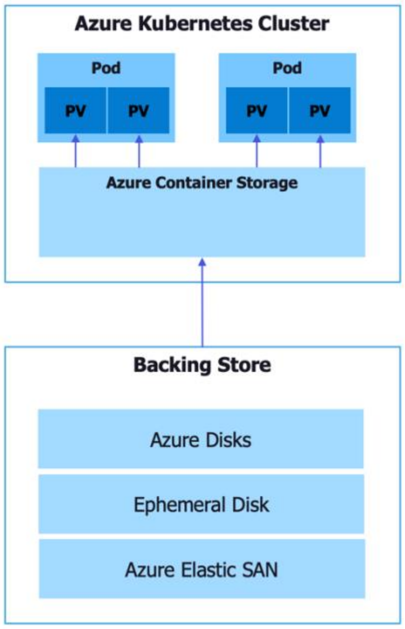
# AZURE CONTAINER STORAGE & KASTEN K10

## QUICK START GUIDE

# Kasten K10 & Azure Container Storage

Kasten by Veeam is excited to announce support for Azure Container Storage (ACS) on Azure Kubernentes Service (ACS). With Microsoft's recent preview of their new Azure Container Storage, which supports Volume Snapshot capabilities, Kasten by Veeam can backup, move, and restore workloads in a crash-consistent manner while embracing the flexibility and benefits of multiple block storage offerings while using a single management interface.

Until now, storage offerings have been primarily focused for IaaS offerings, both from a cost and performance perspective. With the growing adoption of persistent data workloads within Kubernetes, Microsoft recognizes a need for a more optimized and flexible storage solution for containerized workloads. With the release of Azure Container Storage (ACS), customers gain the benefits of:



- Optimized Storage offering for containerized Kubernetes Workloads
- Support for multiple block storage offerings using a single driver
- Optimized Persistent Volume (PV) placement for cost optimization

Microsoft has developed a single driver, following the CSI standards and K8s-native tools and CRDs to enable organizations to leverage different backing storage - containerstorage.csi.azure.com.

The backing storage options are highlighted in the table below:

| | **Ephemeral Disk** | **Azure Disks** | **Managed (Azure Elastic SAN)** |
|---|---|---|---|
| **Workloads** | Extremely latency sensitive (low sub-ms), Applications with no data durability requirement or built-in data replicatin support (e.g. Cassandra) | Tier-1 and general purpose databases (e.g. MySQL, MongoDB, PostgreSQL) | General purpose databases, streaming and messaging services, CI/CD Environments, and other Tier-1 and Tier-2 workloads |
| **Offerings** | Temporary Storage, NVMe | Premium SSD v2, Premium SSD | Azure Elastic SAN |
| **Provisioning model** | Deployed as part of the cluster node VMs in AKS, ACS discovers available ephemeral storage and acquires them for deployment | Provisioned per target container storage pool size and max volume size Minimal Footprint: 1TiB | Provisioned on-demand, per created volume and volume snapshot<br><br>Minimal Footprint: 1TiB |

Below is an example of how to deploy and leverage the new Microsoft Azure Container Storage offering with Kasten to backup and protect workloads:

1. Follow the Use Azure Container Storage with AKS Quickstart guide
2. Choose which backing storage you would like to use for ACS. In this example, we will use Azure Container Storage with Azure Disks
3. Create a Storage Pool per the guidance in the link above.
4. Patch the storage classes, setting the newly deploy acstor-azuredisk storage class as the default storage class:
```
kubectl patch storageclass default -p '{"metadata":
{"annotations":{"storageclass.kubernetes.io/is-default-
class":"false"}}}'
kubectl patch storageclass acstor-azuredisk -p '{"metadata":
{"annotations":{"storageclass.kubernetes.io/is-default-
class":"true"}}}'
```

5. Create a VolumeSnapshotClass corresponding to the Azure Container Storage driver containterstorage.csi.azure.com, setting the annotation for Kasten:
```
cat <<EOF | kubectl create -f -
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshotClass
metadata:
  name: csi-acstor-vsc
  annotations:
    k10.kasten.io/is-snapshot-class: "true"
driver: containerstorage.csi.azure.com
deletionPolicy: Delete
parameters:
  incremental: "true"
EOF
```

6. Run k10_primer to confirm all is configured correctly and Kasten can be deployed:

```
curl -s https://docs.kasten.io/tools/k10_primer.sh | bash
/dev/stdin csi -s acstor-azuredisk
```

7. Confirm the k10_primer script executed successfully and the CSI Snapshot Walkthrough succeeds in the output:

```
CSI Snapshot Walkthrough:
  Using annotated VolumeSnapshotClass (csi-acstor-vsc)
  Successfully tested snapshot restore functionality.  -  OK
```

8. Install Kasten K10 via helm:

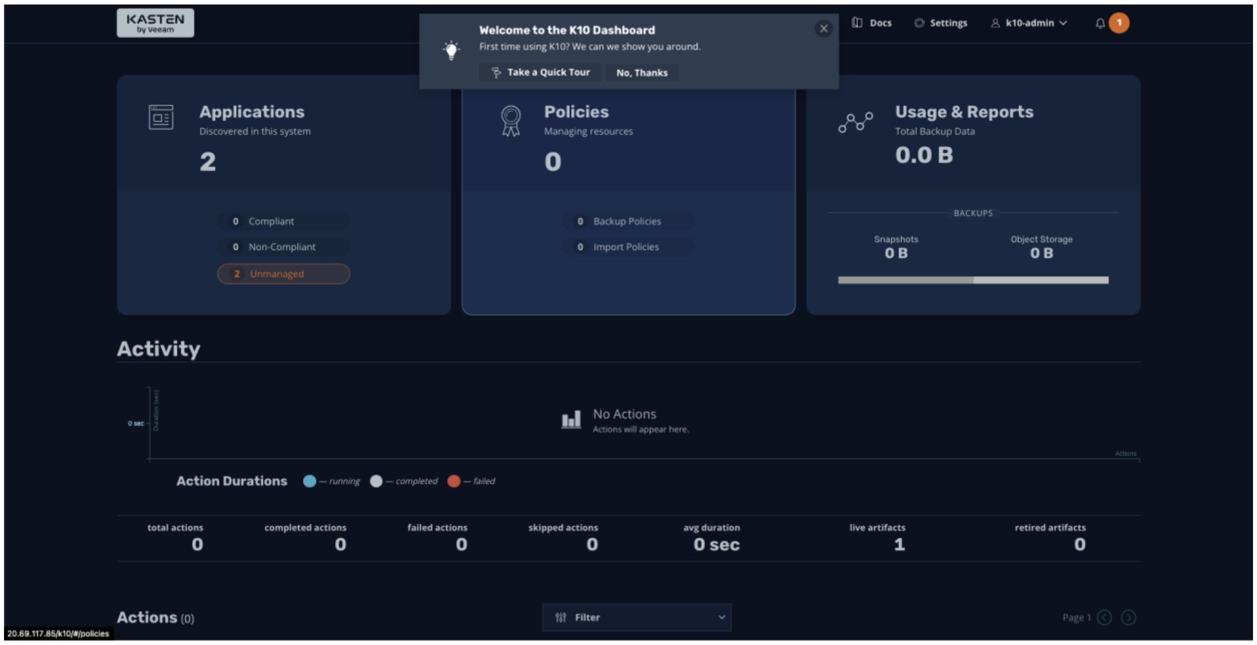
```
helm repo add kasten https://charts.kasten.io/
helm install k10 kasten/k10 --create-namespace -n kasten-io --set
externalGateway.create=true --set auth.basicAuth.enabled=true --
set auth.basicAuth.htpasswd='<htpasswd user and password>'
```

**Note:** htpasswd can be generated using an online generator, such as https://www.web2generators.com/apache-tools/htpasswd-generator

9. After a few minutes, all pods in the kasten-io namespace should start. Retrieve the external IP of the gateway-ext service:

```
kubectl get svc gateway-ext -n kasten-io
NAME           TYPE           CLUSTER-IP     EXTERNAL-IP
PORT(S)         AGE
gateway-ext    LoadBalancer   10.0.180.225   <external_ip>
80:30908/TCP    4m29s
```

10. In a browser, navigate to the Kasten K10 UI via the external IP: http://<external_ip>/k10/#
11. A username password and prompt should appear. Enter the credentials generated for the htpasswd used during the helm install above
12. Review the Terms of Service and if you agree, accept the terms
13. Congratulations! Kasten K10 is now deployed using Azure Container Storage and you’re ready to begin configuring Kasten K10 Location Profiles, Policies, etc and backing up workloads:



## Learn More

You can learn more about Kasten and why cloud native backup and recovery is the best way to protect your Kubernetes data and application workloads by reading our white paper, "5 Kubernetes Backup Best Practices."

To learn more about Microsoft Azure Container Storage, visit the Azure Container Storage Documentation.