# Configuring Azure resources for private networking with GitHub-hosted runners

**In this article**



Learn how to configure your Azure Virtual Network (VNET) to use GitHub-hosted runners.

**Note:** Using GitHub-hosted larger runners with an Azure Virtual Network (VNET) is in private beta and subject to change. This feature may not be available to all users.

## About configuring your Azure resources

To use an Azure VNET for private networking, you must configure your Azure resources. You can use the following script to automate the process. For more information about private networking, see "About private networking with GitHub-hosted runners."

## Prerequisites

To configure GitHub Actions for VNET-injection, you must use an Azure account with the Subscription Contributor role and the Network Contributor role. These roles enable you to register the `GitHub.Network` resource provider and delegate the subnet. For more information, see Azure built-in roles in the Azure documentation.

To correctly associate the subnets with the right user, Azure `NetworkSettings` resources must be created in the same subscriptions where virtual networks are created.

To ensure resource availability/data residency, resources must be created in the same Azure region.

After you configure your Azure subscription, share your Azure Subscription ID with your GitHub contact to enroll in the beta.

Obtain your enterprise `databaseId`. For more information, see "Using GraphQL to obtain your `databaseId`."

Save the following `.bicep` file in the same directory location of the script. Name the file `actions-nsg-deployment.bicep`.

YAML

```
@description('NSG for outbound rules')
param location string
param nsgName string = 'actions_NSG'
```

```
resource actions_NSG 'Microsoft.Network/networkSecurityGroups@2017-06-01' = {
  name: nsgName
  location: location
  properties: {
    securityRules: [
      {
        name: 'DenyInternetOutBoundOverwrite'
        properties: {
          protocol: '*'
          sourcePortRange: '*'
          destinationPortRange: '*'
          sourceAddressPrefix: '*'
          destinationAddressPrefix: 'Internet'
          access: 'Deny'
          priority: 400
          direction: 'Outbound'
        }
      }
      {
        name: 'AllowVnetOutBoundOverwrite'
        properties: {
          protocol: 'TCP'
          sourcePortRange: '*'
          destinationPortRange: '443'
          sourceAddressPrefix: '*'
          destinationAddressPrefix: 'VirtualNetwork'
          access: 'Allow'
          priority: 200
          direction: 'Outbound'
          destinationAddressPrefixes: []
        }
      }
      {
        name: 'AllowAzureCloudOutBound'
        properties: {
          protocol: 'TCP'
          sourcePortRange: '*'
          destinationPortRange: '443'
          sourceAddressPrefix: '*'
          destinationAddressPrefix: 'AzureCloud'
          access: 'Allow'
          priority: 210
          direction: 'Outbound'
          destinationAddressPrefixes: []
        }
      }
      {
        name: 'AllowInternetOutBoundGitHub'
        properties: {
          protocol: 'TCP'
          sourcePortRange: '*'
          destinationPortRange: '443'
          sourceAddressPrefix: '*'
          access: 'Allow'
          priority: 220
          direction: 'Outbound'
          destinationAddressPrefixes: [
            '140.82.112.0/20'
            '142.250.0.0/15'
            '143.55.64.0/20'
            '192.30.252.0/22'
            '185.199.108.0/22'
          ]
        }
      }
      {
        name: 'AllowInternetOutBoundMicrosoft'
        properties: {
          protocol: 'TCP'
          sourcePortRange: '*'
```

```
          destinationPortRange: '443'
          sourceAddressPrefix: '*'
          access: 'Allow'
          priority: 230
          direction: 'Outbound'
          destinationAddressPrefixes: [
            '13.64.0.0/11'
            '13.96.0.0/13'
            '13.104.0.0/14'
            '20.33.0.0/16'
            '20.34.0.0/15'
            '20.36.0.0/14'
            '20.40.0.0/13'
            '20.48.0.0/12'
            '20.64.0.0/10'
            '20.128.0.0/16'
            '52.224.0.0/11'
            '204.79.197.200'
          ]
        }
      }
    {
        name: 'AllowInternetOutBoundCannonical'
        properties: {
          protocol: 'TCP'
          sourcePortRange: '*'
          destinationPortRange: '443'
          sourceAddressPrefix: '*'
          destinationAddressPrefix: '185.125.188.0/22'
          access: 'Allow'
          priority: 240
          direction: 'Outbound'
          destinationAddressPrefixes: []
        }
      }
    ]
  }
}
```

## Using GraphQL to obtain your `databaseId`

Use the following GraphQL query to retrieve your enterprise `databaseId`. You will use the enterprise `databaseId` for the value of the `DATABASE_ID` environment variable in the next step. For more information on working with GraphQL, see "Forming calls with GraphQL."

```
query(
  $slug: String!
){
  enterprise (slug: $slug)
  {
    slug
    databaseId
  }
}
```

| Query variable | Description |
| --- | --- |
| `slug` | The slug for your enterprise account, which you can identify by looking at the URL for your enterprise, `https://github.com/enterprises/SLUG`. |

## Using a script to configure your Azure resources

Use the following script to set up a subnet with VNET-injection in Azure. The script creates all resources in the same resource group.

To use the script, fill in the placeholder environment variable values with the actual values and run the script from a bash shell or Windows Subsystem for Linux.

Bash

```
#!/bin/bash

# This script creates the following resources in the specified subscription:
# - Resource group
# - Network Security Group rules
# - Virtual network (vnet) and subnet
# - Network Settings with specified subnet and GitHub Enterprise databse ID
#
# It also registers the `GitHub.Network` resource provider with the subscription,
# delegates the created subnet to the Actions service via the
`GitHub.Network/NetworkSettings`
# resource type, and applies the NSG rules to the created subnet.

# stop on failure
set -e

#set environment
AZURE_LOCATION=YOUR_AZURE_LOCATION
SUBSCRIPTION_ID=YOUR_SUBSCRIPTION_ID
RESOURCE_GROUP_NAME=YOUR_RESOURCE_GROUP_NAME
VNET_NAME=YOUR_VNET_NAME
SUBNET_NAME=YOUR_SUBNET_NAME
NSG_NAME=YOUR_NSG_NAME
NETWORK_SETTINGS_RESOURCE_NAME=YOUR_NETWORK_SETTINGS_RESOURCE_NAME
DATABASE_ID=YOUR_DATABASE_ID

echo login to Azure
. az login --output none

echo set account context $SUBSCRIPTION_ID
. az account set --subscription $SUBSCRIPTION_ID

echo Register resource provider GitHub.Network
. az provider register --namespace GitHub.Network

echo Create resource group $RESOURCE_GROUP_NAME at $AZURE_LOCATION
. az group create --name $RESOURCE_GROUP_NAME --location $AZURE_LOCATION

echo Create NSG rules deployed with 'actions-nsg-deployment.bicep' file
. az deployment group create --resource-group $RESOURCE_GROUP_NAME --template-
file ./actions-nsg-deployment.bicep --parameters location=$AZURE_LOCATION
nsgName=$NSG_NAME

echo Create vnet $VNET_NAME and subnet $SUBNET_NAME
. az network vnet create --resource-group $RESOURCE_GROUP_NAME --name $VNET_NAME
--address-prefix 10.0.0.0/16 --subnet-name $SUBNET_NAME --subnet-prefixes
10.0.0.0/24

echo Delegate subnet to GitHub.Network/networkSettings and apply NSG rules
. az network vnet subnet update --resource-group $RESOURCE_GROUP_NAME --name
$SUBNET_NAME --vnet-name $VNET_NAME --delegations GitHub.Network/networkSettings
--network-security-group $NSG_NAME

echo Create network settings resource $NETWORK_SETTINGS_RESOURCE_NAME
. az resource create --resource-group $RESOURCE_GROUP_NAME  --name
$NETWORK_SETTINGS_RESOURCE_NAME --resource-type GitHub.Network/networkSettings --
properties "{ \"location\": \"$AZURE_LOCATION\", \"properties\" : {
\"subnetId\":
\"/subscriptions/$SUBSCRIPTION_ID/resourceGroups/$RESOURCE_GROUP_NAME/providers/Mic
 \"organizationId\": \"$DATABASE_ID\" }}" --is-full-object --api-version 2023-03-
15-beta
```

```
echo To clean up and delete resources run the following command:
echo az group delete --resource-group $RESOURCE_GROUP_NAME
```

The script will return the full payload for the created resource. The `GitHubId` hash value returned in the payload for the created resource is the network settings resource ID you will use in the next steps while configuring VNET settings with GitHub. For more information, see "Configuring your GitHub settings for use with Azure Virtual Network."