Deployment Guide - Azure DevOps VMSS with Microsoft Prebuilt Images

This comprehensive guide provides step-by-step instructions for deploying the Azure DevOps VMSS template with Microsoft prebuilt runner images in a secure, vWAN-integrated environment.

Prerequisites Checklist

Before starting the deployment, ensure you have:

Azure Requirements

- [ ] Azure subscription with **Contributor** access to target resource group
- [ ] Existing **vWAN with Azure Firewall** (or compatible network infrastructure)
- [ ] **Azure CLI** installed and configured (`az --version`)
- [ ] **PowerShell** 5.1 or later (for Windows) or **PowerShell Core** (for Linux/macOS)

Azure DevOps Requirements

- [ ] Azure DevOps organization with **Project Administrator** access
- [ ] **Personal Access Token** with Agent Pools (read, manage) permissions
- [ ] **Agent pool** created (or permission to use Default pool)

Development Environment

- [ ] **Git** installed for repository operations
- [ ] **Text editor** for configuration file modifications
- [ ] **Network connectivity** to Azure and Azure DevOps services

Marketplace Terms (Required for Visual Studio Images)

- [ ] Marketplace terms accepted for chosen Visual Studio images

Step-by-Step Deployment

Step 1: Prepare Azure Environment

1.1 Create Resource Group

```
# Set variables
RESOURCE_GROUP="rg-devops-vmss"
LOCATION="East US"

# Create resource group
az group create \
  --name $RESOURCE_GROUP \
  --location "$LOCATION"

# Verify creation
az group show --name $RESOURCE_GROUP --output table
```

1.2 Accept Marketplace Terms

Required for Visual Studio images:

```
# Visual Studio 2022 Enterprise (most common)
az vm image terms accept \
  --publisher MicrosoftVisualStudio \
  --offer visualstudio2022 \
  --plan vs-2022-ent-latest-ws2022

# Visual Studio 2022 Build Tools (for CI/CD)
az vm image terms accept \
  --publisher MicrosoftVisualStudio \
  --offer visualstudio2022 \
  --plan vs-2022-buildtools-latest-ws2022

# Visual Studio 2019 Enterprise (for legacy projects)
az vm image terms accept \
  --publisher MicrosoftVisualStudio \
  --offer visualstudio2019 \
  --plan vs-2019-ent-latest-ws2019
```

1.3 Verify Network Infrastructure

```
# If using existing vWAN, verify connectivity
az network vwan list --output table

# Check virtual network if using existing VNet
az network vnet list --resource-group <network-rg> --output table

# Verify Azure Firewall configuration
az network firewall list --output table
```

Step 2: Configure Azure DevOps

2.1 Create Personal Access Token

1. Navigate to Azure DevOps → User Settings → Personal Access Tokens
2. Click **"New Token"**
3. Configure token:
   - **Name**: `VMSS-Agent-Token`
   - **Expiration**: 90 days (or as per policy)
   - **Scopes**: Custom defined
   - **Agent Pools**: Read & manage
4. Copy the token securely

2.2 Create or Verify Agent Pool

```
# List existing agent pools
az pipelines pool list --organization https://dev.azure.com/yourorg

# Create new agent pool (if needed)
az pipelines pool create \
  --name "VS2022-Agents" \
  --organization https://dev.azure.com/yourorg
```

Step 3: Prepare Template Configuration

3.1 Clone Repository

```
# Clone the repository
git clone <your-repository-url>
cd azure-devops-vmss-template

# Create working branch
git checkout -b deployment/production
```

3.2 Configure Parameters

Create a production parameters file:

```
cp bicep/parameters.json bicep/parameters-prod.json
```

Edit `bicep/parameters-prod.json`:

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentParamet-
ers.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "location": {
      "value": "East US"
    },
    "vmssName": {
      "value": "vmss-devops-agents"
    },
    "vmSize": {
      "value": "Standard_D4s_v3"
    },
    "instanceCount": {
      "value": 2
    },
    "adminUsername": {
      "value": "azureuser"
    },
    "agentPoolName": {
      "value": "VS2022-Agents"
    },
    "microsoftImageType": {
      "value": "vs2022-enterprise"
    },
    "azureDevOpsUrl": {
      "value": "https://dev.azure.com/yourorganization"
    },
    "configureDevOpsAgent": {
      "value": true
    },
    "useExistingVnet": {
      "value": false
    },
    "vnetName": {
      "value": "vnet-devops"
    },
    "subnetName": {
      "value": "subnet-agents"
    },
    "vnetAddressPrefix": {
      "value": "10.0.0.0/16"
    },
    "subnetAddressPrefix": {
      "value": "10.0.1.0/24"
    },
    "projectTag": {
      "value": "DevOps-Infrastructure"
    },
    "environmentTag": {
      "value": "Production"
    },
    "ownerTag": {
      "value": "DevOps-Team"
    }
  }
}
```

3.3 Configure for Existing vWAN (Optional)

If integrating with existing vWAN infrastructure:

```
{
  "useExistingVnet": {
    "value": true
  },
  "existingVnetResourceGroup": {
    "value": "rg-network-hub"
  },
  "existingVnetName": {
    "value": "vnet-hub"
  },
  "existingSubnetName": {
    "value": "subnet-devops-agents"
  }
}
```

Step 4: Deploy Infrastructure

4.1 Validate Template

```
# Validate Bicep template
az deployment group validate \
  --resource-group $RESOURCE_GROUP \
  --template-file bicep/vmss-infrastructure.bicep \
  --parameters @bicep/parameters-prod.json \
  --parameters adminPassword='TempPassword123!' \
  --parameters azureDevOpsPat='temp-token'
```

4.2 Deploy Template

```
# Set secure parameters
read -s -p "Enter admin password: " ADMIN_PASSWORD
echo
read -s -p "Enter Azure DevOps PAT: " DEVOPS_PAT
echo

# Deploy infrastructure
az deployment group create \
  --resource-group $RESOURCE_GROUP \
  --template-file bicep/vmss-infrastructure.bicep \
  --parameters @bicep/parameters-prod.json \
  --parameters adminPassword="$ADMIN_PASSWORD" \
  --parameters azureDevOpsPat="$DEVOPS_PAT" \
  --name "vmss-deployment-$(date +%Y%m%d-%H%M%S)"
```

4.3 Monitor Deployment

```
# Check deployment status
az deployment group list \
  --resource-group $RESOURCE_GROUP \
  --output table

# Get deployment details
az deployment group show \
  --resource-group $RESOURCE_GROUP \
  --name <deployment-name> \
  --output json
```

Step 5: Verify Deployment

5.1 Check Azure Resources

```
# List all resources in the resource group
az resource list \
  --resource-group $RESOURCE_GROUP \
  --output table

# Check VMSS status
az vmss show \
  --resource-group $RESOURCE_GROUP \
  --name vmss-devops-agents \
  --output table

# Check VMSS instances
az vmss list-instances \
  --resource-group $RESOURCE_GROUP \
  --name vmss-devops-agents \
  --output table
```

5.2 Verify Agent Registration

1. **Azure DevOps Portal**:
   - Navigate to Organization Settings → Agent pools
   - Select your agent pool
   - Verify agents appear as “Online”
   - Check agent capabilities include required tools

2. **Azure CLI**:

```
# List agents in pool
az pipelines agent list \
  --pool-id <pool-id> \
  --organization https://dev.azure.com/yourorg
```

5.3 Test Agent Functionality

Create a test pipeline to verify agent capabilities:

```yaml
# test-agent-pipeline.yml
trigger: none

pool:
  name: 'VS2022-Agents'

steps:
- task: PowerShell@2
  displayName: 'Test Agent Environment'
  inputs:
    targetType: 'inline'
    script: |
      Write-Host "Agent Name: $(Agent.Name)"
      Write-Host "Agent OS: $(Agent.OS)"
      Write-Host "Computer Name: $env:COMPUTERNAME"

      # Test .NET installation
      if (Test-Path "C:\Program Files\dotnet\dotnet.exe") {
        $dotnetVersion = & "C:\Program Files\dotnet\dotnet.exe" --version
        Write-Host ".NET SDK Version: $dotnetVersion"
      }

      # Test Visual Studio installation
      $vsPath = Get-ChildItem -Path "C:\Program Files*" -Name "*Visual Studio*" -
Directory -ErrorAction SilentlyContinue
      if ($vsPath) {
        Write-Host "Visual Studio Found: $($vsPath -join ', ')"
      }

      # Test Git installation
      try {
        $gitVersion = git --version
        Write-Host "Git Version: $gitVersion"
      } catch {
        Write-Host "Git not found in PATH"
      }

      # Test Chocolatey (if installed)
      try {
        $chocoVersion = choco --version
        Write-Host "Chocolatey Version: $chocoVersion"
      } catch {
        Write-Host "Chocolatey not installed"
      }

- task: DotNetCoreCLI@2
  displayName: 'Test .NET CLI'
  inputs:
    command: 'custom'
    custom: '--info'
```

Step 6: Post-Deployment Configuration

6.1 Configure Auto-scaling (Optional)

```
# Update auto-scale settings if needed
az monitor autoscale rule create \
  --resource-group $RESOURCE_GROUP \
  --autoscale-name vmss-devops-agents-autoscale \
  --condition "Percentage CPU > 80 avg 5m" \
  --scale out 2 \
  --cooldown 5

az monitor autoscale rule create \
  --resource-group $RESOURCE_GROUP \
  --autoscale-name vmss-devops-agents-autoscale \
  --condition "Percentage CPU < 20 avg 10m" \
  --scale in 1 \
  --cooldown 10
```

6.2 Set Up Monitoring

```
# Enable VM insights
az vm extension set \
  --resource-group $RESOURCE_GROUP \
  --vmss-name vmss-devops-agents \
  --name MicrosoftMonitoringAgent \
  --publisher Microsoft.EnterpriseCloud.Monitoring

# Create CPU alert
az monitor metrics alert create \
  --name "VMSS High CPU Usage" \
  --resource-group $RESOURCE_GROUP \
  --scopes "/subscriptions/$(az account show --query id -o tsv)/resourceGroups/$RE-
SOURCE_GROUP/providers/Microsoft.Compute/virtualMachineScaleSets/vmss-devops-agents" \
  --condition "avg Percentage CPU > 85" \
  --description "Alert when CPU usage is high across VMSS"
```

6.3 Security Hardening

```
# Update NSG rules if needed
az network nsg rule create \
  --resource-group $RESOURCE_GROUP \
  --nsg-name nsg-devops-agents \
  --name AllowAzureDevOpsHTTPS \
  --priority 1100 \
  --source-address-prefixes VirtualNetwork \
  --destination-address-prefixes Internet \
  --destination-port-ranges 443 \
  --access Allow \
  --protocol Tcp \
  --description "Allow HTTPS to Azure DevOps services"
```

Configuration Scenarios

Scenario 1: Visual Studio 2022 Enterprise Environment

Use Case: Full development environment with Visual Studio 2022

```
{
  "microsoftImageType": { "value": "vs2022-enterprise" },
  "vmSize": { "value": "Standard_D4s_v3" },
  "instanceCount": { "value": 3 },
  "agentPoolName": { "value": "VS2022-Development" }
}
```

Included Tools:
- Visual Studio 2022 Enterprise
- .NET 6, 7, and 8 SDKs
- Azure CLI, Git, MSBuild
- PowerShell Core, Windows SDK

Scenario 2: Build Tools Only Environment

Use Case: Minimal CI/CD build environment

```
{
  "microsoftImageType": { "value": "vs2022-buildtools" },
  "vmSize": { "value": "Standard_D2s_v3" },
  "instanceCount": { "value": 5 },
  "agentPoolName": { "value": "BuildTools-CICD" }
}
```

Included Tools:
- MSBuild, .NET SDKs
- C++ Build Tools, Windows SDK
- NuGet, Git (minimal)

Scenario 3: Legacy Visual Studio 2019 Environment

Use Case: Legacy applications requiring VS2019

```
{
  "microsoftImageType": { "value": "vs2019-enterprise" },
  "vmSize": { "value": "Standard_D4s_v3" },
  "instanceCount": { "value": 2 },
  "agentPoolName": { "value": "VS2019-Legacy" }
}
```

Included Tools:
- Visual Studio 2019 Enterprise
- .NET Framework 4.8, .NET Core 3.1
- Azure CLI, Git, MSBuild

Scenario 4: Custom Tooling with Chocolatey

Use Case: Windows Server with custom package installations

```
{
  "microsoftImageType": { "value": "windowsserver-2022" },
  "vmSize": { "value": "Standard_D2s_v3" },
  "configureDevOpsAgent": { "value": true }
}
```

PowerShell Script Configuration:

```
$ChocoPackages = @("nodejs", "python", "docker-desktop", "terraform")
$ChocoPackageParams = @{
    "nodejs" = "--version=18.17.0"
    "python" = "--version=3.11.0"
}
```

Troubleshooting Deployment Issues

Common Deployment Failures

1. Marketplace Terms Not Accepted

Error Message:

```
The subscription is not registered for the offer 'visualstudio2022' with publisher 'Mi-
crosoftVisualStudio'
```

Solution:

```
az vm image terms accept \
  --publisher MicrosoftVisualStudio \
  --offer visualstudio2022 \
  --plan vs-2022-ent-latest-ws2022
```

2. Insufficient Permissions

Error Message:

```
Authorization failed. The client does not have authorization to perform action
```

Solutions:
- Verify service principal has Contributor role on resource group
- Check subscription-level permissions
- Ensure proper Azure CLI authentication: `az login`

3. Quota Exceeded

Error Message:

```
Operation could not be completed as it results in exceeding approved quota
```

Solutions:

```
# Check current usage
az vm list-usage --location "East US" --output table

# Check specific quota
az vm list-skus --location "East US" --size Standard_D --output table

# Request quota increase through Azure portal
```

4. Network Connectivity Issues

Error Message:

```
Failed to configure Azure DevOps agent - network connectivity
```

Solutions:
- Verify Azure Firewall allows outbound HTTPS (443) to `*.dev.azure.com`
- Check NSG rules allow VirtualNetwork traffic
- Verify subnet routing to Azure Firewall
- Test from existing VM: `Test-NetConnection dev.azure.com -Port 443`

5. Agent Registration Fails

Error Message:

```
TF400813: The user is not authorized to access this resource
```

Solutions:
- Verify PAT token permissions include Agent Pools (read, manage)
- Check Azure DevOps URL format (must include organization)
- Ensure agent pool exists and is accessible
- Verify token hasn't expired

Debugging Commands

Check Deployment Status

```
# List all deployments
az deployment group list \
  --resource-group $RESOURCE_GROUP \
  --output table

# Get detailed deployment information
az deployment group show \
  --resource-group $RESOURCE_GROUP \
  --name <deployment-name> \
  --output json

# Check deployment operations
az deployment operation group list \
  --resource-group $RESOURCE_GROUP \
  --name <deployment-name> \
  --output table
```

Check VMSS Status

```
# Check VMSS health
az vmss show \
  --resource-group $RESOURCE_GROUP \
  --name vmss-devops-agents \
  --query "provisioningState"

# Check instance status
az vmss list-instances \
  --resource-group $RESOURCE_GROUP \
  --name vmss-devops-agents \
  --query "[].{Name:name, State:provisioningState, Health:latestModelApplied}" \
  --output table

# Check auto-scale settings
az monitor autoscale show \
  --resource-group $RESOURCE_GROUP \
  --name vmss-devops-agents-autoscale
```

Check VM Extensions

```
# List extensions
az vmss extension list \
  --resource-group $RESOURCE_GROUP \
  --vmss-name vmss-devops-agents \
  --output table

# Get extension details
az vmss extension show \
  --resource-group $RESOURCE_GROUP \
  --vmss-name vmss-devops-agents \
  --name ConfigureDevOpsAgent \
  --instance-id 0
```

Access VM for Debugging

```
# Get instance connection info
az vmss list-instance-connection-info \
  --resource-group $RESOURCE_GROUP \
  --name vmss-devops-agents

# If using bastion or jump box
az vmss run-command invoke \
  --resource-group $RESOURCE_GROUP \
  --name vmss-devops-agents \
  --command-id RunPowerShellScript \
  --instance-id 0 \
  --scripts "Get-Content C:\temp\devops-agent-config.log -Tail 20"
```

Log File Locations

Log Type	Location	Description
Agent Configuration	`C:\temp\devops-agent-config.log`	PowerShell script execution log
Agent Summary	`C:\temp\agent-config-summary.txt`	Configuration summary
Agent Service	`C:\agent\_diag\Agent_*.log`	Azure DevOps agent service logs
Chocolatey	`C:\ProgramData\chocolatey\logs\chocolatey.log`	Package installation logs
Windows Event	Event Viewer → Applications and Services Logs	System and application events

Cleanup and Rollback

Complete Environment Cleanup

```
# Delete entire resource group (removes all resources)
az group delete \
  --name $RESOURCE_GROUP \
  --yes \
  --no-wait

# Verify deletion
az group exists --name $RESOURCE_GROUP
```

Partial Cleanup (Keep Network Resources)

```
# Delete only VMSS
az vmss delete \
  --resource-group $RESOURCE_GROUP \
  --name vmss-devops-agents

# Delete load balancer
az network lb delete \
  --resource-group $RESOURCE_GROUP \
  --name lb-devops-agents

# Keep VNet and NSG for reuse
```

Agent Pool Cleanup

```
# Remove agents from pool (if needed)
az pipelines agent delete \
  --pool-id <pool-id> \
  --agent-id <agent-id> \
  --organization https://dev.azure.com/yourorg
```

Next Steps

After successful deployment:

1. **Configure Build Pipelines**: Update existing pipelines to use the new agent pool
2. **Set Up Monitoring**: Configure comprehensive monitoring and alerting
3. **Implement Backup**: Set up backup policies for critical configurations
4. **Security Review**: Conduct security assessment and implement additional controls
5. **Cost Optimization**: Review and optimize resource sizing and auto-scaling rules
6. **Documentation**: Document your specific configuration and customizations

Support and Resources

Documentation Links

- Azure Virtual Machine Scale Sets (https://docs.microsoft.com/en-us/azure/virtual-machine-scale-sets/)
- Azure DevOps Self-hosted Agents (https://docs.microsoft.com/en-us/azure/devops/pipelines/agents/agents)
- Microsoft Visual Studio VM Images (https://docs.microsoft.com/en-us/azure/virtual-machines/windows/using-visual-studio-vm)
- Azure Bicep Documentation (https://docs.microsoft.com/en-us/azure/azure-resource-manager/bicep/)

Getting Help

1. Check the Troubleshooting Guide (TROUBLESHOOTING.md)
2. Review Azure Activity Log for deployment errors
3. Check Azure DevOps service health
4. Open an issue in the repository with:
   - Deployment parameters used
   - Error messages and logs
   - Azure CLI version and environment details

Community Resources

- Azure DevOps Community (https://developercommunity.visualstudio.com/spaces/21/index.html)
- Azure Bicep Community (https://github.com/Azure/bicep)
- Stack Overflow - Azure DevOps (https://stackoverflow.com/questions/tagged/azure-devops)

Congratulations! You've successfully deployed a scalable, secure Azure DevOps build infrastructure using Microsoft prebuilt images.