TECHNICAL WHITE PAPER: **November 2026**



Akuity Platform on vSphere Kubernetes Service on VMware Cloud Foundation

Reference Architecture

Table of Contents

Executive Summary 3
vSphere Kubernetes Service 3
Solution Architecture 4
Akuity Platform Architecture on VKS 4
VKS Cluster Configuration 5
Software Requirements 5
Solution Validation 6
Install Akuity on VKS 6
Validate Akuity Platform install on VKS 11
Conclusion 13

Executive Summary

Akuity is the enterprise software delivery platform built by the creators of Argo CD and Kargo. It helps platform engineering and application teams deploy, promote, and operate applications at scale. Platform teams define shared delivery workflows, release checks, permissions, and approval requirements. Application teams use those workflows to deploy changes, promote releases across environments, and investigate production issues with fewer handoffs. By bringing application state and delivery history together, Akuity helps teams understand what is running and what changed.

Organizations using VMware Cloud Foundation can combine the Kubernetes infrastructure provided by vSphere Kubernetes Service (VKS) with Akuity's application delivery capabilities. This paper describes how to install and verify the self-hosted Akuity Platform on VKS, enabling organizations to run their delivery platform within infrastructure they control. The reference deployment provides a foundation for repeatable application delivery while retaining existing CI systems and familiar Kubernetes and GitOps workflows.

vSphere Kubernetes Service

vSphere Kubernetes Service (VKS) is the Kubernetes runtime built directly into VMware Cloud Foundation (VCF). With CNCF certified Kubernetes, VKS enables platform engineers to deploy and manage Kubernetes clusters while leveraging a comprehensive set of cloud services in VCF. Cloud admins benefit from the support for N-2 Kubernetes versions, enterprise grade security, and simplified lifecycle management for modern apps adoption.

Benefits of using VKS with modern applications:

Lower TCO: With VKS organizations can reduce silos, leverage existing tools and skill sets without having to retrain staff and/or change existing processes. Utilizing unified lifecycle management across infrastructure components to stay up to date with the most recent patches and minimizing security risks.

Operational Simplicity: VKS is engineered for unparalleled operational simplicity, leveraging the familiarity of existing vSphere tools, skills, and workflows. This design philosophy significantly reduces the learning curve for IT teams and streamlines management processes. With VKS, organizations benefit from automated cluster provisioning, which accelerates deployment times and minimizes manual configuration errors. Furthermore, its robust capabilities extend to automated upgrades and comprehensive lifecycle management. This integrated approach ensures consistent operations, reduces overhead, and frees up valuable resources to focus on innovation rather than infrastructure maintenance.

Run and Manage Kubernetes at Scale: Effortlessly deploy and manage Kubernetes clusters at scale, leveraging a built-in, Cloud Native Computing Foundation (CNCF) certified Kubernetes distribution. VKS provides fully automated lifecycle management, streamlining operations from initial setup to ongoing maintenance and upgrades. This comprehensive approach ensures that organizations can harness the power of Kubernetes for their containerized applications with unparalleled efficiency and reliability, without the complexities typically associated with large-scale Kubernetes deployments.

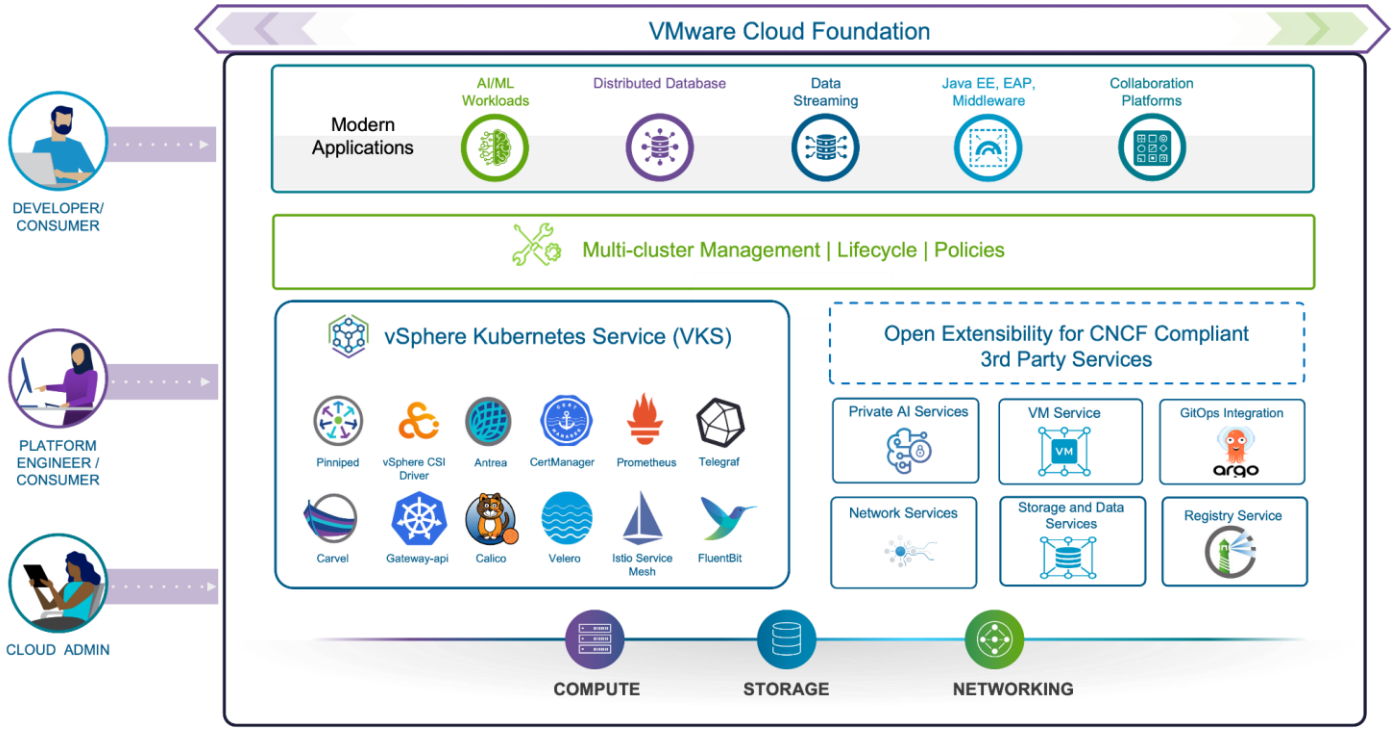


Figure 1: VKS on VCF Ecosystem

Solution Architecture

Akuity Platform Architecture on VKS

The self-hosted Akuity Platform runs on a dedicated VKS cluster and provides a central management layer for Argo CD and Kargo instances. Argo CD provides GitOps deployment by reconciling application configuration stored in Git with resources in target Kubernetes clusters. Kargo coordinates the promotion of application versions through environments using defined promotion workflows and verification. Existing CI systems build and test application artifacts; Argo CD and Kargo manage the subsequent deployment and promotion workflows. Teams can inspect application health, deployed versions, and delivery history to investigate changes and release issues.

In the installation described in this paper, Helm values and Kustomize configure the platform and its supporting components. PostgreSQL persists platform data, and an OpenID Connect (OIDC) identity provider authenticates users. The bundled Traefik ingress controller routes traffic to platform services behind a load balancer, with DNS and wildcard TLS certificates providing service endpoints. A compatible Kubernetes network policy engine enforces the platform's NetworkPolicy resources to restrict workload connectivity and cross-namespace traffic between tenants. The dedicated VKS cluster hosts the delivery services, while Argo CD manages application deployments to configured target clusters. The following sections detail the infrastructure prerequisites, installation steps, and verification checks for this deployment.

Each Kargo Instance runs on its own lightweight k3s API server. Developers work through kargo-api to define projects, stages and promotions. kargo-webhooks-server validates their changes, and kargo-management-controller handles instance-level management.

In the customer's environment, a self-hosted agent runs in each stage and production cluster. kargo-controller connects outbound through a security proxy and watches Git and image repositories for new versions. It then promotes those versions by updating Git or by triggering Argo CD, which is optional and self-hosted. rollouts-controller uses Prometheus metrics to verify each release. All connections start on the customer's side, so repository credentials and workloads stay inside their network.

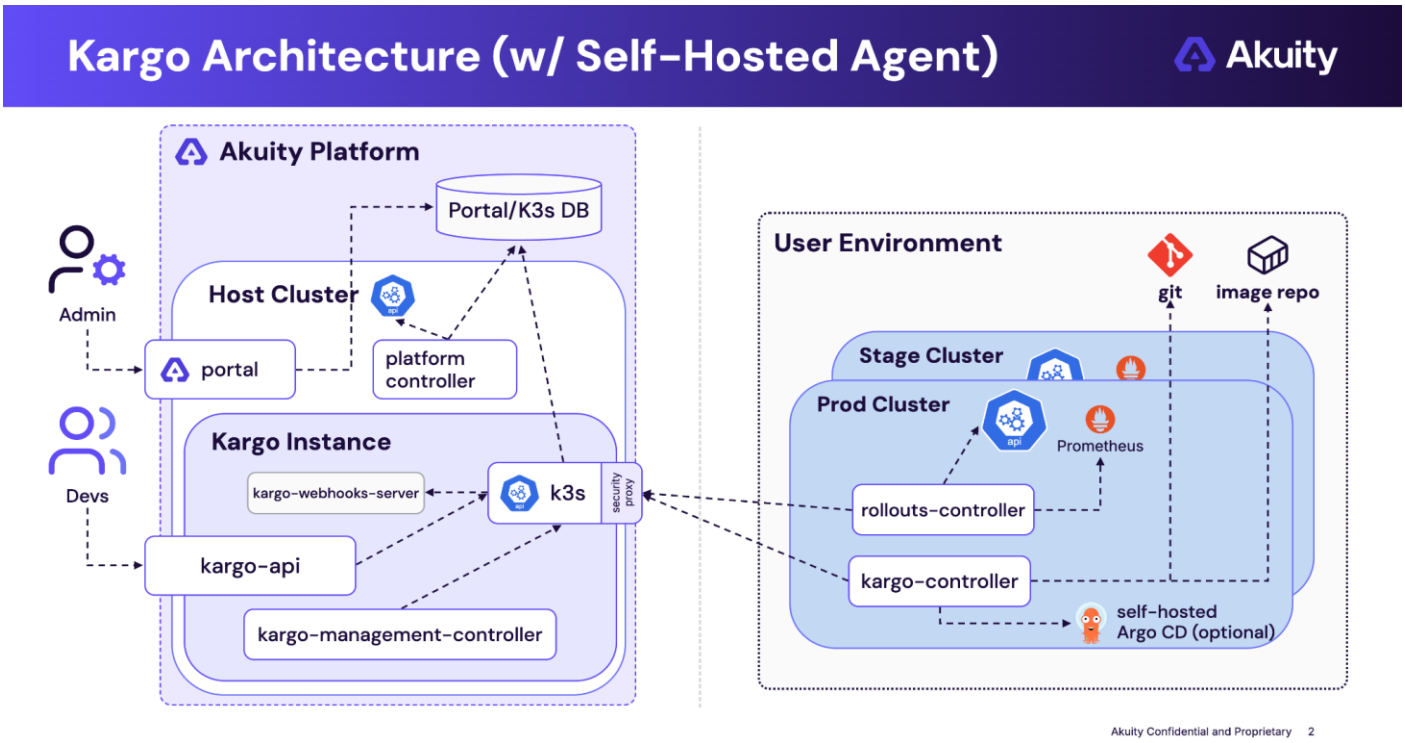


Figure 2: Akuity Platform Architecture on VKS

VKS Cluster Configuration

The cluster comprised of one control plane node and four worker nodes running Kubernetes v1.36.2+vmware.2-vkr.3 on Ubuntu 24.04. Storage was provided by vSAN Express Storage Architecture (ESA), using the default RAID 5 storage policy, which the vSphere CSI driver exposes as the storage class **vsan-esa-default-policy-raid5**.

Table 1: Kubernetes Cluster Configuration Summary

Component	Configuration	Notes
Kubernetes Control Plane	1 Replica: 2x vCPU and 8GB RAM	VM Class: “best-effort-medium” Storage Class: “vsan-esa-default-policy-raid5”
Kubernetes Worker Nodes	4 Replicas: 4x vCPU and 16GB RAM	VM Class: “best-effort-large” Storage Class: “vsan-esa-default-policy-raid5”
Kubernetes Release	v1.36.2+vmware.2-vkr.3	
OS Image	Ubuntu 24.04	Kernel version 6.17.0-35-generic

Software Requirements

Listed below is a set of software components required to successfully deploy Akuity Platform on VKS clusters. The software versions are current as of the validation testing and may not reflect the latest update.

Table 2: Kubernetes Cluster Software Versions

Software	Version
VKS	3.7.2
Akuity Platform	0.32.0
CloudNativePG	1.18.4
Kargo	1.11.0
Argo CD	3.4.5
K3s	1.33.8
Traefik	3.3.6

Solution Validation

Install Akuity on VKS

This is a comprehensive guide that will walk you through the process of setting up and using the Akuity Platform on VKS.

Prerequisites

Before starting, you will need two key pieces of information:

- Your Akuity Platform license key
- A key for the Akuity Platform OCI registry

Logging into the registry

First, use the license key to login to the helm registry.

```
# This will prompt you for your registry key
helm registry login us-docker.pkg.dev -u _json_key_base64
```

Infrastructure requirements

The Akuity Platform is designed to run on a dedicated Kubernetes cluster. Generally speaking, any cluster, no matter where it is hosted, you will need the following infrastructure available in addition to the Kubernetes cluster:

- A PostgreSQL database accessible from the Kubernetes cluster
- Kubernetes network policy engine (e.g. Calico, Cilium, etc.)
- A network-level load balancer to route traffic to the platform
- DNS configured to point traffic to the load balancer
- An OIDC identity provider for authentication (e.g. Google, Okta, etc.)
- A wildcard TLS certificate for the domain you are using for the platform (or the ability to generate one on the fly using cert-manager)
- SMTP (optional)
- A cluster autoscaler (optional)

Network Security

Akuity Platform deploys NetworkPolicy resources in order to ensure ingress and egress connectivity to Pods are allowed on an as-needed basis, and to disallow cross-namespace communication between tenants. For this to work, a Kubernetes network policy engine must be installed to enforce these policies.

Akuity recommends **Calico** for network policies in an On-Prem cluster, but any Kubernetes network policy engine that is compatible with your Kubernetes cluster should work.

Kubernetes Autoscaling

As Argo CD instances are created, and as more clusters connect to Akuity Platform, additional capacity is needed in the host control-plane cluster. Akuity Platform has no direct dependency on a specific autoscaler, but deploys HPA resources to automatically scale based on CPU/memory load. A Kubernetes cluster autoscaler allows for automatically adding compute capacity to the cluster.

Kubernetes Nodes

If auto-scaling is not used, then we recommend having a minimum of 3 nodes with a minimum of 2 vCPUs and 8GB RAM for each node. This number is highly variable and dependent on the number of applications, number of control planes managed by AKP, etc. This is something which you will need to keep a close eye on and tune accordingly depending on your workload.

PostgreSQL Database

Akuity Platform requires a PostgreSQL database to store its data. The database can be self-hosted or be a managed service. The database must be accessible from the Kubernetes cluster where the Akuity Platform is installed and securing the connection to the database is left to the user. No matter which option you choose, the database must be configured with the following requirements:

- Engine: PostgreSQL
- Version: 14.6+

The appropriate instance class/size is highly variable and depends on a number of factors:

- Number of Argo CD instances
- Number of total clusters across instances
- Number of applications across instances

Akuity recommends using a PostgreSQL database that meets the requirements above in addition to the following

Encryption enabled

Replica or Reader node in a different AZ/datacenter/rack for the purposes of high-availability failover and to provide read-only instances.

DB Instance Size:

The following are examples of configurations that have been tested:

Limits	DB Instance Class
10 instances, 10 clusters, 100 applications	2 vCPU, 4 GiB RAM
5 instances, 1000 Clusters, 5000 Applications	4 vCPU, 32 GiB RAM

Non-Superuser Permissions

If your database user is not running as a superuser or equivalent (e.g., rds_superuser on RDS), you will need to grant additional permissions. The user must have the CREATEROLE privilege and ability to create schemas, as well as being allowed to GRANT connect privileges to other users.

Substitute admin_user for the actual role you intend to use. The example below assumes the installation of all the database objects to the akuity schema; the Akuity Platform default is public for legacy reasons but new installations should install the Akuity Platform to a schema other than public

```
-- CREATEROLE is a requirement since we need to create a user per instance.
CREATE USER [redacted] WITH PASSWORD '<your password>' [redacted]
```

```
-- We need this to be able to perform database migrations.
GRANT CREATE ON DATABASE <your database name> TO admin_user;

-- The admin user also needs to be able to GRANT CONNECT to the provisioned instance user, so
the provisioned user can actually connect to the database.
GRANT CONNECT ON DATABASE <your database name> TO admin_user WITH GRANT OPTION

GRANT USAGE ON SCHEMA akuity TO admin_user;
GRANT CREATE ON SCHEMA akuity TO admin_user;
GRANT ALL ON ALL TABLES IN SCHEMA akuity TO admin_user;
GRANT ALL ON ALL SEQUENCES IN SCHEMA akuity TO admin_user;
GRANT ALL ON ALL FUNCTIONS IN SCHEMA akuity TO admin_user;

ALTER DEFAULT PRIVILEGES IN SCHEMA akuity GRANT ALL ON TABLES TO admin_user;
ALTER DEFAULT PRIVILEGES IN SCHEMA akuity GRANT ALL ON SEQUENCES TO admin_user;
ALTER DEFAULT PRIVILEGES IN SCHEMA akuity GRANT ALL ON FUNCTIONS TO admin_user;
```

Load Balancing and Ingress

The Akuity Platform installation bundles Traefik Ingress Controller as part of its installation. Upgrades of Traefik and CRDs will be handled as part of upgrades of Akuity Platform.

If using an on-prem cluster, that cluster will need to be able to create a LoadBalancer Service that an external load balancer can route traffic to.

TLS

The default behavior of Akuity Platform requires a public certificate and private key of a wildcard TLS certificate. TLS is needed to terminate inside cluster at Ingress/Traefik – not at the NLB. This is due to the fact that, by default, Akuity Platform leverages features of Traefik which require passthrough behavior to Ingress (e.g. IP allowlist, websocket support, HTTP2/gRPC).

The wildcard certificate given to Akuity Platform must be valid for the following domains (replace akuity.example.com with your own domain):

- akuity.example.com
- *.cd.akuity.example.com
- *.cdsvcs.akuity.example.com
- *.kargo.akuity.example.com
- *.kargosvcs.akuity.example.com

Akuity Platform does not have any requirements on how the certificate is provisioned. TLS certificates can be obtained in different ways. Some options include:

Purchased from a traditional certificate provider (e.g. Verisign, DigiCert)

Free, time-limited (90 day) certificates from Let's Encrypt

If the TLS Secret is managed separately and not supplied to the Helm chart (e.g. generated by cert-manager, deployed via External Secret) the resulting secret should be named akuity-platform-tls in the traefik-external namespace. Akuity Platform will assume the presence of this Secret so that it can be referenced by Traefik ingress.

Cert-Manager (optional)

If you are using cert-manager to manage your TLS certificates, the following example shows how to do so with a Let's Encrypt issuer. These instructions can be adapted for other issuers as well.

```
apiVersion: cert-manager.io/v1
```

```
kind: Issuer
metadata:
  name: letsencrypt-prod
spec:
  acme:
    email: admin@example.com
    privateKeySecretRef:
      name: letsencrypt-prod
    server: https://acme-v02.api.letsencrypt.org/directory
    solvers:
      - dns01:
          route53:
            region: us-east-1
        selector:
          dnsZones:
            - akuity.example.com
            - cd.akuity.example.com
            - cdsvcs.akuity.example.com
            - kargo.akuity.example.com
            - kargosvcs.akuity.example.com

apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: akuity.example.com
  namespace: akuity-platform
spec:
  dnsNames:
    - akuity.example.com
    - '*.cd.akuity.example.com'
    - '*.cdsvcs.akuity.example.com'
    - '*.kargo.akuity.example.com'
    - '*.kargosvcs.akuity.example.com'
  issuerRef:
    name: letsencrypt-prod
  # This secret name is important, as it is the name that Akuity Platform expects
  secretName: akuity-platform-tls
```

DNS

Akuity Platform requires DNS to be configured for the domain you are using for the platform. When the platform creates resources:

Argo CD instances will be provisioned with human-friendly and reconfigurable DNS names like:

- myargocd-1.cd.akuity.example.com
- myargocd-2.cd.akuity.example.com

Agent components will connect to static, ID specific DNS names like:

- x8ax38dmb1d81h-cplane.cdsvcs.akuity.example.com
- x8ax38dmb1d81h-cache.cdsvcs.akuity.example.com

Configuring DNS is platform specific and instructions are specified below.

```
apiVersion: v1
kind: Service
metadata:
```

```
  name: traefik
  namespace: traefik-external
  annotations:
    external-dns.alpha.kubernetes.io/alias: "true"
    external-dns.alpha.kubernetes.io/hostname: |
      '*.cd.akuity.example.com.,*.cdsvcs.akuity.example.com.'
```

Akuity Platform requires a DNS Zone (e.g. akuity.example.com) with the following records pointing to the load balancer hostname/IP:

Record Name	Type	Value/Route traffic to
akuity.example.com	A	Traefik Load Balancer Service
*.cd.akuity.example.com	A	Traefik Load Balancer Service
*.cdsvcs.akuity.example.com	A	Traefik Load Balancer Service
*.kargo.akuity.example.com	A	Traefik Load Balancer Service
*.kargosvcs.akuity.example.com	A	Traefik Load Balancer Service

OIDC Identity Provider

Akuity Platform requires login via an OIDC-compliant identity provider (IdP) using OAuth 2.0 login flow. Akuity Platform can be configured to integrate directly with an OIDC provider, or through Dex as an intermediary. Dex is bundled as an optional component of the helm chart. You may wish to enable Dex for the following reasons:

- Dex supports local, static users/passwords with dummy email addresses. This is useful for getting started, reducing upfront requirements, and for testing purposes.
- If obtaining a client application from your organization's IdP is a difficult process or not possible, Dex can be used as an IdP to other services you may have more access to, such as GitHub.
- If you are using Azure AD and you wish to limit Akuity Platform authentication through Azure AD groups. Dex is able to perform translation of Azure AD groups to OIDC claims, and limit logins to particular groups.

SMTP

SMTP is used to send emails notifying users that they have been invited to an organization. It is optional. If not configured users will not be notified about a pending invite but can still navigate to their account page to accept an invite.

Installation

We recommend using Argo CD to manage the platform. However, there is a chicken and egg problem as you need to do the initial installation and creation of the manifests before they can be managed by Argo CD. To assist with this we have a scaffolding for bootstrapping your installation. To use it, run the following command in a directory where you want to keep the manifest files:

```
curl -fsSL -o quickstart.zip https://dl.akuity.io/self-hosted/quickstart.zip && unzip
quickstart.zip && rm quickstart.zip

# Feel free to rename the directory if you want before continuing
cd quickstart
```

```
helm pull oci://us-docker.pkg.dev/akuity/akp-sh/charts/akuity-platform --untar --untardir
charts
```

This directory contains a values.yaml file with some of the most common configuration items available for you to customize. You will need to edit this file and fill out the details specific to your installation. If you need to make additional customizations that are not supported by the chart, you can use the kustomization.yaml file to make those last-mile modifications. Once you have configured your values and customizations, use the following command to deploy the Akuity Platform:

```
kustomize build --enable-helm | kubectl apply -f -
```

Once you have run this command, the Akuity Platform will be installed in your cluster. You can also commit the manifests to your Git repository and use Argo CD to manage the platform going forward.

Validate Akuity Platform install on VKS

When installation is complete, there are a number of ways to verify the environment is working.

Kubectl verification of namespaces, pods, and workloads

Nodes

To verify the creation of the various nodes for Akuity Platform, run the following command

```
kubectl get nodes
```

The output should contain a number of nodes, similar to the following:

```
NAME                                  STATUS   ROLES           AGE   VERSION
akuity-c1-fq65w-vf9hr                 Ready    control-plane   64d   v1.36.2+vmware.2
akuity-c1-node-pool-1-zw2zl-dtrzn-6vl8k   Ready    <none>          64d   v1.36.2+vmware.2
akuity-c1-node-pool-1-zw2zl-dtrzn-cxbzh   Ready    <none>          64d   v1.36.2+vmware.2
akuity-c1-node-pool-1-zw2zl-dtrzn-f24b2   Ready    <none>          64d   v1.36.2+vmware.2
akuity-c1-node-pool-1-zw2zl-dtrzn-wtkbr   Ready    <none>          64d   v1.36.2+vmware.2
```

Figure 3: VKS Cluster Nodes Reported by kubectl get nodes

Namespaces

```
kubectl get namespace
```

Verify that the output of the above command contains at least “akuity” and “akuity-platform”.

Pods

```
kubectl get pods -n akuity-platform
```

The output should look similar to the following:

```
NAME                                    READY   STATUS      RESTARTS   AGE
addon-controller-76c556464b-zgcjq       1/1     Running     0          34d
dex-b5dc7bcd5-gjfx5                     1/1     Running     0          62d
event-cleaner-29884320-svbqz            0/1     Completed   0          22h
kubevision-event-cleaner-29885660-gb9c4 0/1     Completed   0          10m
litellm-7d59b88f4c-p4jp5                1/1     Running     0          34d
litellm-7d59b88f4c-pcfc6                1/1     Running     0          34d
platform-controller-78c7897f58-df4jz    1/1     Running     0          34d
portal-server-5468459959-86p2f          1/1     Running     0          34d
portal-server-5468459959-bbqsx          1/1     Running     0          34d
portal-server-5468459959-qwzj7          1/1     Running     0          34d
records-aggregator-29884380-k75b2       0/1     Completed   0          21h
```

Figure 4: Akuity Platform Pods in the akuity-platform Namespace

Validate in browser by logging in to the platform

Open a browser and navigate to the DNS which you used when installing self-hosted Akuity Platform.
You will be presented with a Dex login screen:

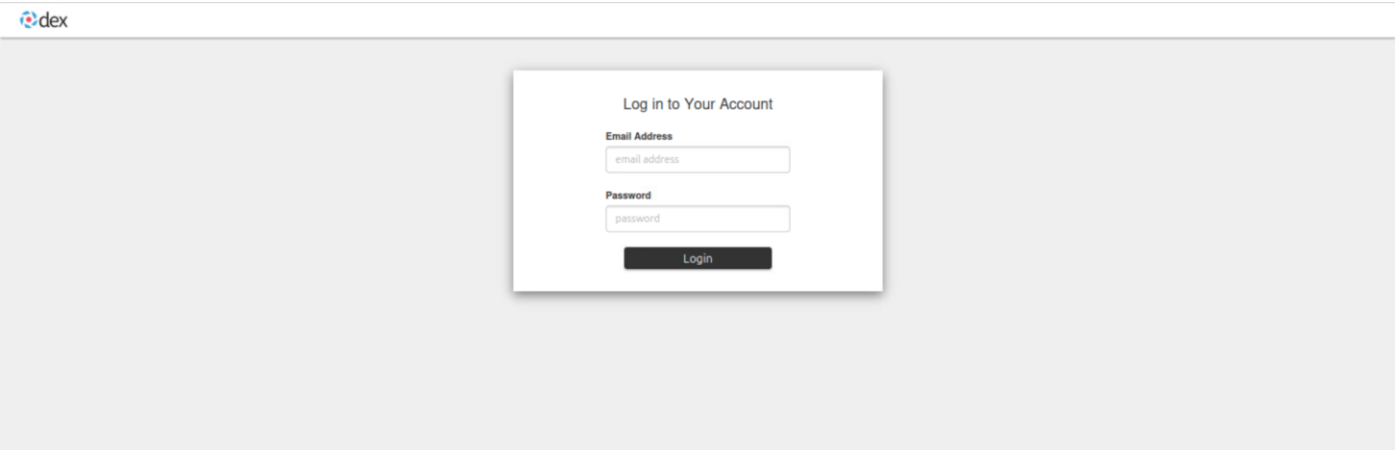


Figure 5: Dex Login Screen for the Akuity Platform

Enter the credentials for the account which was set up using the values.yaml file.
The default test account, if you didn't change it, is:

- Email Address: admin@example.com
- Password: password

You will now be taken to the Akuity Platform account dashboard, where you can begin working with your Akuity Platform instance to manage Argo CD & Kargo Instances.

Argo Verification

When you set up instances for Argo CD, you should have a "Deploy" section which looks like this:

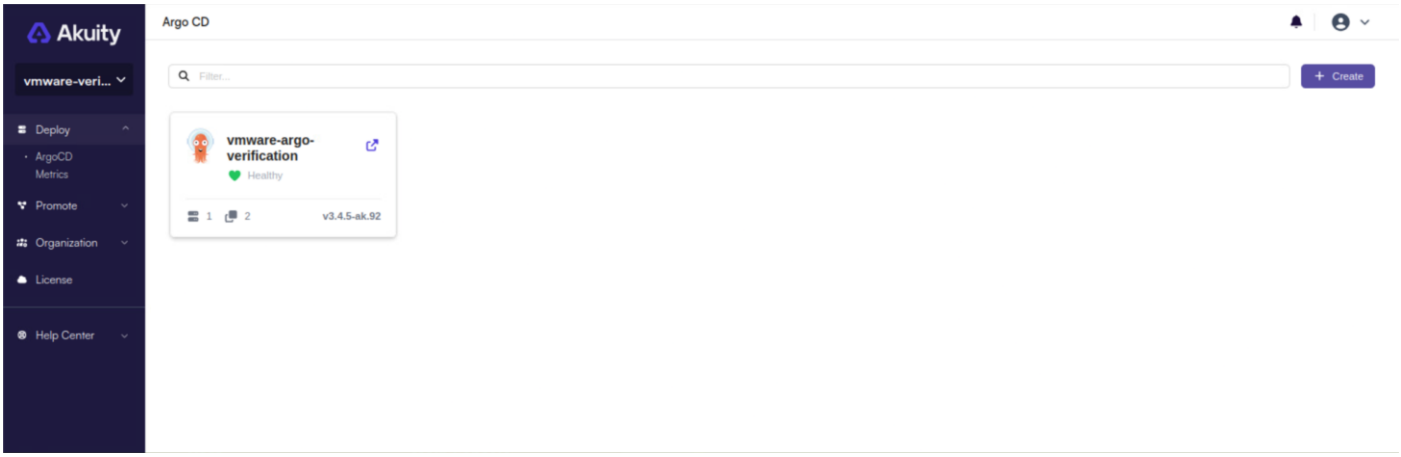


Figure 6: Argo CD Instance in the Akuity Platform Deploy View

Kargo Verification

When you set up instances for Kargo, you should have a “Promote” section which looks like this:

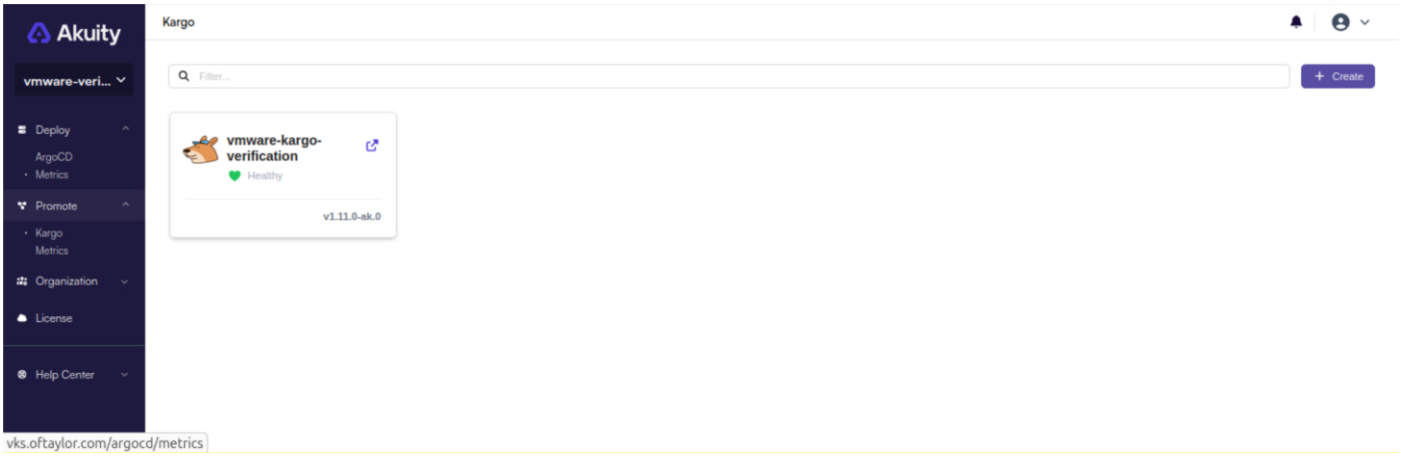


Figure 7: Kargo Instance in the Akuity Platform Promote View

Conclusion

Akuity Platform is an enterprise software delivery platform that helps teams deploy, promote, and operate applications through shared workflows and defined production controls. Platform teams define delivery workflows, release checks, permissions, and approval requirements. Application teams use those workflows to deploy changes, promote releases across environments, and investigate production issues with fewer handoffs. Built by the creators of Argo CD and Kargo, Akuity combines familiar open-source foundations with enterprise capabilities and expert support. The self-hosted installation described in this guide enables organizations to run Akuity Platform on vSphere Kubernetes Service (VKS) with VMware Cloud Foundation. Its compatibility with VKS offering makes for a powerful combination of tools and infrastructure, which could help teams get back to the business of what they are great at, building world class applications, instead of front loading the effort into infrastructure management.



Copyright © 2026 Broadcom. All rights reserved.
The term "Broadcom" refers to Broadcom Inc. and/or its subsidiaries. For more information, go to www.broadcom.com. All trademarks, trade names, service marks, and logos referenced herein belong to their respective companies. Broadcom reserves the right to make changes without further notice to any products or data herein to improve reliability, function, or design. Information furnished by Broadcom is believed to be accurate and reliable. However, Broadcom does not assume any liability arising out of the application or use of this information, nor the application or use of any product or circuit described herein, neither does it convey any license under its patent rights nor the rights of others.
Item No: vmw-bc-wp-tech-uslet-word-2026; May-26