Azure IPI Quickstart Deployment Guide

This document walks through how to deploy an OpenShift Installer Provisioned Infrastructure (IPI) Quickstart architecture on Azure.

OpenShift IPI is an alternate OpenShift deployment method to Azure Redhat OpenShift (ARO) on Azure. Using IPI gives more flexibility to the installation including choice of OpenShift version.

Goals for Guide

- Be able to deploy a quickstart OpenShift IPI architecture on Azure
- Understand the input variables and their impact on the deployment

Prerequisites

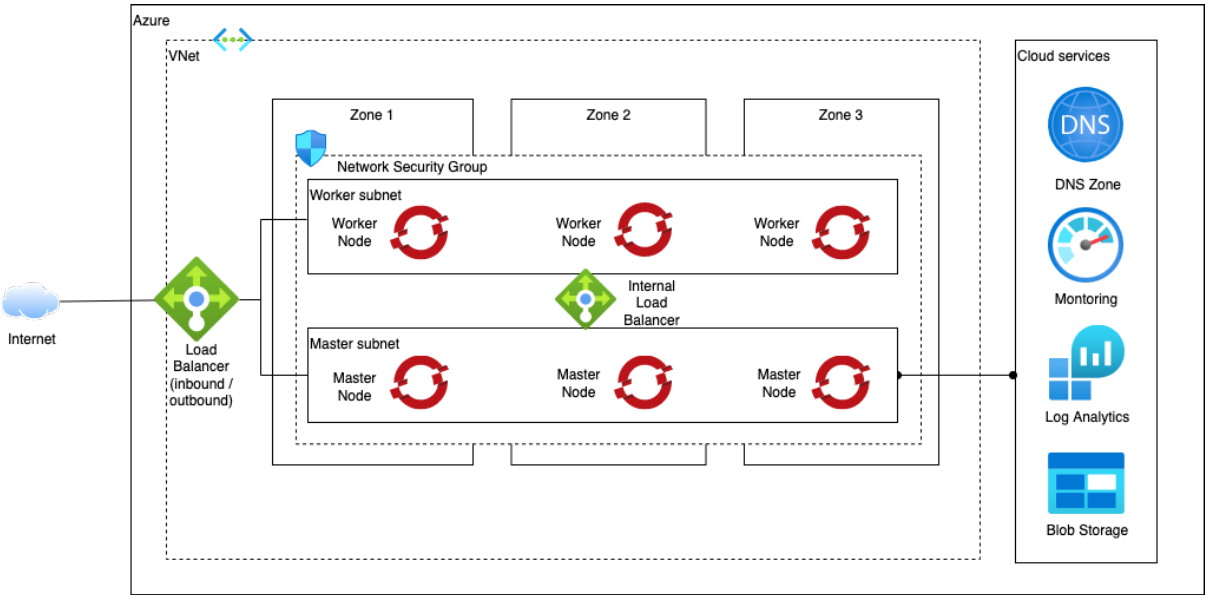
- Access to an Azure account with "Owner" and "User Access Administrator" roles in an Azure Subscription. The user must be able to create a service principal per the below prerequisite.
- Install Azure CLI. This is required to setup the service principal per the below instructions, not to deploy OpenShift. So if you already have a service principal or create the service principal via the Azure portal, than the Azure CLI is not required.
- A configured DNS subdomain in Azure (refer to Appendix A for details)
- A service principal with proper IAM roles (refer to Appendix B for details)
- Have an OpenShift installer pull secret
- (Optional for MacOS) Install and start Colima to run the automation in a local bootstrapped container image.

```
brew install docker colima

colima start
```

- (Optional for Linux) Install and run docker to run the automation in a local bootstrapped container image.

Reference Architecture

This guide will deploy the following architecture onto an Azure subscription.



The automation will build the Azure VNet, subnets, network security group, load balancers and other components to obtain a functioning OpenShift cluster with persistent storage and developer tools.

The following layers are available:

Layer Name	Layer Description	Provided Resources
105 – Azure OpenShift IPI	This layer provisions the Azure infrastructure and OpenShift. It will create a new VNet and other networking components required to support the OpenShift cluster. An existing registered DNS zone for the required domain name is required (refer to prerequisites).	**Network** • Virtual network • VNet Master and Worker Subnets • Network Security Group • Inbound and outbound Load Balancer • Red Hat OpenShift cluster
110 – Ingress Certificate	This layer replaces the self-signed certificates with one of two options, either auto-generated ones from LetsEncrypt or supplied certificates. This allows web browser console access to the cluster. Note that this layer invalidates the access key in the existing kubeconfig. It is necessary to get a new access key from the console to login at the command line after applying this layer.	**Acme Certificates** • API Certificate • Apps certificate • OpenShift Cluster Update **BYO Certificates** • OpenShift Cluster Update
200 – OpenShift Gitops	Provisions CI/CD tools into the cluster including a GitOps repository.	**Software** • OpenShift GitOps (ArgoCD) • OpenShift Pipelines (Tekton) • Sealed Secrets (Kubeseal) • GitOps Repository

210 – Azure Storage	The storage layer has two options - default or Portworx. The default option uses Azure's storage for OpenShift persistent volumes. For quickstart, this is Premium_LRS. Other options can be configured post implementation through the OpenShift console. The Portworx option implements either a Portworx Essentials or Portworx Enterprise deployment onto the OpenShift cluster. The type of Portworx deployment is determined by the supplied Portworx specification file.	**Default** • Azure storage class **Portworx** • Portworx operator • Portworx storage classes
220 – Dev Tools	This layer provisions standard continuous integration pipelines to integrate with the software development lifecycle.	**Software** • Artifactory • IBM Developer Dashboard • Pact Broker • Sonarqube • Tekton Resources

Step by Step Guide

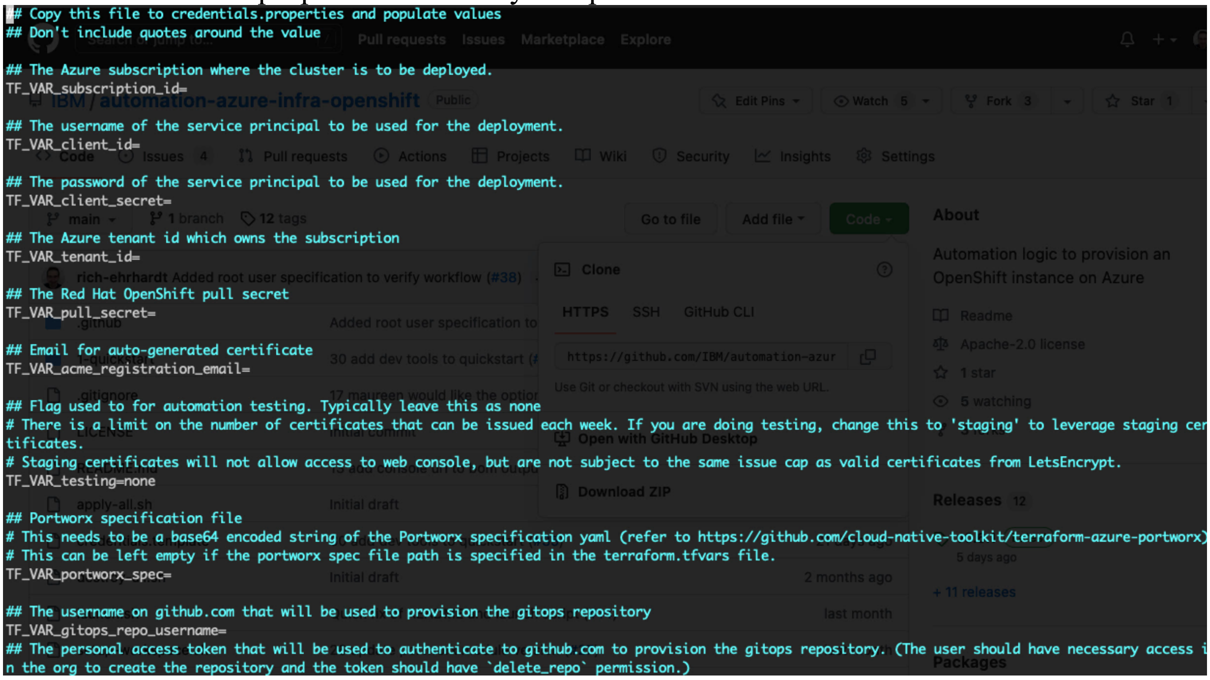
1. Clone the automation repository to your laptop or a secure terminal with access to internet.

```
cd ~/Documents
git clone https://github.com/IBM/automation-azure-infra-openshift.git
cd ~/Document/automation-azure-infra-openshift
```

2. Copy the credentials.template to credentials.properties.

```
cp credentials.template credentials.properties
```

3. Edit the credentials.properties file with your specific details.

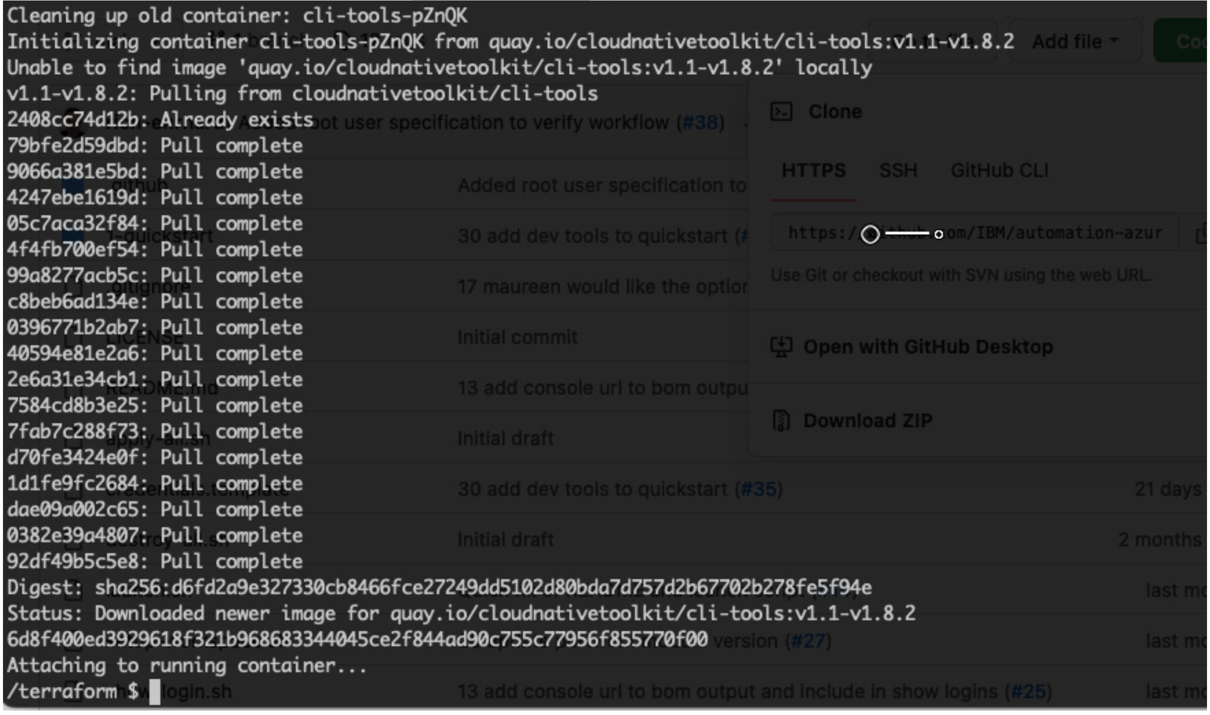


```
## Copy this file to credentials.properties and populate values
## Don't include quotes around the value

## The Azure subscription where the cluster is to be deployed.
TF_VAR_subscription_id=

## The username of the service principal to be used for the deployment.
TF_VAR_client_id=

## The password of the service principal to be used for the deployment.
TF_VAR_client_secret=

## The Azure tenant id which owns the subscription
TF_VAR_tenant_id=

## The Red Hat OpenShift pull secret
TF_VAR_pull_secret=

## Email for auto-generated certificate
TF_VAR_acme_registration_email=

## Flag used to for automation testing. Typically leave this as none
# There is a limit on the number of certificates that can be issued each week. If you are doing testing, change this to 'staging' to leverage staging cer
tificates.
# Staging certificates will not allow access to web console, but are not subject to the same issue cap as valid certificates from LetsEncrypt.
TF_VAR_testing=none

## Portworx specification file
# This needs to be a base64 encoded string of the Portworx specification yaml (refer to https://github.com/cloud-native-toolkit/terraform-azure-portworx)
# This can be left empty if the portworx spec file path is specified in the terraform.tfvars file.
TF_VAR_portworx_spec=

## The username on github.com that will be used to provision the gitops repository
TF_VAR_gitops_repo_username=
## The personal access token that will be used to authenticate to github.com to provision the gitops repository. (The user should have necessary access i
n the org to create the repository and the token should have `delete_repo` permission.)
```

- **TF_VAR_subscription_id** - The Azure subscription id where the cluster will be deployed
- **TF_VAR_tenant_id** - The Azure tenant id that owns the subscription
- **TV_VAR_client_id** - The id of the service principal with Owner and User Administrator access to the subscription for cluster creation
- **TV_VAR_client_secret** - The password of the service principal with Owner and User Administrator access to the subscription for cluster creation
- **TV_VAR_pull_secret** - The contents of the Red Hat OpenShift pull secret downloaded in the prerequsite steps
- **TF_VAR_acme_registration_email** - (Optional) If using an auto-generated ingress certificate, this is the email address with which to register the certificate with LetsEncrypt.
- **TF_VAR_testing** - This value is used to determine whether testing or staging variables should be utilised. Lease as none for production deployments. A value other than none will request in a non-production deployment.
- **TF_VAR_portworx_spec** - A base64 encoded string of the Portworx specificatin yaml file. If left blank and using Portworx, ensure you specify the path to the Portworx specification yaml file in the terraform.tfvars file. For a Portworx

implementation, either the portworx_spec or the portworx_spec_file values must be specified. If neither if specified, Portworx will not implement correctly.

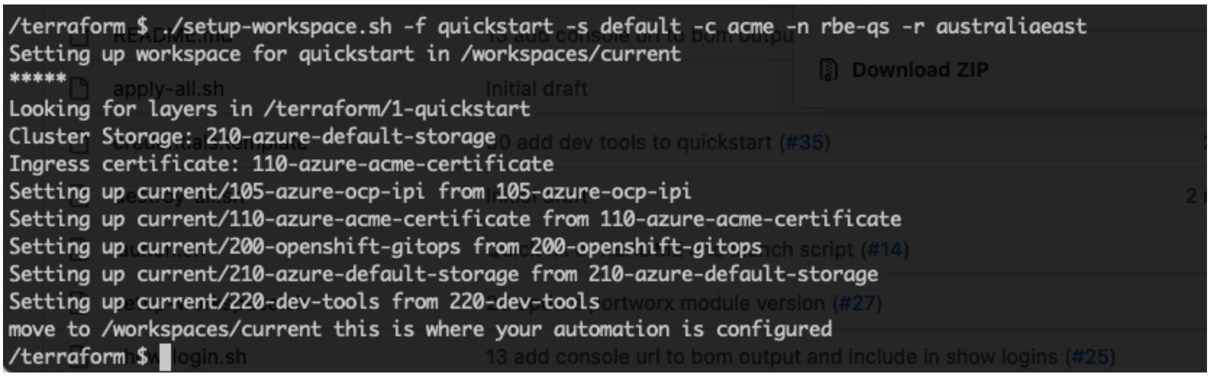
4. For the container approach, run `./launch.sh`



```
Cleaning up old container: cli-tools-pZnQK
Initializing container cli-tools-pZnQK from quay.io/cloudnativetoolkit/cli-tools:v1.1-v1.8.2
Unable to find image 'quay.io/cloudnativetoolkit/cli-tools:v1.1-v1.8.2' locally
v1.1-v1.8.2: Pulling from cloudnativetoolkit/cli-tools
2408cc74d12b: Already exists
79bfe2d59dbd: Pull complete
9066a381e5bd: Pull complete
4247ebe1619d: Pull complete
05c7aca32f84: Pull complete
4f4fb700ef54: Pull complete
99a8277acb5c: Pull complete
c8beb6ad134e: Pull complete
0396771b2ab7: Pull complete
40594e81e2a6: Pull complete
2e6a31e34cb1: Pull complete
7584cd8b3e25: Pull complete
7fab7c288f73: Pull complete
d70fe3424e0f: Pull complete
1d1fe9fc2684: Pull complete
dae09a002c65: Pull complete
0382e39a4807: Pull complete
92df49b5c5e8: Pull complete
Digest: sha256:d6fd2a9e327330cb8466fce27249dd5102d80bda7d757d2b67702b278fe5f94e
Status: Downloaded newer image for quay.io/cloudnativetoolkit/cli-tools:v1.1-v1.8.2
6d8f400ed3929618f321b968683344045ce2f844ad90c755c77956f855770f00
Attaching to running container...
/terraform $
```

5. Create a working copy of the terraform code by running ./setup-workspace.sh . The script makes a copy of the terraform in /workspaces/current and set up a "terraform.tfvars" file populated with default values. The script can be run interactively by just running ./setup-workspace.sh or by providing command line parameters as specified below.

```
Usage: setup-workspace.sh [-f FLAVOR] [-s STORAGE] [-c CERT_TYPE] [-r REGION] [-n
PREFIX_NAME]


where:

  - **FLAVOR** - the type of deployment `quickstart`, `standard` or `advanced`. If not
provided, will default to quickstart.

  - **STORAGE** - The storage provider. Possible options are `portworx` or `odf`. If
not provided as an argument, a prompt will be shown.

  - **CERT_TYPE** - The type of ingress certificate to apply. Possible options are
`acme` or `byo`. Acme will obtain certificates from LetsEncrypt for the new cluster.
BYO requires providing the paths to valid certificates in the **terraform.tfvars**
file.

  - **REGION** - the Azure location where the infrastructure will be provided
([available regions](https://docs.microsoft.com/en-us/azure/availability-zones/az-
overview)). Codes for each location can be obtained from the CLI using,

        az account list-locations -o table

    If not provided the value defaults to `eastus`

  - **PREFIX_NAME** - the name prefix that should be added to all the resources. If
not provided a prefix will not be added.
```



```
/terraform $ ./setup-workspace.sh -f quickstart -s default -c acme -n rbe-qs -r australiaeast
Setting up workspace for quickstart in /workspaces/current
*****
Looking for layers in /terraform/1-quickstart
Cluster Storage: 210-azure-default-storage
Ingress certificate: 110-azure-acme-certificate
Setting up current/105-azure-ocp-ipi from 105-azure-ocp-ipi
Setting up current/110-azure-acme-certificate from 110-azure-acme-certificate
Setting up current/200-openshift-gitops from 200-openshift-gitops
Setting up current/210-azure-default-storage from 210-azure-default-storage
Setting up current/220-dev-tools from 220-dev-tools
move to /workspaces/current this is where your automation is configured
/terraform $ 
```

6. Change to the /workspaces/current directory (if using the container approach)

```
cd /workspaces/current
```

7. Edit the terraform.tfvars file. For example, with vi terraform.tfvars. At a minimum, modify the following values to align with the environment.
   - **base_domain_name** - the full subdomain delegated to Azure in the DNS zone (for example ocp.azure.example.com)
   - **resource_group_name** - the Azure resource group where the DNS zone has been defined
8. Run all the terraform layers automatically.

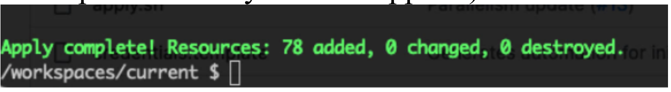
```
./apply-all.sh
```

9. Alternately, run each layer in turn by changing to that layer's directory and running:

```
cd <directory>
terragrunt init
terragrunt apply —auto-approve
```

This will kick-off the terraform code application.

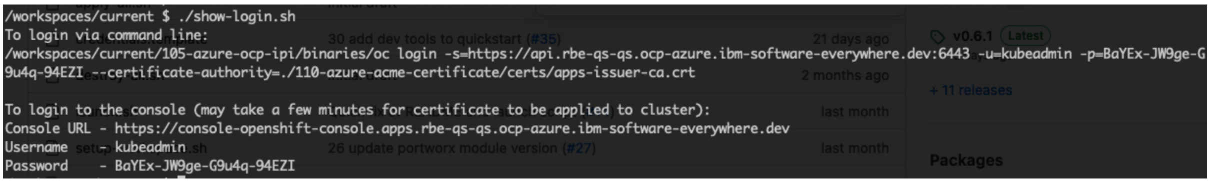
At the completion, it should display such as the following (the actual number of resources will depend upon which layer is last applied).



```
Apply complete! Resources: 78 added, 0 changed, 0 destroyed.
/workspaces/current $ 
```

Once completed, obtain the console and cli login details by running ./show-login.sh from the working directory (like /workspaces/current).

```
cd /workspaces/current
./show-login.sh
```



```
/workspaces/current $ ./show-login.sh
To login via command line:
/workspaces/current/105-azure-ocp-ipi/binaries/oc login -s=https://api.rbe-qs-qs.ocp-azure.ibm-software-everywhere.dev:6443 -u=kubeadmin -p=BaYEx-JW9ge-G
9u4q-94EZI --certificate-authority=./110-azure-acme-certificate/certs/apps-issuer-ca.crt

To login to the console (may take a few minutes for certificate to be applied to cluster):
Console URL - https://console-openshift-console.apps.rbe-qs-qs.ocp-azure.ibm-software-everywhere.dev
Username    - kubeadmin
Password    - BaYEx-JW9ge-G9u4q-94EZI
```

Appendix A – Configure DNS Subdomain

1. Buy or have an existing domain

There are a few suppliers which provide domain registration services including GoDaddy (godaddy.com). Whichever one is chosen needs to be able to delegate DNS lookup to Azure.

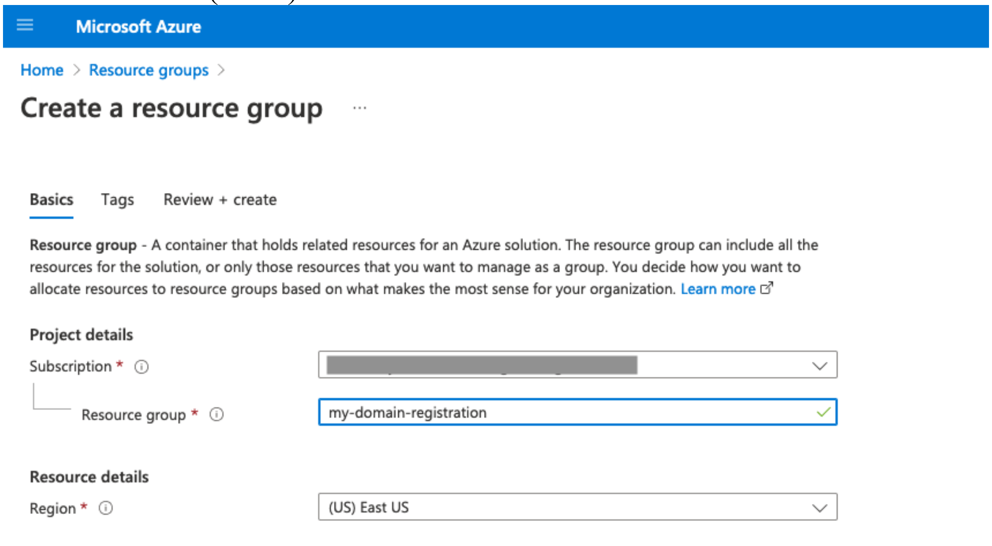
2. Decide on a subdomain of the existing domain for OpenShift clusters

Create a subdomain of the one registered in the prior step. This will be used for all clusters created in the Azure subscription. For example, if the domain in the prior step were mydomain.com, then a subdomain could be my-azure.mydomain.com. Clusters created under this subdomain would then have the format:

```
https://api.<cluster-name>.my-azure.mydomain.com:6443/
```

3. Create a new resource group in the Azure subscription

This resource group will be used to store the domain registrations under the subdomain. It is separate to the resource group created for the OpenShift clusters or virtual network (VNet).



Create a resource group ···

Resource group - A container that holds related resources for an Azure solution. The resource group can include all the resources for the solution, or only those resources that you want to manage as a group. You decide how you want to allocate resources to resource groups based on what makes the most sense for your organization. Learn more

Project details

Subscription * : [redacted]

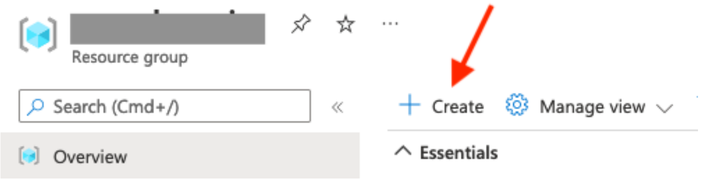
Resource group * : my-domain-registration

Resource details

Region * : (US) East US

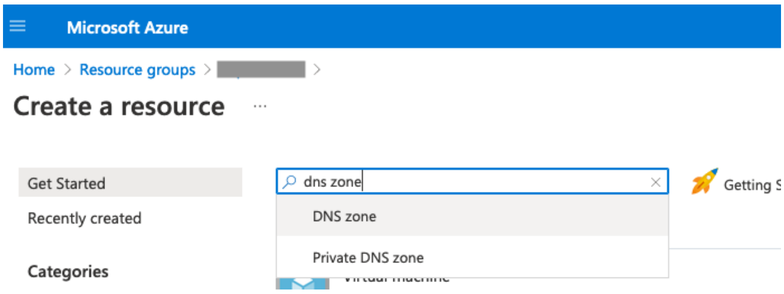
4. Create a DNS zone in the resource group using the subdomain and existing domain

From within the newly created domain resource group select “create” at the top left.



[redacted]

Search for DNS Zone

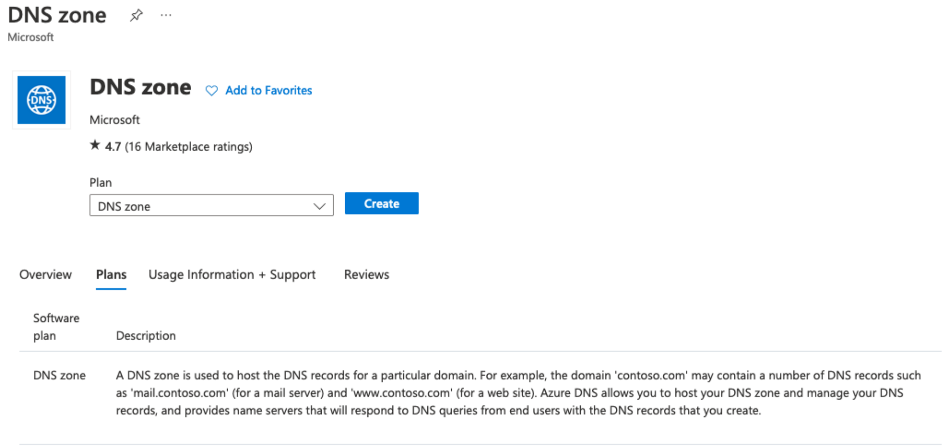


dns zone

DNS zone

Private DNS zone

Then choose DNS zone as the plan.



DNS zone

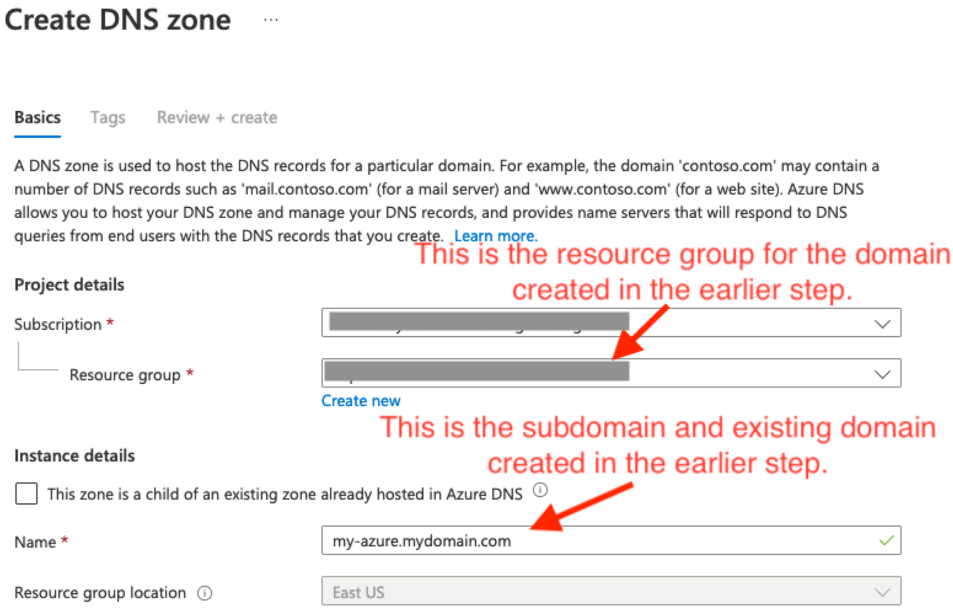
Microsoft

★ 4.7 (16 Marketplace ratings)

Plan: DNS zone

Software plan	Description
DNS zone	A DNS zone is used to host the DNS records for a particular domain. For example, the domain 'contoso.com' may contain a number of DNS records such as 'mail.contoso.com' (for a mail server) and 'www.contoso.com' (for a web site). Azure DNS allows you to host your DNS zone and manage your DNS records, and provides name servers that will respond to DNS queries from end users with the DNS records that you create.

Complete the DNS zone details, review and create.



A DNS zone is used to host the DNS records for a particular domain. For example, the domain 'contoso.com' may contain a number of DNS records such as 'mail.contoso.com' (for a mail server) and 'www.contoso.com' (for a web site). Azure DNS allows you to host your DNS zone and manage your DNS records, and provides name servers that will respond to DNS queries from end users with the DNS records that you create. Learn more.

Project details

Subscription *: [redacted]

Resource group *: [redacted]

This is the resource group for the domain created in the earlier step.

Instance details

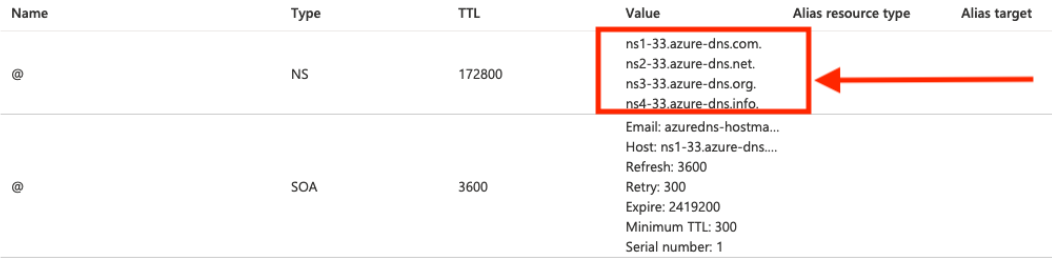
□ This zone is a child of an existing zone already hosted in Azure DNS

This is the subdomain and existing domain created in the earlier step.

Name *: my-azure.mydomain.com

Resource group location: East US

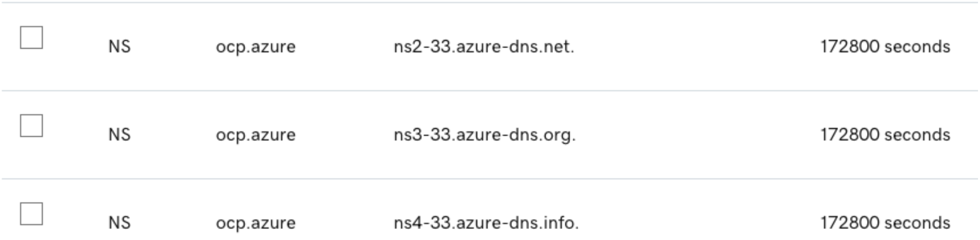
This will create a new name service (NS) and Start of Authority (SOA) record for the subdomain. Note the name server details as these will be required to delegate access from your domain name service.



Name	Type	TTL	Value	Alias resource type	Alias target
@	NS	172800	ns1-33.azure-dns.com. ns2-33.azure-dns.net. ns3-33.azure-dns.org. ns4-33.azure-dns.info.		
@	SOA	3600	Email: azuredns-hostma... Host: ns1-33.azure-dns.... Refresh: 3600 Retry: 300 Expire: 2419200 Minimum TTL: 300 Serial number: 1		

5. Delegate access to Azure for your domain provider

The next step needs to be done from your domain provider and the process will vary depending upon the provider. For example, in GoDaddy, select “DNS” next to the domain name or “Manage DNS” in the domain details page, then “Add” a NS details for each of the Azure domain servers specified in the prior step so that you end up with the entries for azure-dns.com., azure-dns.net. , azure-dns.org. and azure-dns.info. similar to the following:



□	NS	ocp.azure	ns2-33.azure-dns.net.	172800 seconds
□	NS	ocp.azure	ns3-33.azure-dns.org.	172800 seconds
□	NS	ocp.azure	ns4-33.azure-dns.info.	172800 seconds

Congratulations! The DNS Zone is now ready to be used with an OpenShift Installer Provisioned Infrastructure installation. During the installation, be sure to include the resource group created for the DNS Zone as the domain resource group and the DNS subdomain as the base domain. For example, if the resource group created were “my-domain-resource” and the subdomain “my-azure.mydomain.com”, then the variables input to the installation in the terraform.tfvars file would be:

```
## Resource group name containing the base domain
resource_group_name="my-domain-resource"


## Base domain name (e.g. myclusters.mydomain.com)

base_domain_name="my-azure.mydomain.com"
```

Appendix B – Create a service principal

1. Create the service principal account if it does not already exist:

```
az ad sp create-for-rbac --role Contributor \
--name <service_principal_name> \
--scopes /subscriptions/$SUBSCRIPTION_ID
```

   where $SUBSCRIPTION_ID is the Azure subscription where the cluster is to be deployed and `service_principal_name` is the name to be assigned to the service principal. Make a copy of the details provided as they will be needed for the credentials.properties file during the installation.

```
"addId":"<this is the CLIENT_ID value>",
"displayName":"<service principal name>",
"password":"<this is the CLIENT_SECRET value>",
"tenant":"<this is the TENANT_ID value>"
```

2. Assign contributor and User Access Administrator roles to the service principal if not already in place.

```
az role assignment create \
--role "User Access Administrator" \
--assignee-object-id \
$(az ad sp list –filter “appId eq ‘$CLIENT_ID’” | jq ‘.[0].id’ -r)
```

   where $CLIENT_ID is the appId of the service principal created in the prior step.

Appendix C – Troubleshooting

Cluster installation log

To check the installation logs, navigate to the OpenShift install directory. If you are using the bootstrapped container approach, then this directory can be found:

```
cd /workspaces/current/105-azure-ocp-ipi/install
```

The OpenShift install log is called `.openshift-install.log`.

```
view .openshift-install.log
```

Manual cluster clean up

In the event that terraform did not complete the openshift installation, you may be left with a half-built cluster. To remove this, perform the following steps from the Azure portal:

1. Remove the CNAME and A records from the DNS Zone in the domain resource group. Depending upon when the cluster install failed, there will be only the CNAME for `api.<cluster_name>` or this and the A record for `*.apps.<cluster_name>`
2. Remove the resource group containing the failed OpenShift cluster. Navigate to the resource group (`Home -> Resource groups -> <resource-group-name>`). Note that the resource group name will have a random number appended to the cluster name. For example, if the cluster name were `failed-qs`, then the resource group name would `failed-qs-<5_digit_random>-rg`. Then select the `Delete resource group` button at the top and enter the resource group name as instructed, then click delete at the bottom