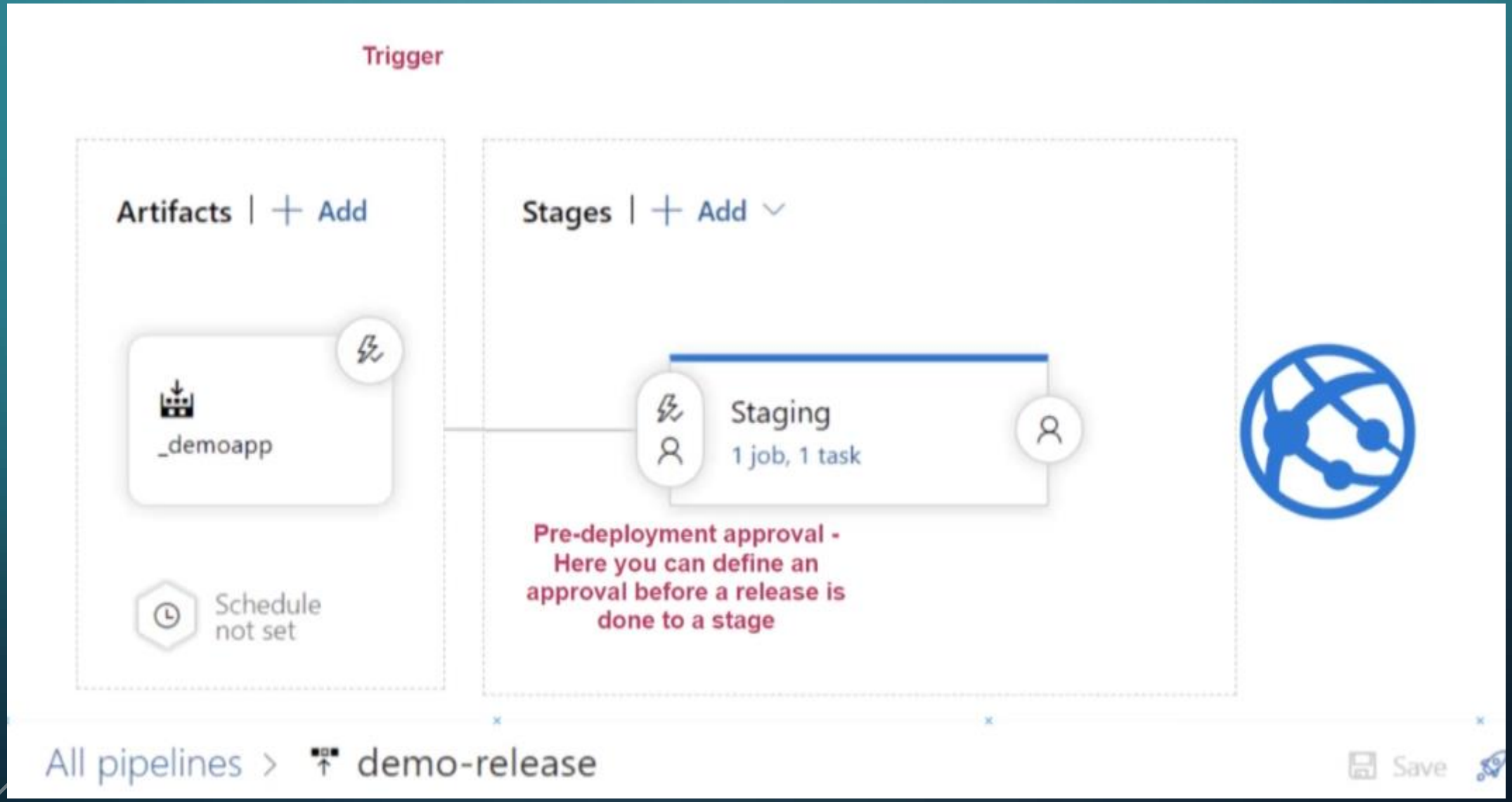
Terraform – VM

Demo lab

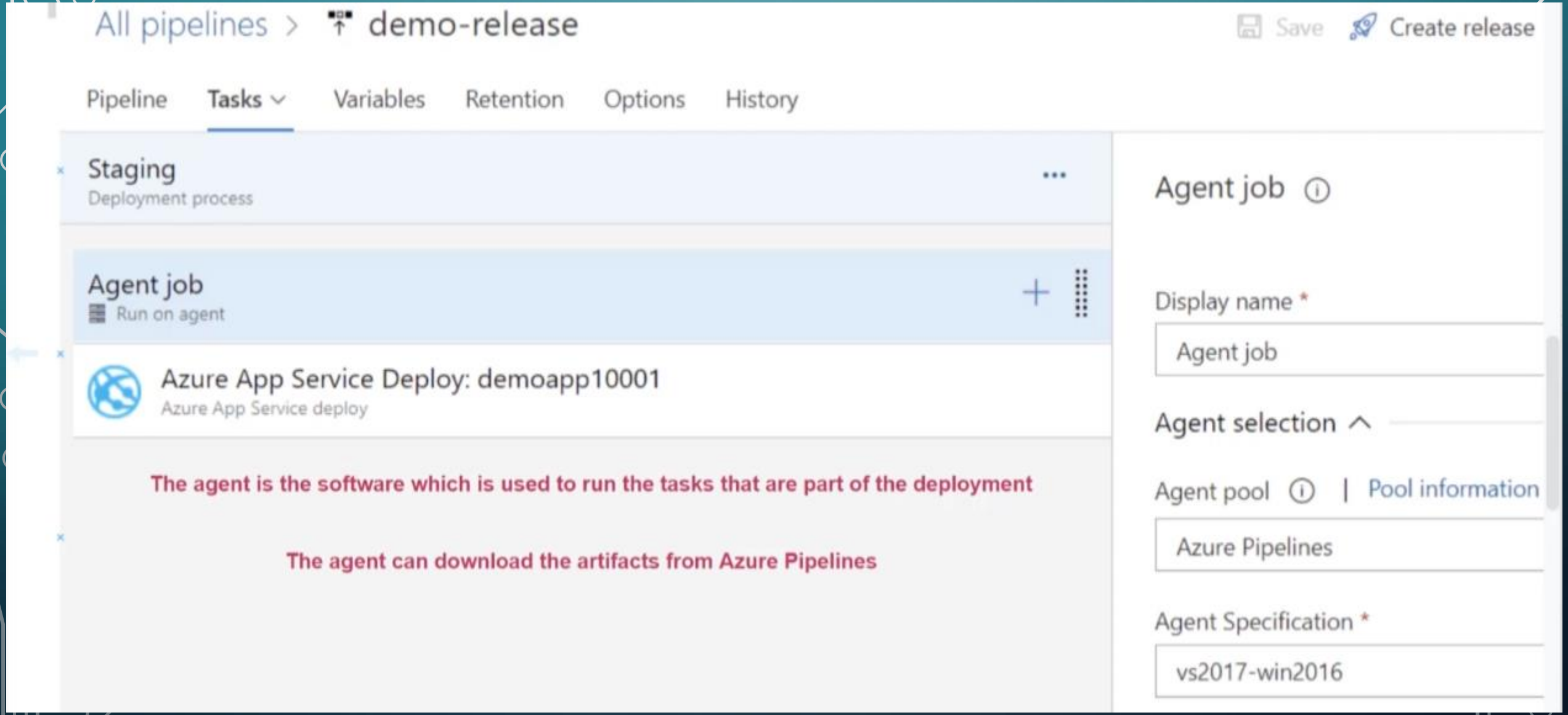
Continuous Delivery



Azure Release Pipeline



Azure Release Pipeline



Release Pipeline – Az webApp

Demo lab

- Create new azure web app
- Create new Build pipeline based on Azure repos
- Publish build artifact
- Create Azure pipeline release

Release Pipeline – Continuous trigger

Demo lab

- Enable trigger
- Make a change in our code

Release Pipeline – Multi stages

Demo lab

- Edit the last release and add new stage
- Parallel and sequential
- Create new web app
- Create new release with both stages

Release Pipeline – Approvals and Gates

- More control over the deployment pipeline.
- Scenarios:
  - Users need to validate a change request before the deployment – Pre-deployment approval.
  - What you need to do after the current stage – Post deployment gates

Release Pipeline – Approvals

Demo lab

- Edit the stages release pipeline
- Try pre-deployment conditions – pre-deployment approvals
- Change the timeout
- Change in you code again

Release Pipeline – Gates

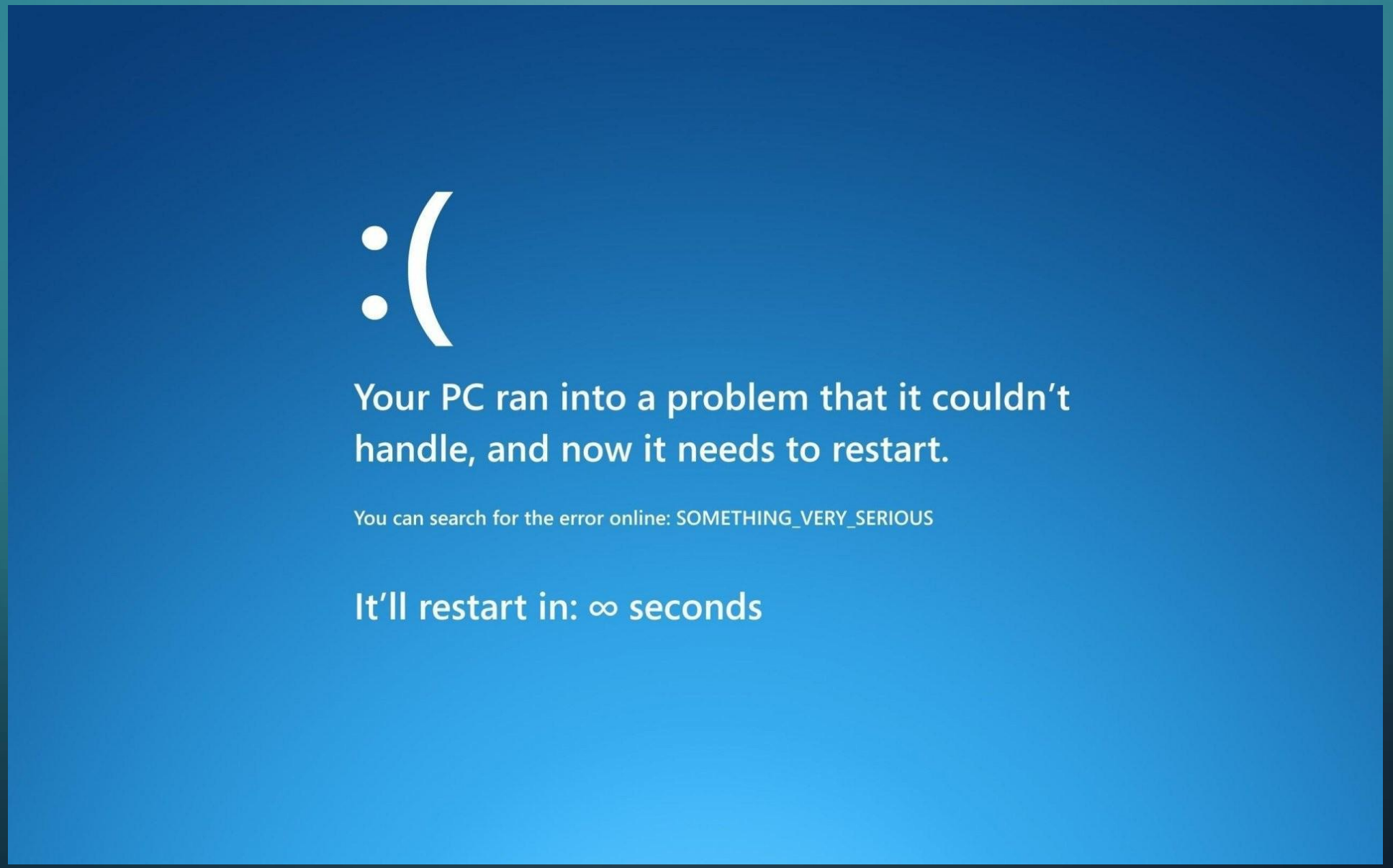
Demo lab

- Edit the stages release pipeline
- Disable the pre-deployment approvals
- Enable Gates – connect with Query work items
- Make a new query “if bug not equal to closed” – open Query security – projectName build service “permission”
- Run the release and after some time close the bugs

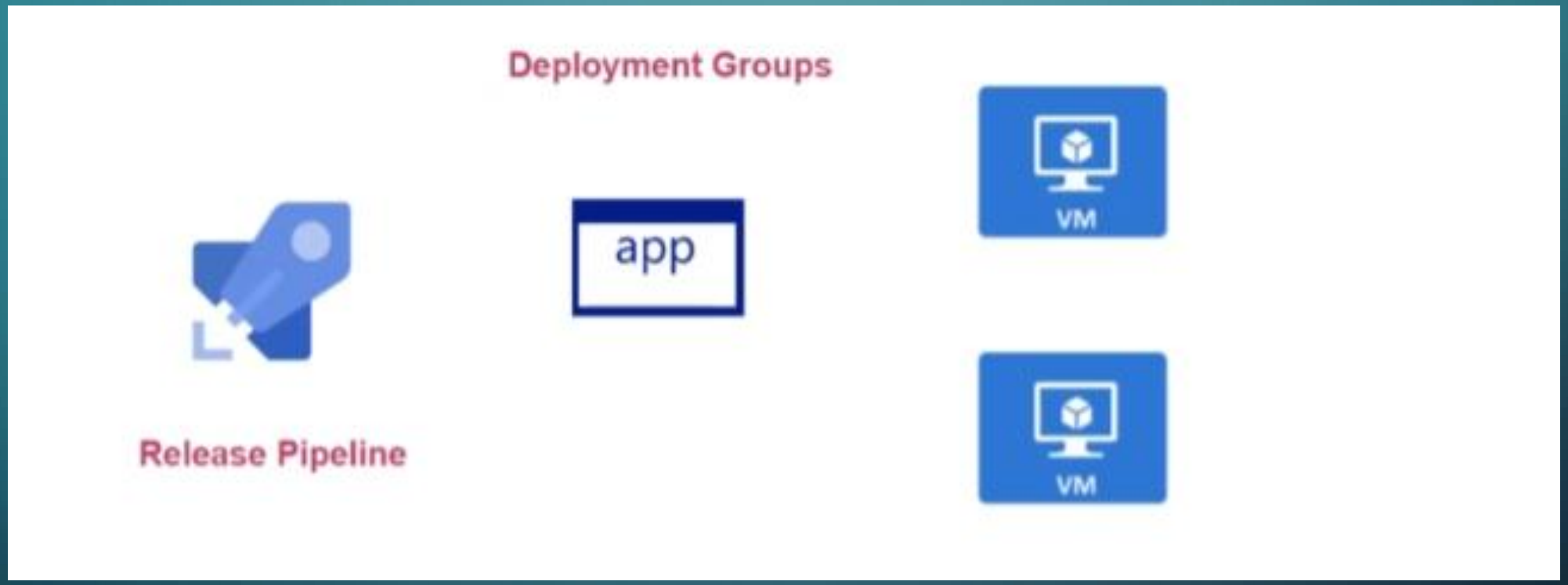
Release Pipeline – Gates – azure policies

Demo lab

- Add definition in Azure policy – tag required for resources
- Enable Gates – Check azure policy
- Run the release and after some time add tag to the resource


:(
Your PC ran into a problem that it couldn't handle, and now it needs to restart.
You can search for the error online: SOMETHING_VERY_SERIOUS
It'll restart in: ∞ seconds

Release Pipeline – deployment group



Release Pipeline – deployment group

Demo lab

- Create a virtual machine and install IIS and .NET Core Hosting bundle
- Add deployment group and run the script to add the agent
- Use PAT for authentication
- Replace the agent by Deployment group job
- Add task: “IIS web app manage“
- Add task: “IIS web app deploy“

Docker - revision

Demo lab

- Create Linux vm
- Install docker on it
- Pull Nginx service
- Run Nginx service on port 80

Revision– Dockerize .NET Core app

Demo lab

- Create new project based on .NET Core
- Publish the project: dotnet publish
- Copy the publish folder to our Linux machine
- Make the Dockerfile
- Build your image
- Run your image as a container

Revision–Azure Container Registry "ACR"

Demo lab

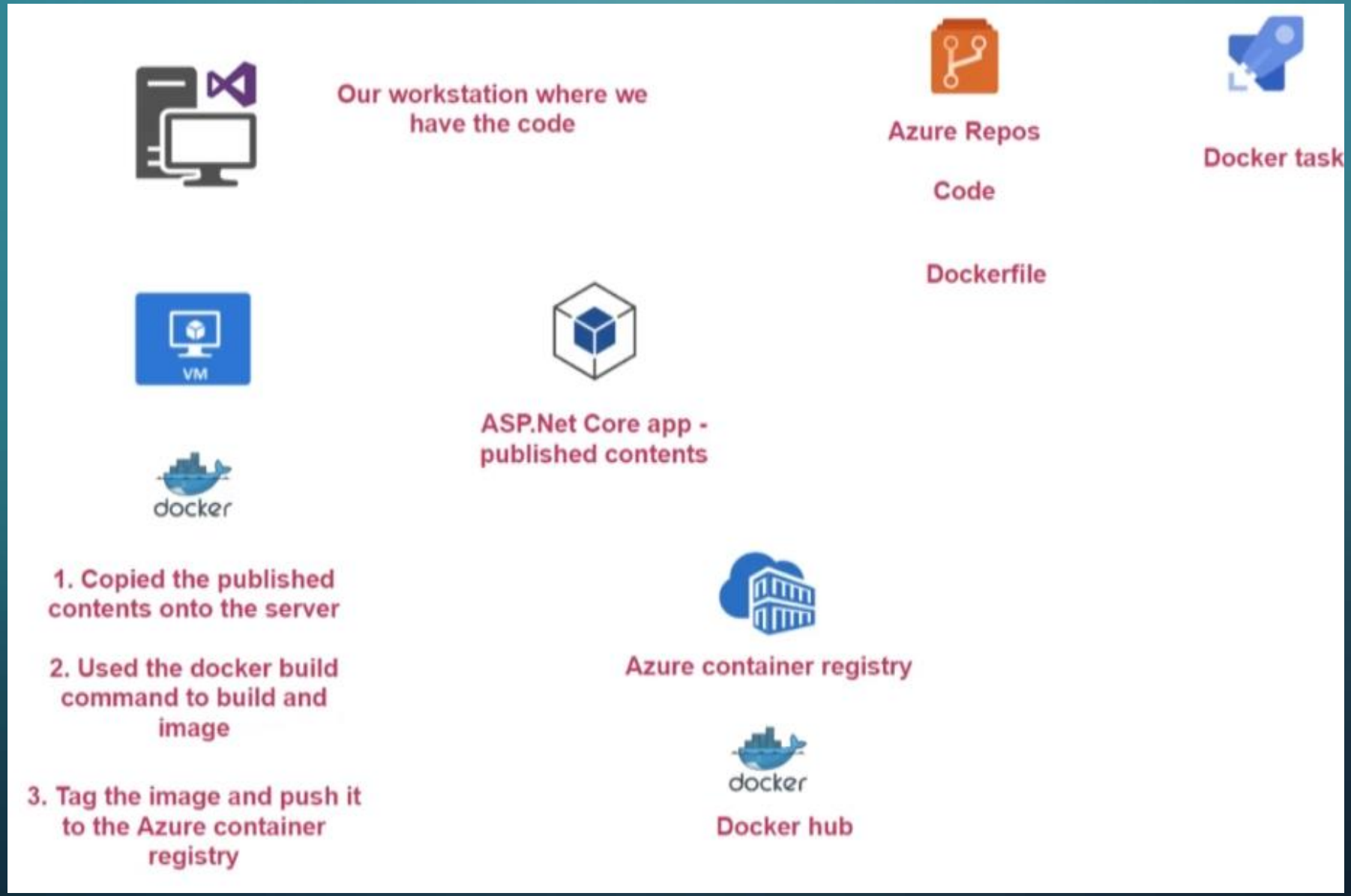
- Create Azure Container Registry
- Install azure-cli to the Linux machine
- Login to Azure subscription and ACR
- Tag our image by the acr name and url
- Push the image to the acr
- Enable admin user option if you will use ACI – from Access keys

Revision–Deploy to Kubernetes

Demo lab

- Create Azure k8s service
- Install Kubernetes-cli on your local machine by choco
- Apply the manifest files “deployment&service”
- You can also deploy these file from the portal

Publish to Azure Container Instance ACI



Publish to Azure Container Instance ACI

Demo lab

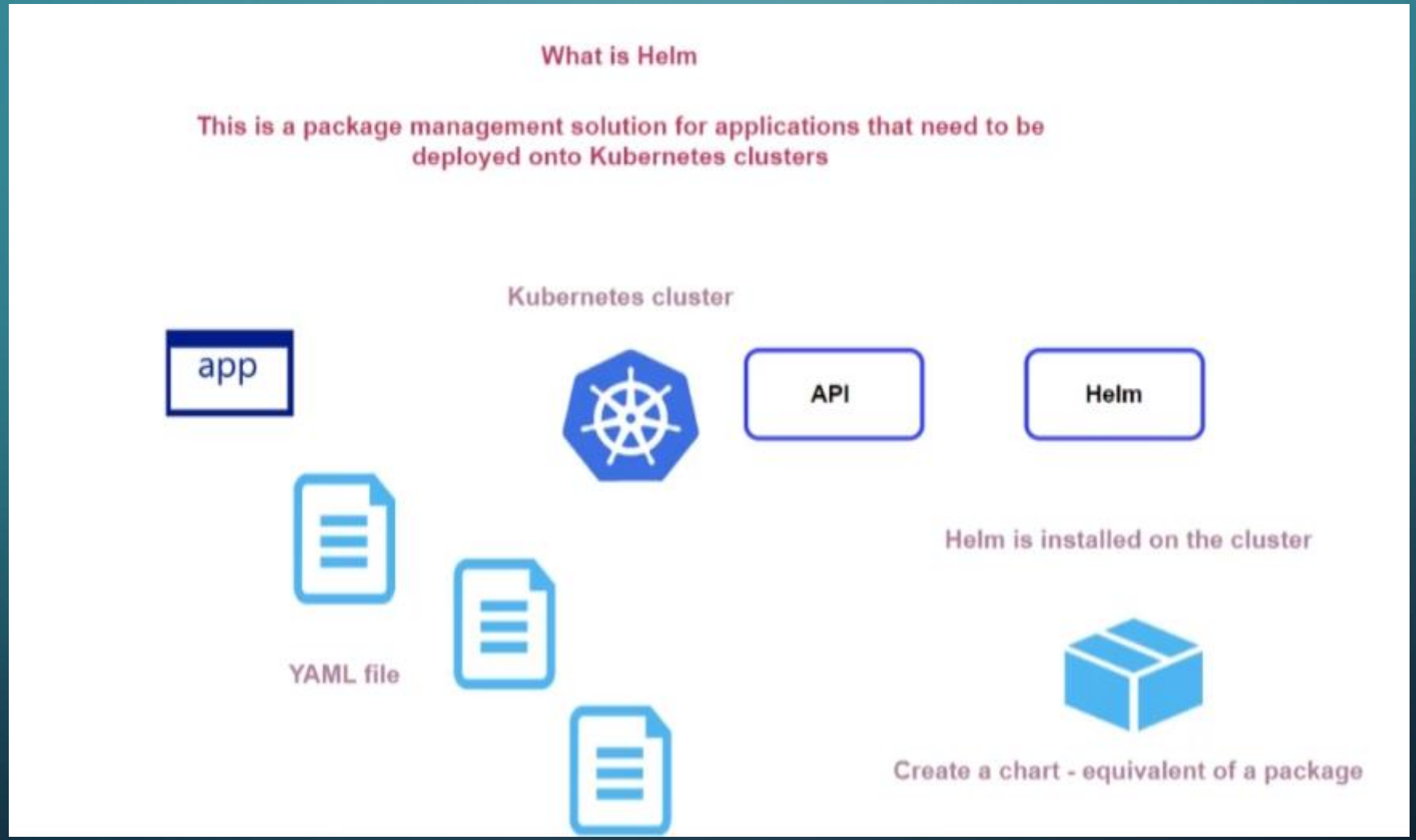
- Stop the last ACI we use
- Push your code to the repo and make your Dockerfile
- Build your Azure pipeline
- Choose to build and push to ACR
  - IT will create new service connection with our ACR
- Start the ACI and check the new release

Azure Pipeline - AKS

Demo lab

- Remove your latest deployment and service from AKS
- Upload your manifest to the azure repo
- Create service connection for AKS
- Build your Azure pipeline
  - Add task: copy the manifest files
  - Add task: Kubernetes manifest
  - Run the pipeline

Helm Chart


What is Helm
This is a package management solution for applications that need to be deployed onto Kubernetes clusters
Kubernetes cluster
app
API
Helm
YAML file
Helm is installed on the cluster
Create a chart - equivalent of a package

Helm Chart

Demo lab

- Create K8s Cluster
- Install helm on your local machine “choco install Kubernetes-helm”
- Install mysql app on you cluster by using helm: “helm install app-name stable/mysql
- Check the pods on your k8s cluster

Azure pipeline – system defined variables

```yaml
variables:
  # Container registry service connection established during pipeline creation
  dockerRegistryServiceConnection: 'd8aaf5e0-c674-4dc7-a221-ef0d41ee62ec'
  imageRepository: 'dotnet'
  containerRegistry: 'appregistry210.azurecr.io'
  dockerfilePath: '$(Build.SourcesDirectory)/Dockerfile'
  tag: 'latest'
```

Settings

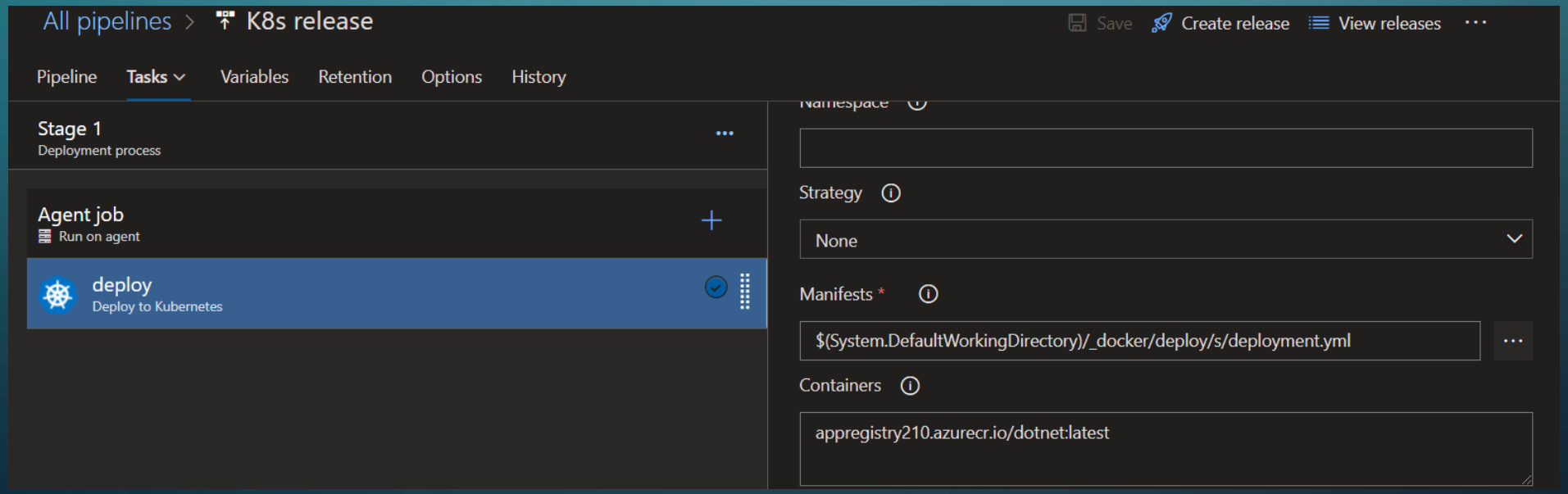
```yaml
    - task: PublishBuildArtifacts@1
      inputs:
        PathtoPublish: '$(Build.ArtifactStagingDirectory)'
        ArtifactName: 'drop'
        publishLocation: 'Container'
```

Settings

```yaml
    - task: PublishPipelineArtifact@1
      inputs:
        targetPath: '$(Pipeline.Workspace)'
        artifact: 'deploy'
        publishLocation: 'pipeline'
```

Azure pipeline – system defined variables

```
Settings
41      - task: PublishPipelineArtifact@1
42        inputs:
43          targetPath: '$(Pipeline.Workspace)'
44          artifact: 'deploy'
45          publishLocation: 'pipeline'
```



Dockerfile

```
  GNU nano 2.5.3                                File: .\Dockerfile

FROM mcr.microsoft.com/dotnet/core/sdk:3.1 AS build-env
WORKDIR /app

# Copy csproj and restore as distinct layers
COPY *.csproj ./
RUN dotnet restore

# Copy everything else and build
COPY . ./
RUN dotnet publish -c Release -o out

# Build runtime image
FROM mcr.microsoft.com/dotnet/core/aspnet:3.1
WORKDIR /app
COPY --from=build-env /app/out .
ENTRYPOINT ["dotnet", "docker-app.dll"]
```

Azure web app – Azure sql database

Demo lab

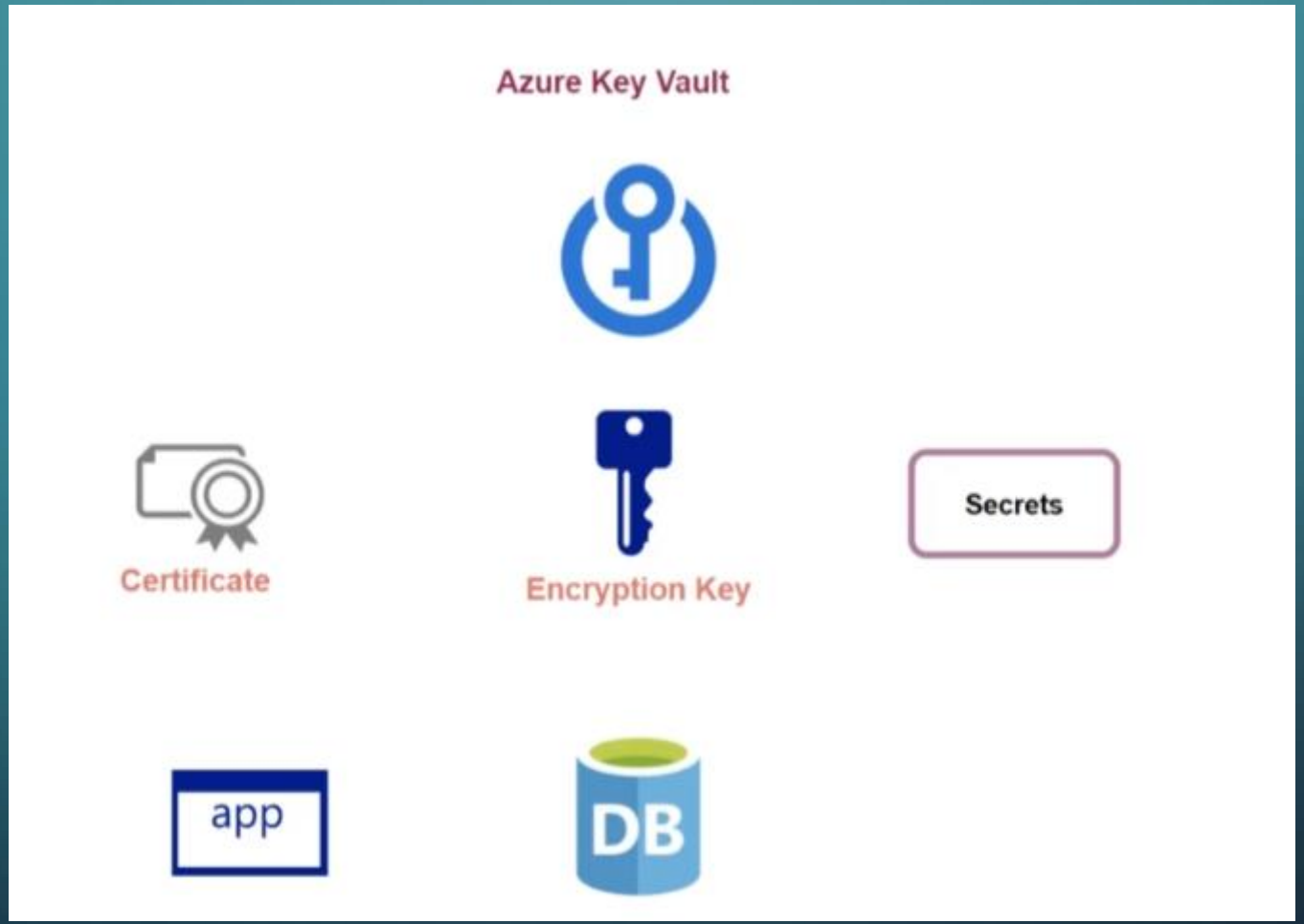
- Create Azure sql database
- Open the database by Microsoft sql server management studio or the azure editor.
- Create table and insert some data
- Connect your application with the database: “connection string” and check locally.
- Create new web app and go to show that you can configuration the connection string on it.

Complete – with Azure pipeline

Demo lab

- Drop the table
- Create new app and push your code
- Create new pipeline and add task: “publish build artifact”
- Create new release pipeline.
  - Add task: Azure sql database deployment
  - Specify the database password as a variable
  - Input your inline sql script to create a table again
  - Add task to deploy to web app
  - Add another task: azure app service settings, to add connection strings
  - Add the variables which you need
- Go to the repo add remove the connection string to make a trigger

Review topics – Azure key vault



Azure Release pipelines – Azure key vault

Demo lab

- Create Azure key vault and add secret vmpassword
- Create new release pipeline
  - Add task: Azure key vault
  - Add task: Azure CLI
- Add access policy and understand it
  - Copy the complete service principle id for your azure devops
  - Add the SP to access policy on azure key vault

Azure pipelines - Variable Groups

Demo lab

- In the Azure pipeline section > Library > can create variable group
- Add variable what you need.
- You can use this variable across of all pipelines you have

```
12  variables:
13  - group: app-group
14
15  jobs:
16  - job: Test_variable_group
17    steps:
18    - bash: echo $(secret)
```

Variable Groups – Azure key vault

Demo lab

- Create a new secret in Azure key vault
- Enable the link option in the variable group “link with azure key vault”
- Add secrets from key vault which you need to your variable group.
- Run the last pipeline and it still work and can fetch the secret.

Release pipeline - Variable Groups

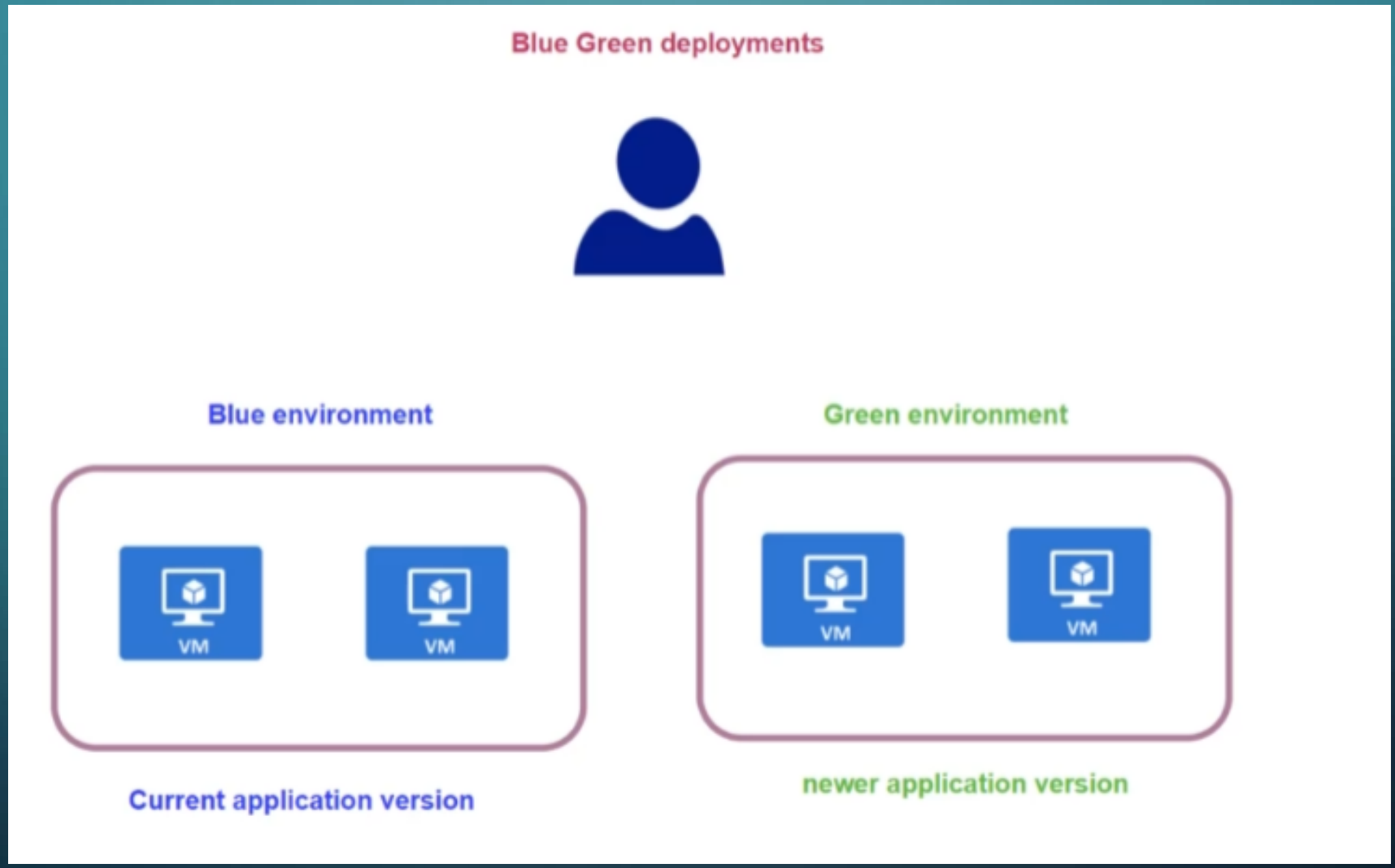
Demo lab

- Go to the last release pipeline which we use it with azure key vault task.
- Remove the Azure key vault task
- Link the release variable with the variable group which you have.
- And try to deploy the Azure CLI task and it will work successfully

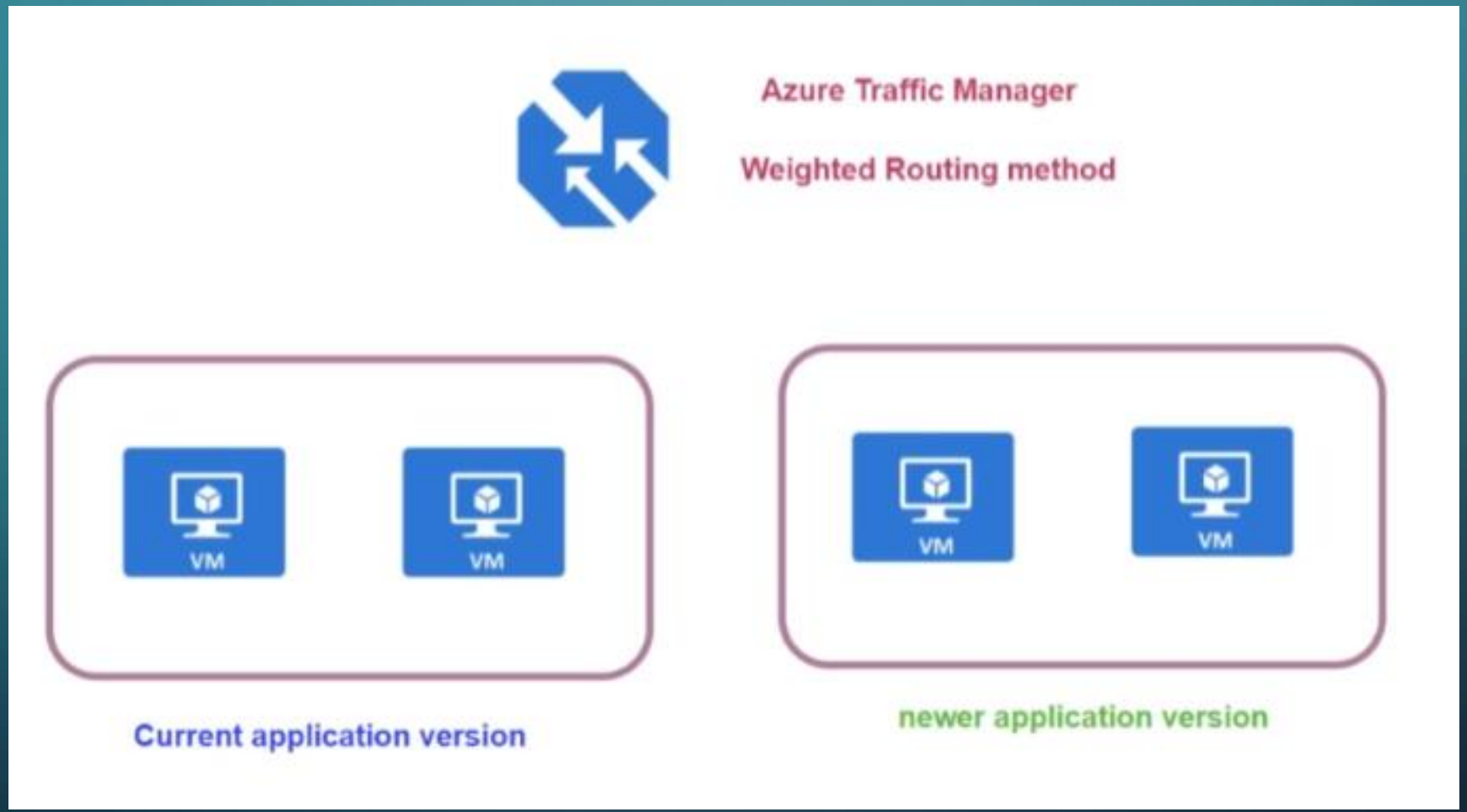
AZURE WEB APP – DEPLOYMENT SLOT

- Azure web app – deployment slot

Deployment strategies – blue green



Deployment strategies - canary



Azure web app deployment slot

Demo lab

- Create new azure web app – standard or higher
- Deploy from visual studio
- Add new slot
- Deploy on the slot from visual studio
- Swap between the slots

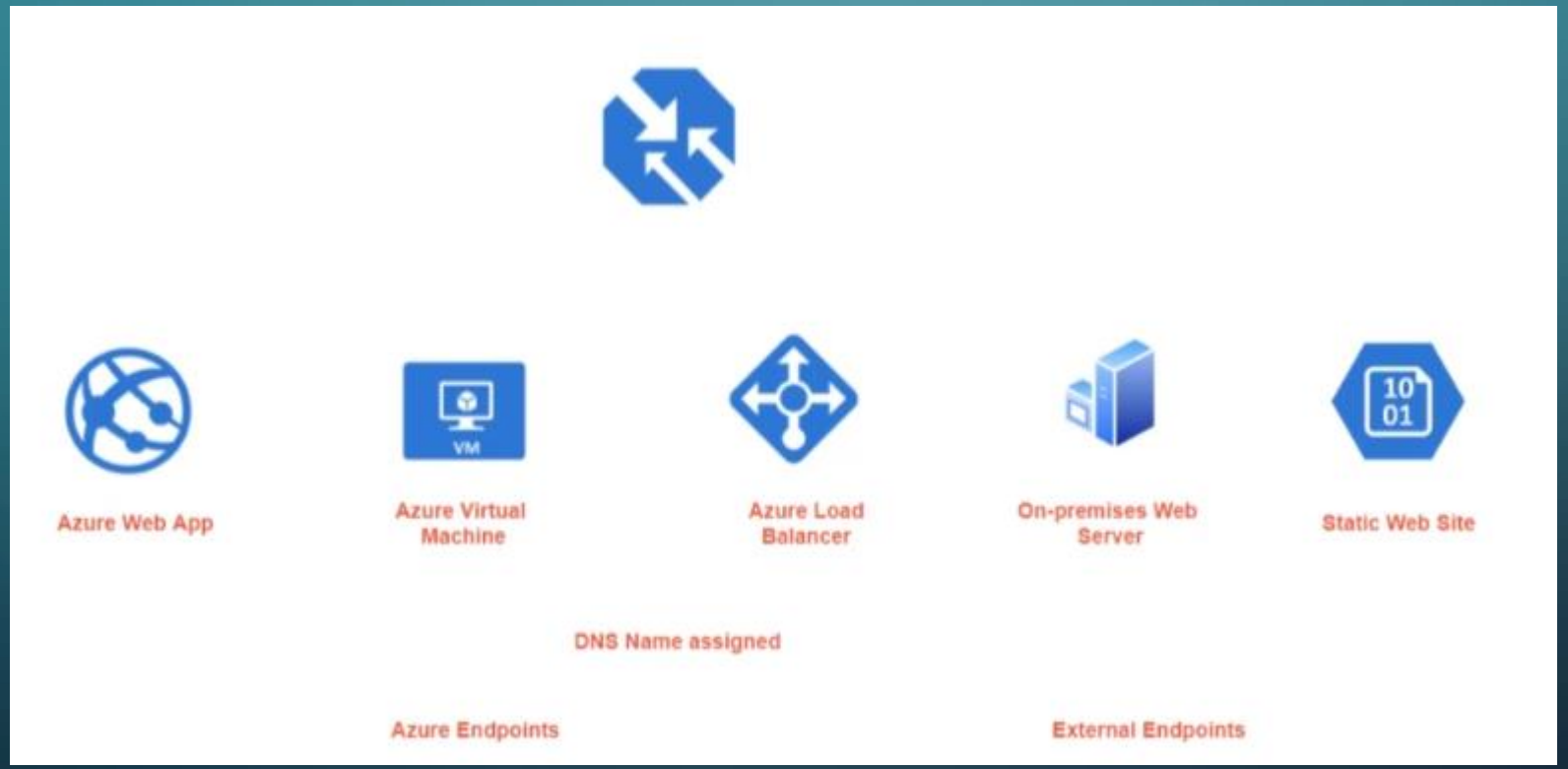
Azure pipelines - deployment slot

Demo lab

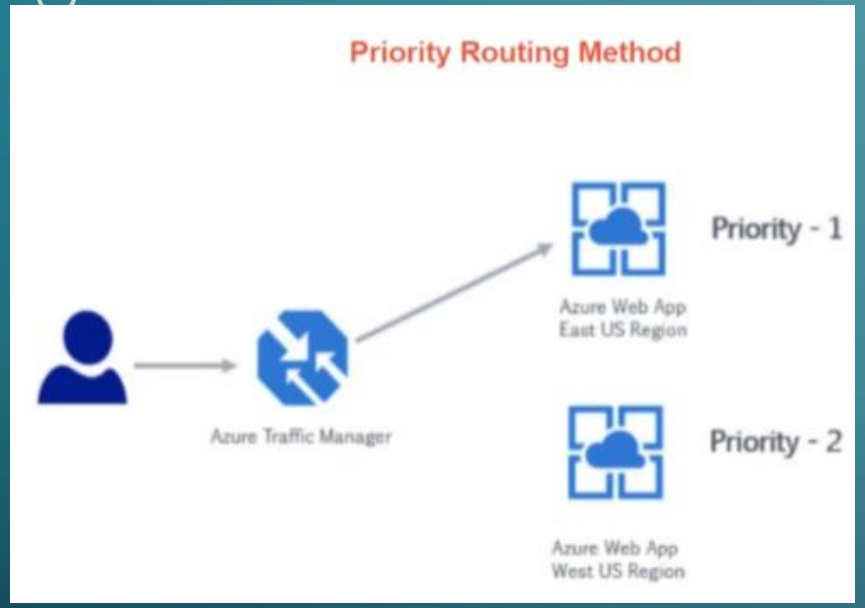
- Create new repo and push you code on it
- Delete the slot which we create it in the last demo
- Create new build pipeline and publish our build.
- Create new release pipeline – with two stages
  - Add task “pre-prod”: azure cli – to crease deployment slot to the web app
  - Add task “pre-prod”: app service deploy – choose option to deploy to slot
  - Add task “prod”: app service manage – choose action to swap slots
- Edit in your code to make a trigger

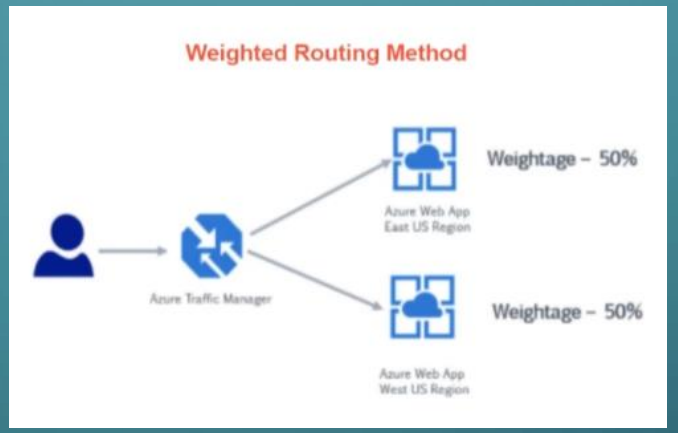
Review topics

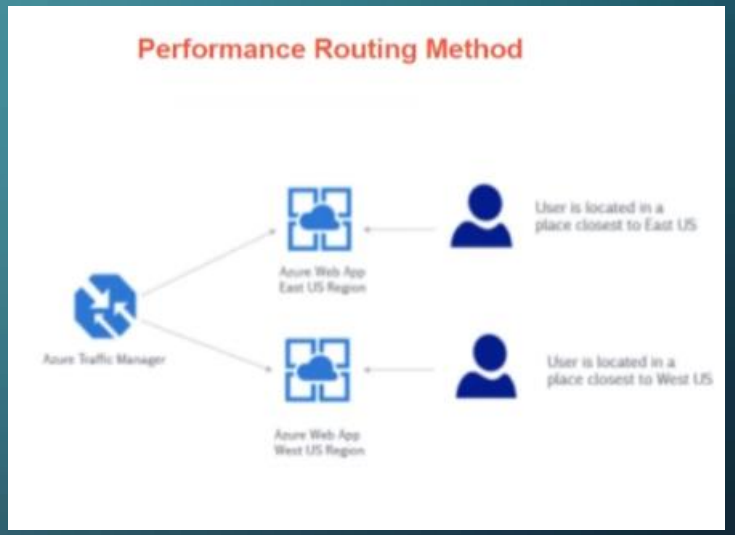
- Azure traffic manager



Azure traffic manager







AZURE TRAFFIC MANAGER

Demo lab

- Create two web app in different regions
- Publish to the both web app by VS
- Create traffic manager profile – priority method
- Add two endpoints for both web app – wait to endpoint to be online state
- In the configuration change the port to 443
- Check the traffic manager url

Azure pipeline - Azure traffic manager

Demo lab

- Delete the second endpoint
- Create new repo and push the code by VS
- Create new pipeline and publish build artifact
- Create new release pipeline with two stages
  - Add task “pre-prod”: app service deploy
  - Add task “pre-prod”: Azure CLI
  - Add task “prod”: app service manage and choose stop action
- Check the traffic manager url

Azure pipeline - terraform

Demo lab

- Create new repo and push our code “terraform code for create vm”
- Create build pipeline and publish build pipeline
- Create new release pipeline:
  - Add task: Azure CLI – to create storage account
  - Add task: terraform tool installer
  - Add task: terraform – init – define the container name for the .tfstate
  - Add task: terraform – plan and apply –auto-approve

Ansible Demo

Demo lab

SRE – Site Reliability Engineering

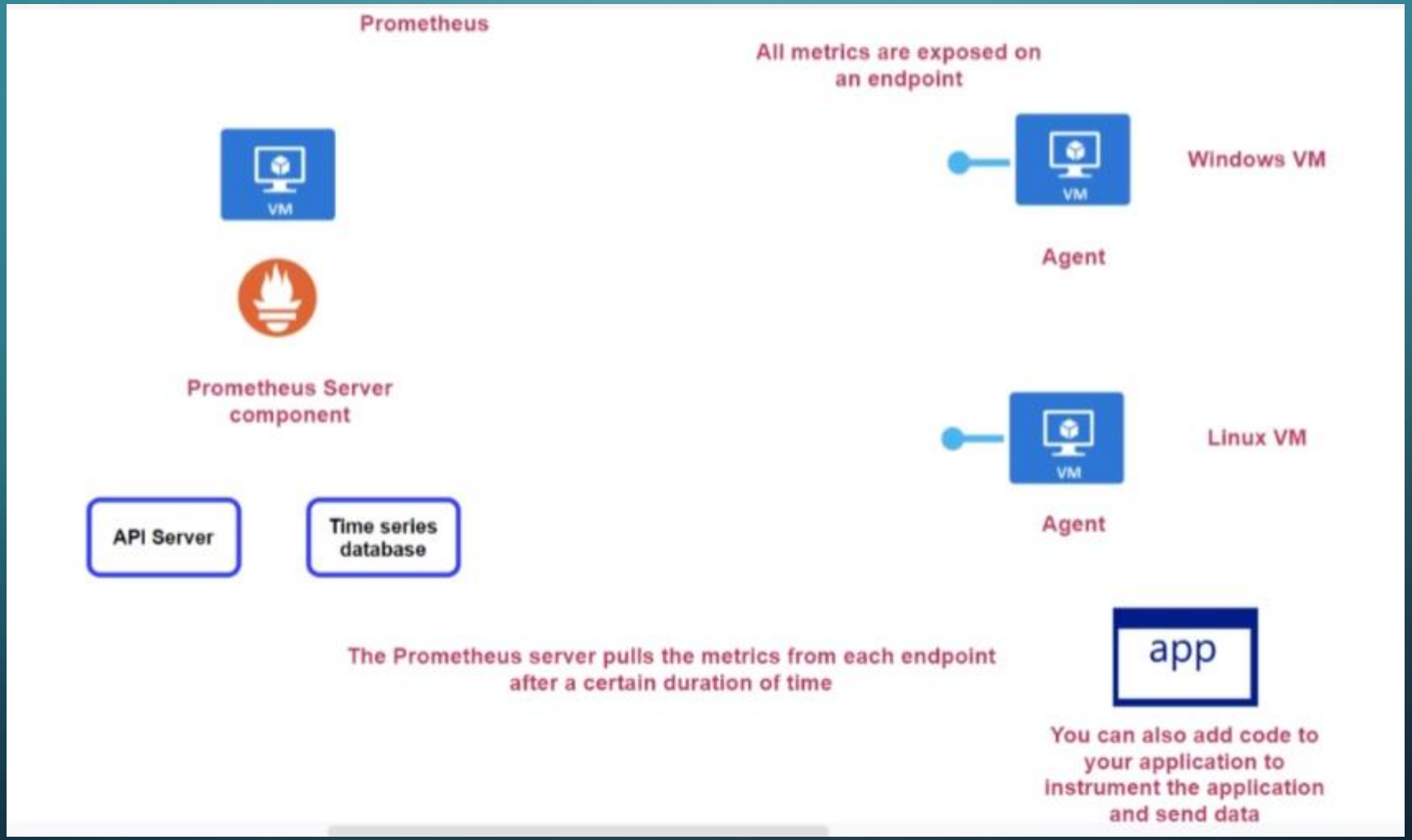
- Azure LoadBalaner
- Availability set
- Diagnostic settings
- Azure virtual machine scale set
- Azure monitor – activity logs – alerts – metrics
- Azure log analytics
- Azure application insights

Azure Application insights

Demo lab

- Monitoring Application Performance with Application Insights
  - Create web app with application insights and publish .NET Core app
  - Live metrics – performance – failure – users
- Smart detection based on AI
- Continuous Export – send data collected by app insights to Azure storage account

Prometheus


Prometheus
All metrics are exposed on an endpoint
VM
Windows VM
Agent
Prometheus Server component
VM
Linux VM
Agent
API Server
Time series database
The Prometheus server pulls the metrics from each endpoint after a certain duration of time
app
You can also add code to your application to instrument the application and send data