***Install Cloud Foundry on Azure Kubernetes Clusters using Korifi***

Korifi is a Cloud Foundry implementation that runs on Kubernetes. It is a community-driven project that aims to provide a simple and efficient way to deploy and manage cloud-native applications on Kubernetes. Korifi aims to preserve the classic Cloud Foundry developer experience when using Kubernetes. Developers can still use the cf push command to deploy applications to Korifi, and they can still use the Cloud Foundry CLI to manage their applications. Here are some of the benefits of using Korifi.

* It is a simple and efficient way to deploy and manage cloud-native applications on Kubernetes.
* It preserves the classic Cloud Foundry developer experience.
* It takes advantage of the many features and capabilities of Kubernetes.
* Cost Effectiveness

***Prerequisites:***

To begin the installation, please install the following tools to start.

* K8s cluster
* kubectl
* cf CLI v8
* Helm
* azure cli
* Azure Container Registry
* cert-Manager
* kpack
* Contour

Azure also provides a container registry that can be used to upload images built when using Korifi. Azure Container Registry (ACR) is a managed Docker registry service that lets you store and manage your container images and artefacts. It serves as a private (or public) repository for storing and managing container images. ACR seamlessly integrates with other Azure services like Azure Kubernetes Service (AKS) and Azure DevOps, allowing developers to use container images stored in ACR for deployments. For this installation, we will make use of a registry. Follow the steps here to create a container registry.

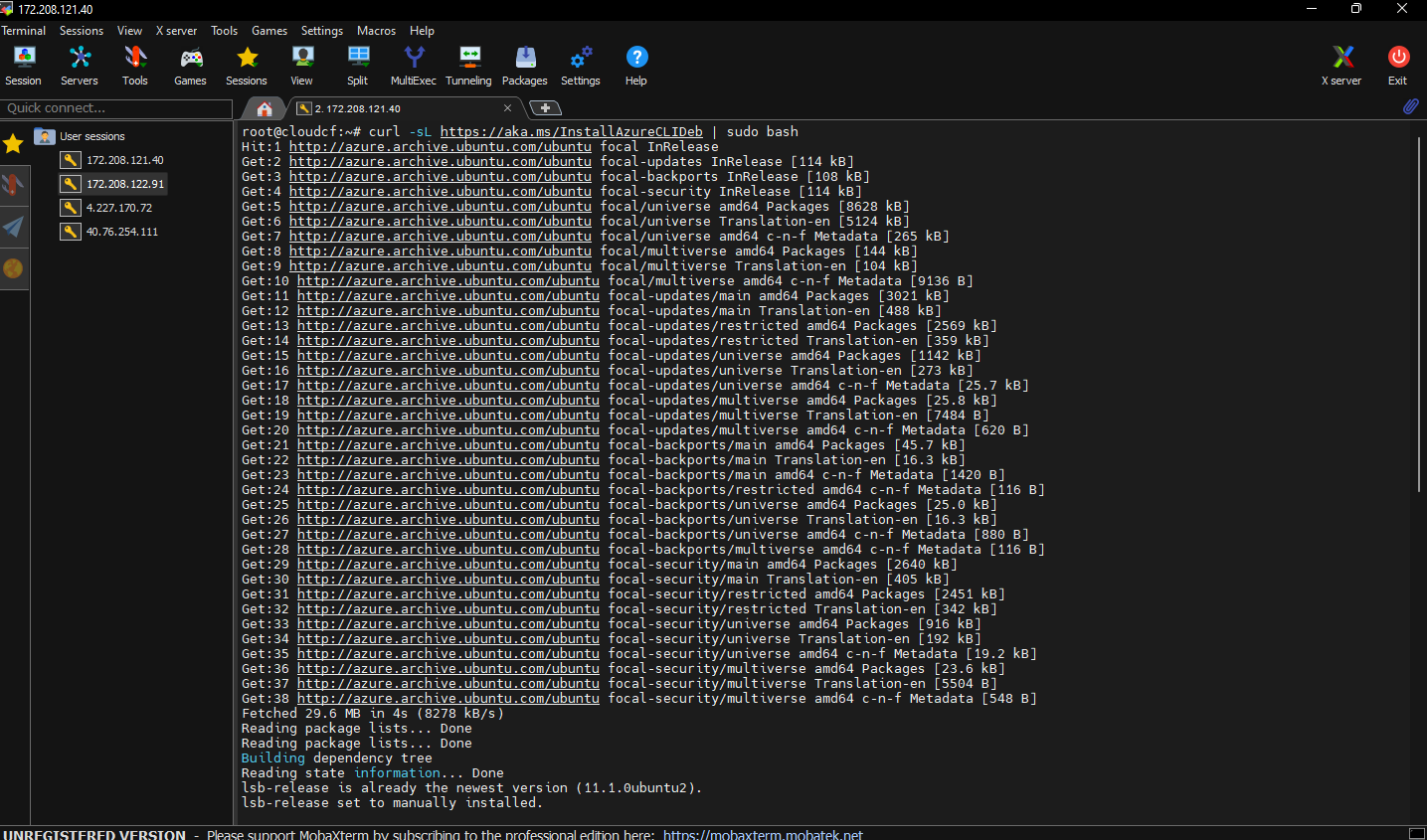
***Installation:***

1. First, Install azure cli:

<https://learn.microsoft.com/en-us/cli/azure/install-azure-cli-linux?pivots=apt>

Run the command on your VM

curl -sL https://aka.ms/InstallAzureCLIDeb | sudo bash



1. Create a Kubernetes cluster. When creating the cluster, make sure that it belongs to the same resource group as the container registry.

Once a Kubernetes cluster has been created, connect to it. Using following command.

az account set --subscription <SubscriptionID>

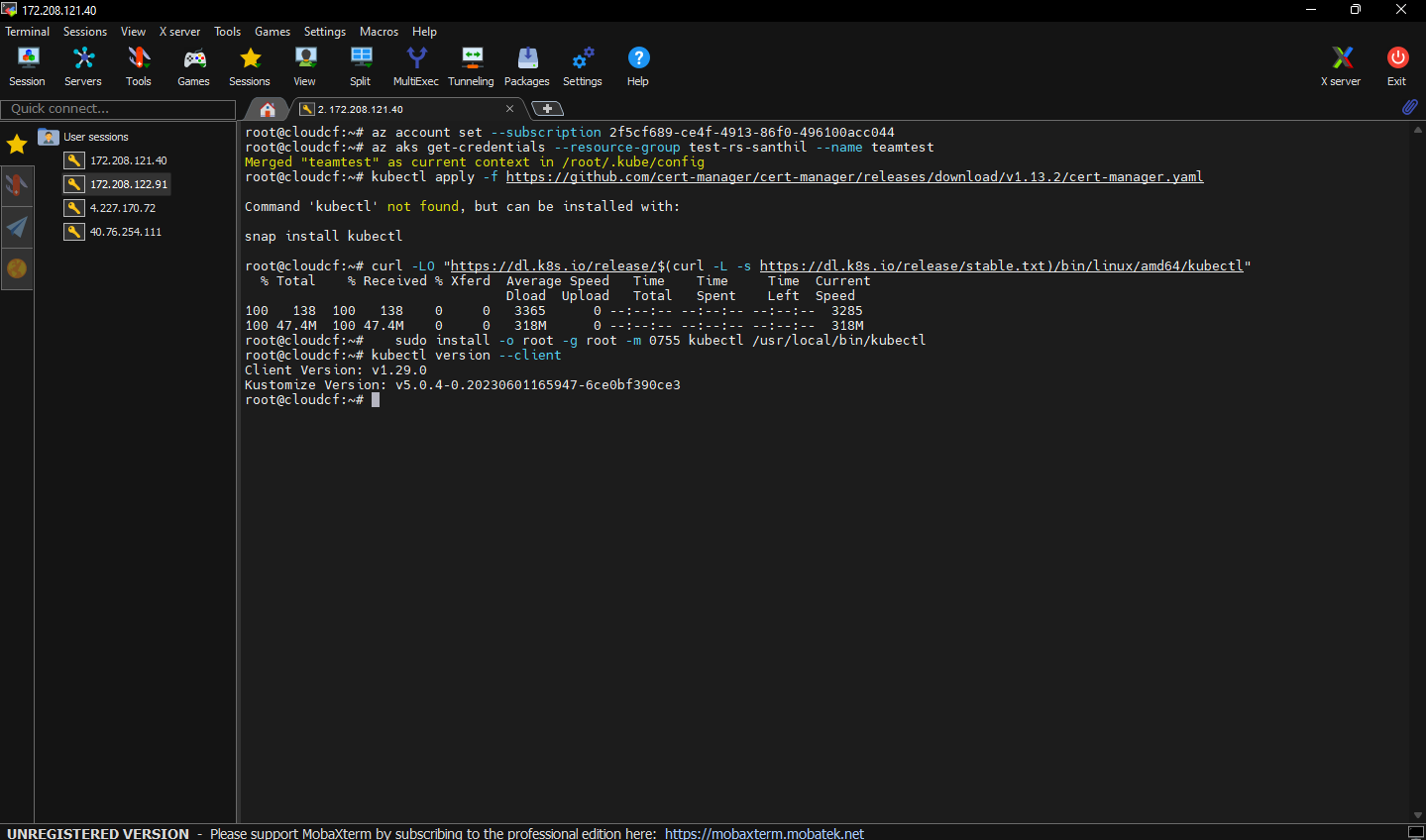
az aks get-credentials --resource-group <RGName> --name <AKSName>

A screen shot of a computer

Description automatically generated

1. Next, install the following dependencies: kubectl, cf cli, cert-Manager, kpack, and Contour.

Steps to install kubectl

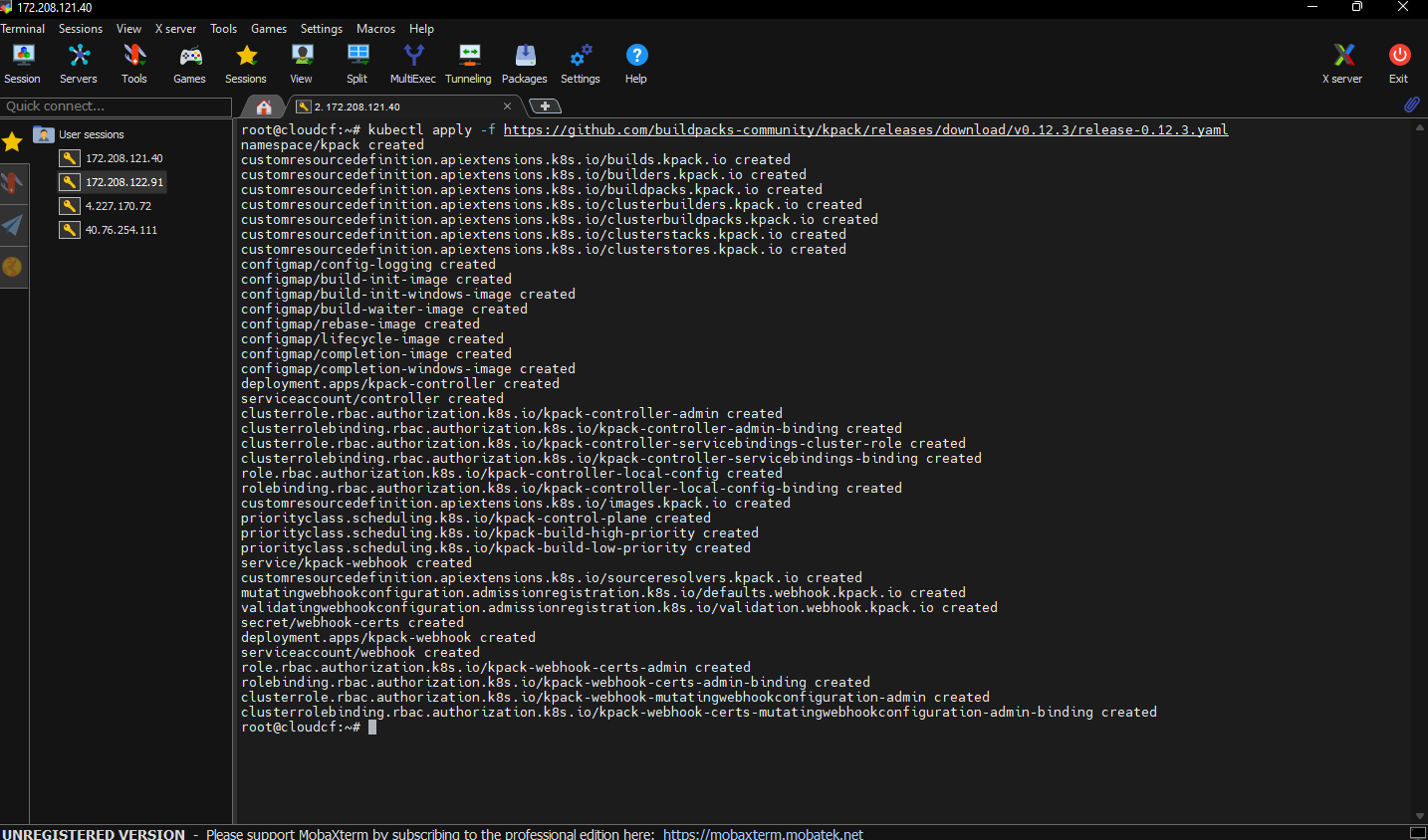
 <https://kubernetes.io/docs/tasks/tools/install-kubectl-linux/>

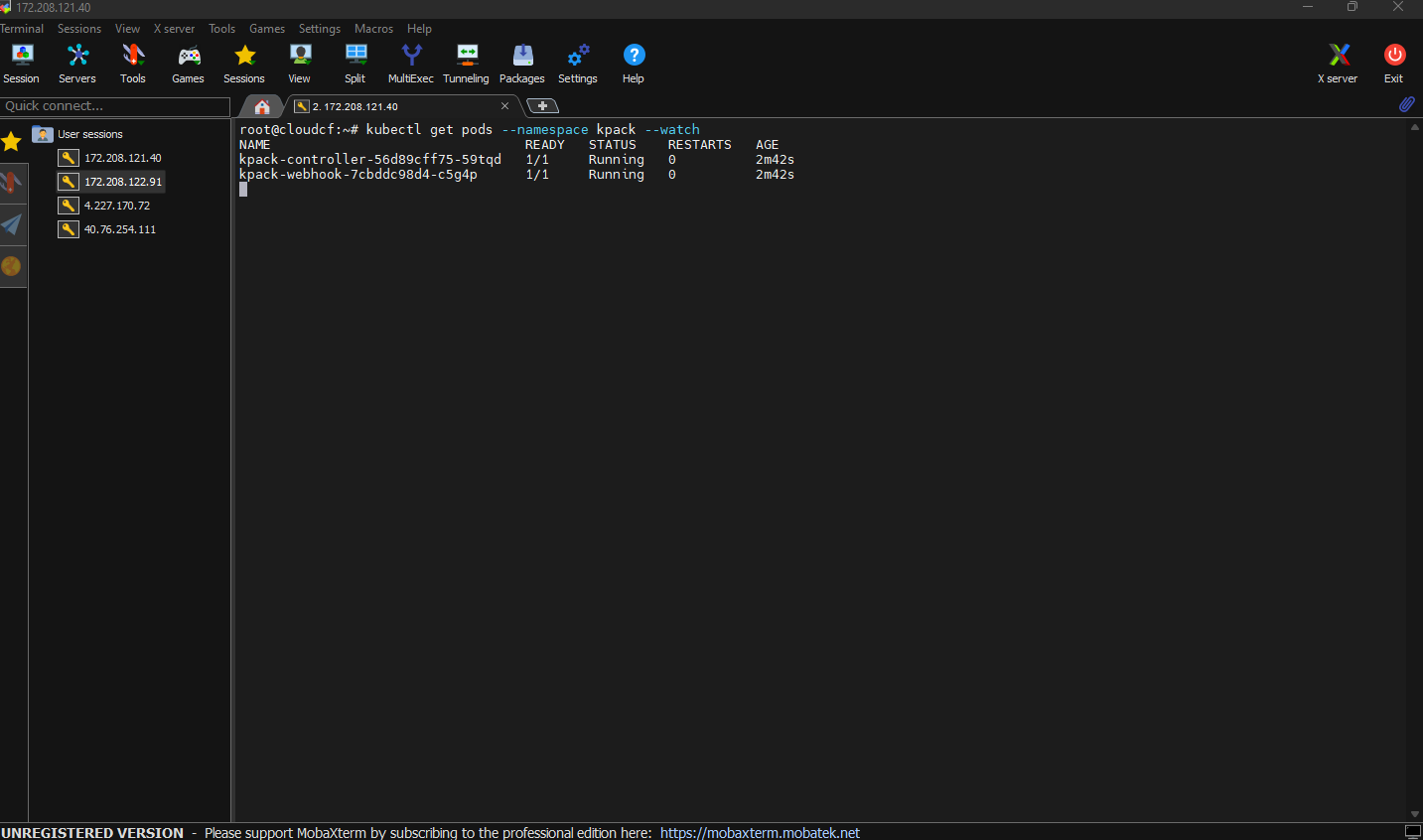
1. Steps to install cert-Manager

A screen shot of a computer screen

Description automatically generated kubectl apply -f <https://github.com/cert-manager/cert-manager/releases/download/v1.13.2/cert-manager.yaml>

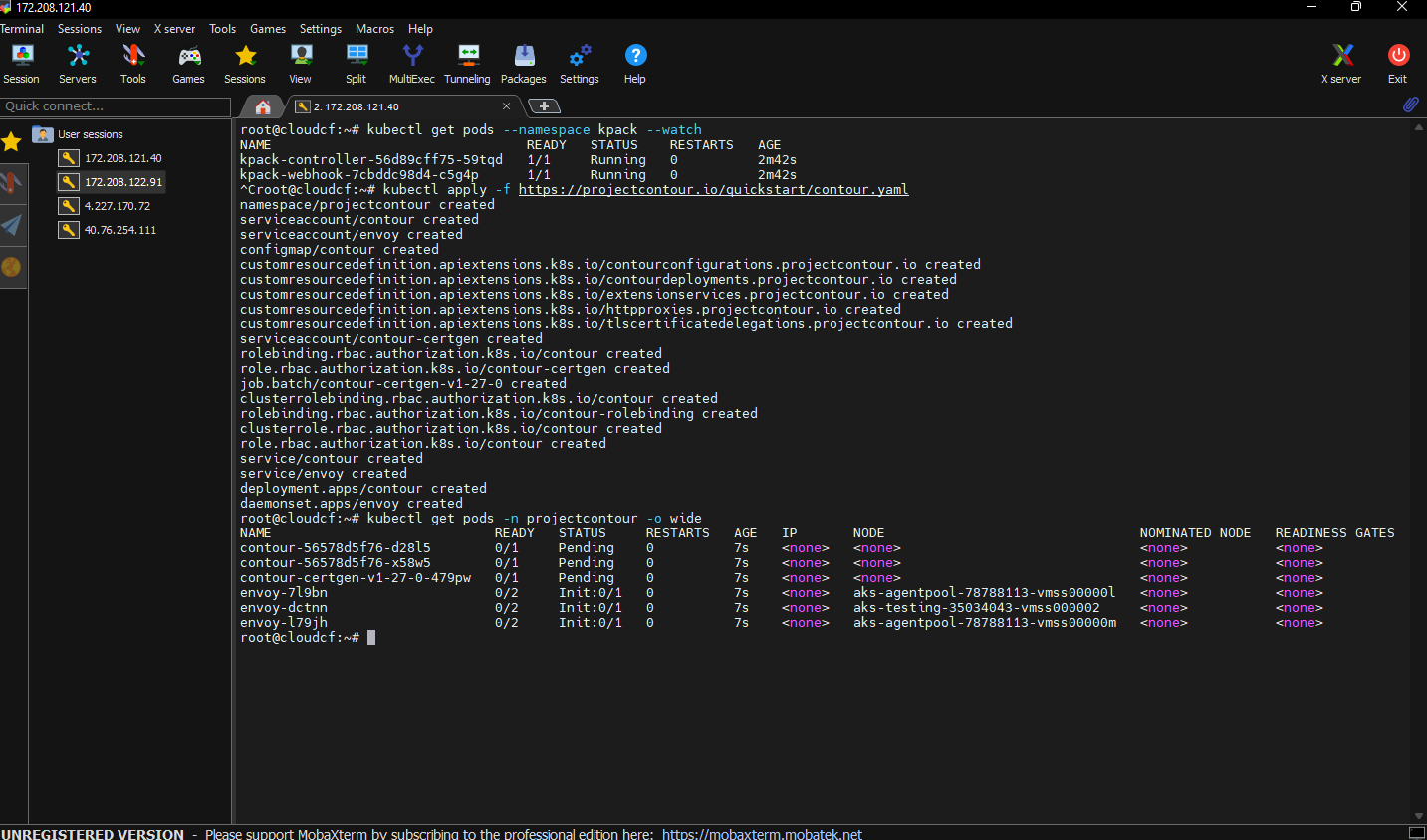
1. Steps to install kpack-

 kubectl apply -f <https://github.com/buildpacks-community/kpack/releases/download/v0.12.3/release-0.12.3.yaml> (replace the version as per your requirement)

 2. kubectl get pods --namespace kpack –watch

1. Steps to install contour

1. kubectl apply -f <https://projectcontour.io/quickstart/contour.yaml>

 2. kubectl get pods -n projectcontour -o wide

1. Upon installing Contour on the Kubernetes cluster, it will generate external-facing IP addresses. This will allow us to access the cluster. The Contour service will need to be queried to ascertain the IP address we are going to map for the ingress into the cluster. The following command will help with that:

kubectl get service envoy -n projectcontour -ojsonpath='{.status.loadBalancer.ingress[0]}'

A screenshot of a computer

Description automatically generated

Note: The output from this command will be an IP address which will be used at various places as the base domain, suffixed with nip.io.

1. Install CF cli:

<https://docs.cloudfoundry.org/cf-cli/install-go-cli.html>

Run the following commands to install cf cli v8

1. wget -q -O - https://packages.cloudfoundry.org/debian/cli.cloudfoundry.org.key | sudo apt-key add –
2. echo "deb https://packages.cloudfoundry.org/debian stable main" | sudo tee /etc/apt/sources.list.d/cloudfoundry-cli.list
3. sudo apt-get update
4. sudo apt-get install cf8-cli

A screen shot of a computer screen

Description automatically generated

The following environment variables will be needed throughout this guide:

ROOT\_NAMESPACE: the namespace at the root of the Korifi org and space hierarchy. The default value is “cf”.

KORIFI\_NAMESPACE: the namespace in which Korifi will be installed. We will use “korifi” for this setup.

ADMIN\_USERNAME: the name of the Kubernetes user who will have CF admin privileges on the Korifi installation. We will use “masterclient” for this setup

BASE\_DOMAIN: the base domain used by both the Korifi API and, by default, all apps running on Korifi.

Here are the example values we'll use in this guide:

ROOT\_NAMESPACE="cf"

KORIFI\_NAMESPACE="korifi"

ADMIN\_USERNAME=" masterclient "

BASE\_DOMAIN="<IPAddress>.nip.io"

A screen shot of a computer

Description automatically generated

Note: We use nip.io as the suffix for the externally available IP address that can reach the cluster. Nip.io is a wildcard DNS provider.

***Namespace creation:***

Create the root and korifi namespaces:

cat <<EOF | kubectl apply -f -

apiVersion: v1

kind: Namespace

metadata:

name: $ROOT\_NAMESPACE

labels:

pod-security.kubernetes.io/audit: restricted

pod-security.kubernetes.io/enforce: restricted

EOF

cat <<EOF | kubectl apply -f -

apiVersion: v1

kind: Namespace

metadata:

name: $KORIFI\_NAMESPACE

labels:

pod-security.kubernetes.io/audit: restricted

pod-security.kubernetes.io/enforce: restricted

EOF

A screenshot of a computer

Description automatically generated

The installation requires a container registry to function. For this installation, we will be using the Azure Container registry. To access this container registry, first, a token and a password have to be generated on the cloud.

Next, the same credentials will be used to create a secret and added to the cluster. The command for creating the registry credentials is as follows:

kubectl --namespace "$ROOT\_NAMESPACE" create secret docker-registry image-registry-credentials --docker-server="azure-registry-url" --docker-username="azure-registry-token-name" --docker-password "azure-registry-token-password"

A screenshot of a computer program

Description automatically generated

Note: Edit your azure registry credentials for docker-server,docker-username,docker-password

Once the secret has been created, use the following Helm chart to install Korifi on the Azure cluster.

Note: We use nip.io as the suffix for the externally available IP address that can reach the cluster. Nip.io is a wildcard DNS provider.

**Installing Korifi by using the following helm chart**

Replace <AzureContainerRegistryURL> with the actual URL of your ACR.

helm install korifi https://github.com/cloudfoundry/korifi/releases/download/v0.9.0/korifi-0.9.0.tgz --namespace="$KORIFI\_NAMESPACE" --set=generateIngressCertificates=true --set=rootNamespace="$ROOT\_NAMESPACE" --set=adminUserName="$ADMIN\_USERNAME" --set=api.apiServer.url="api.$BASE\_DOMAIN" --set=defaultAppDomainName="apps.$BASE\_DOMAIN" --set=containerRepositoryPrefix=<AzureContainerRegistryURL>/korifi/korifi/ --set=kpackImageBuilder.builderRepository=<AzureContainerRegistryURL>/korifi/kpack-builder --wait

***Test Korifi:***

Once installation is completed, use the cf cli to set the API endpoint and log to the cluster:

cf api http://api.$BASE\_DOMAIN --skip-ssl-validation

cf login

cf create-org org1

cf create-space -o org1 space1

cf target -o org1

cd <directory of a test cf app> (~/sandbox/korifi/tests/smoke/assets/test-node-app/ can be used)cf push test-app

***Sample Apps for pushing into cloud foundry:***

Wget <https://gitlab.com/4k1nt4yo/Exchange-Value/-/archive/master/Exchange-Value-master.zip>

Note: 1.Go to the folder where server.js and package.json files are present.

Inside the folder we can run command cf push testappname

2. If it is zip file then unzip it.

Wget <https://github.com/cloudfoundry/cf-for-k8s/archive/refs/heads/develop.zip>